AFSTUDEERVERSLAG

Niels van der Knaap
Capgemini – High Tech Center

Versie:	1.0
Datum wijziging:	14-06-2022
Auteur:	Niels van der Knaap
School:	Hogeschool Leiden
Studentennummer:	S1112367

Documenthistorie

Versie beheer

Datum	Versie	Beschrijving	Auteur
31-05-2022	0.1	Initiële versie	Niels van der Knaap
13-06-2022	1.0	Eerste publicatie	Niels van der Knaap

Relaties met andere documenten

1. Onderzoek_Xamarin_meets_MAUI_Capgemini_HTC_v1.0.pdf
2. Adviesrapport_Xamarin_meets_MAUI_Capgemini_HTC_v1.0.pdf
3. Software_architectuur_document_Xamarin_meets_MAUI.pdf
4. Testplan_Xamarin_meets_MAUI.pdf
5. Usecase_model_Xamarin_meets_MAUI.pdf
6. Vision_document_Xamarin_meets_MAUI.pdf

Voorwoord

Dit document is een afstudeerverslag dat een overzicht geeft over de volledige afstudeerperiode die ik bij het High Tech Center van Capgemini heb doorlopen. Dit verslag is geschreven in het kader van mijn afstuderen aan de opleiding informatica aan de hogeschool Leiden. De afstudeerstage heeft van 1 februari tot 1 juli geduurd waarbij ik gewerkt heb aan het onderzoek en andere bijproducten. Tijdens mijn meeloopstage kwam ik in aanraking met applicatie ontwikkeling voor mobiele apparaten. Dit beviel mij zo goed, dat ik opzoek ging naar een opdracht in deze sector. De opdracht die uitgevoerd is, sluit hierbij goed aan.

In het onderzoek dat uitgevoerd is in deze periode, is gekeken naar de ontwikkel- en performanceverschillen tussen twee ontwikkel frameworks voor mobiele applicaties. Hiervoor is zowel een kwantitatief als een kwalitatief onderzoek uitgevoerd. Tijdens dit onderzoek heb ik veel moeten leren over de verschillende frameworks. Hiervoor kon ik altijd hulp vragen aan mijn stagebegeleiders en de rest van het High Tech Center.

Mede dankzij hen is het gelukt om een mooi resultaat neer te zetten en een advies te geven aan het team. Bij deze wil ik het gehele team van het HTC bedanken voor hun hulp en gezelligheid. In het bijzonder wil ik mijn begeleiders, Wouter de Bruin en Bryan Slop bedanken. Ondanks dat de manager van het team en initiële afstudeerbegeleider is vertrokken, hebben ze mij heel goed kunnen begeleiden. Naast de goede begeleiding hebben ze mij ook het vertrouwen en de vrijheid gegeven om de opdracht naar eigen inzicht in te vullen. Dit heeft zeker bijgedragen aan het resultaat.

Vanuit hogeschool Leiden wil ik graag Heiko van der Heijden bedanken. Hij was deze periode mijn begeleider vanuit school. Ook al is er niet veel contact geweest tussen ons, was het contact wel altijd duidelijk. Daarnaast waren de spontane mails met de vraag hoe het ging, een positieve ervaring. Hierdoor kon ik mijn resultaten en uitdagingen ook met iemand delen buiten het team om.

~ Niels van der Knaap
Bergschenhoek, 9 juni 2022

Inhoudsopgave

Lijst van afkortingen

Afkorting	Betekenis
HTC	High Tech Center
CSD	Custom Software Development
CCA	Cloud and Custom Applications
ICT	Information and Communication Technology
API	Application Programming Interface
MAUI	Multi-platform App UI

Begrippenlijst

Begrip	Betekenis
Benchmark	Een vergelijkend onderzoek waarbij de prestaties van o.a. software op identieke wijze wordt onderzocht en vergeleken.
Benchmark applicatie	De mobiele applicatie waarop de benchmark wordt uitgevoerd.
C#	Een programmeertaal ontwikkeld en onderhouden door Microsoft.
.NET	Een cross-platform ontwikkel framework gebouwd met C#.

1 Samenvatting

Het High Tech Center (HTC) dat onderdeel is van Capgemini Nederland houdt zich voornamelijk bezig met het ontwikkelen van mobiele applicaties. Deze applicaties hebben vaak als eis om beschikbaar te zijn op meerdere platforms (iOS en Android). Om dit te realiseren maakt het HTC gebruik van Xamarin. Xamarin is een framework gebouwd op het .NET framework van Microsoft die het mogelijk maakt om met één codebase te ontwikkelen voor meerdere platforms.

Nu heeft Microsoft aangekondigd dat er een nieuwe framework komt genaamd .NET MAUI. Dit is een framework dat als evolutie van Xamarin wordt gezien en zal langzamerhand Xamarin gaan vervangen. Microsoft geeft aan dat .NET MAUI gebouwd is met performance en uitbreidbaarheid in gedachten. Ook hebben ze aangegeven dat er een aantal aanpassingen zijn gedaan die het ontwikkelproces kunnen versnellen.

Wanneer de claims van Microsoft kloppen, zal dit het ontwikkelproces van het team kunnen versnellen. Dit maakt het mogelijk om applicaties sneller en dus goedkoper te ontwikkelen. Daarnaast kan een performance verschil in de applicaties een positief effect hebben op de gebruikerservaring.

Om er achter te komen of deze claims kloppen is er een onderzoek opgezet. Dit onderzoek bestaat uit een kwantitatief en kwalitatief onderzoek. Hierbij zijn er allereerst interviews afgenomen bij de software ontwikkelaars van het HTC. Het doel van de interviews was inzicht krijgen in het ontwikkelproces en in hoe voorgaande applicaties eruit hebben gezien.

Het resultaat van de interviews over het ontwikkelproces is gecodeerd en er is gekeken welke processen volgens de ontwikkelaars beter of sneller kunnen. De punten die het meeste genoemd zijn waren: het missen van hot-reload en het toevoegen van afbeeldingen voor Android en iOS. Deze processen zijn in beide frameworks uitgevoerd en naast elkaar gezet. Op deze manier zijn de verschillen in kaart gebracht.

Om de performanceverschillen in kaart te brengen tussen de twee frameworks, zijn er benchmarks uitgevoerd. Deze benchmarks zijn uitgevoerd op een applicatie die dezelfde codebase en bibliotheken gebruikt in zowel een Xamarin als .NET MAUI versie. Op deze manier is de enige variabele het framework dat gebruikt is.

Hiervoor is een applicatie ontworpen en ontwikkeld. Het functioneel ontwerp van de applicatie is gebaseerd op de interviewresultaten die keken naar de eerder ontwikkelde applicaties door het team. Hieruit zijn de functionaliteiten meegenomen die het vaakst terugkwamen in de codering van de antwoorden. Op deze manier is de applicatie die gebruikt is voor de benchmark een reflectie zijn van een applicatie die het HTC zou hebben kunnen ontwikkeld.

De performance benchmarks zijn uitgevoerd op de resultaatapplicatie die gebouwd is vanuit het Xamarin en .NET MAUI project. Dit is gedaan op iOS en Android. Hierbij is gekeken naar opstarttijd, werkgeheugen gebruik & processor gebruik. Om dit proces te automatiseren en te zorgen dat het onderzoek eenvoudig te herproduceren is, zijn er scripts geschreven. Deze scripts maken gebruik van de Android Debug Bridge (ADB) en xCode om de applicatie op een apparaat te laden en deze op bovenstaande punten te meten. De gebruiker krijgt nadat de testen zijn uitgevoerd een samenvatting van de resultaten. De scripts maken het ook mogelijk om het proces te herhalen en van de herhalingen een gemiddelde te berekenen.

Het resultaat van het onderzoek is dat er een positieve verandering te zien is in het ontwikkelproces. Vooral het feit dat .NET MAUI hot-reload ondersteunt heeft een positief effect op de ontwikkeltijd. Tijdens het ontwikkelen van de benchmarkapplicatie scheelde dit 9 tot 12 seconden per aanpassing. Ook is het toevoegen van afbeeldingen eenvoudiger geworden waardoor je maar één vector bestand hoeft toe te voegen i.p.v. 47 variaties. Dit verkort dit proces en heeft invloed op de projectgrootte.

Ook bij de performanceverschillen was er bij Android een positief effect te zien. Hierbij was er een verbetering van de opstartsnelheid tot wel 18%. Ook waren het werkgeheugen en processorgebruik gedaald. Dit heeft een positief effect op de batterijconsumptie en snelheid van multitasken.

Op iOS was het een ander verhaal, hier was een verslechtering te zien bij de opstartsnelheid. Dit was 9% trager dan bij de Xamarin applicatie. Een verklaring hiervoor kan zijn dat het .NET MAUI team de iOS performance nog niet als prioriteit ziet. Deze aanname wordt versterkt doordat de artikelen vanuit het .NET MAUI team alleen maar ingaan op de performance van Android.

Wel is er een positief effect te zien op het werkgeheugen en processor gebruik van de iOS applicatie.

De resultaten van het onderzoek zijn gebruikt als basis van het adviesrapport. Het adviesrapport dat aan het HTC team gepresenteerd is, gaat in op drie scenario's:

1. Welk framework te gebruiken bij nieuwe applicaties?
2. Welk framework te gebruiken bij bestaande applicaties met zicht op end of life (EOL)?
3. Welk framework te gebruiken bij bestaande applicaties die nog lange tijd onderhoud worden?

Het advies is gepresenteerd aan het gehele team waarna de presentatie en verslag beschikbaar zijn gesteld aan het team.

2 Beschrijving stageorganisatie

Capgemini is een wereldwijde leider in consulting en ICT. Het bedrijf is in 1967 opgezet door de Franse entrepreneur Serge Kampf, toen nog onder de naam Sogeti. Na de oprichting volgen er verschillende overnames en fusies van bedrijven die actief waren in ICT. Door de verschillende fusies was de naam veranderd in Cap Gemini Sogeti. In 2004 hebben ze de naam versimpeld naar Capgemini en een volledige re-branding gehad. Inmiddels heeft het bedrijf meer dan 300.000 werknemers en is actief in bijna 50 landen *(Capgemini, 2017)*.

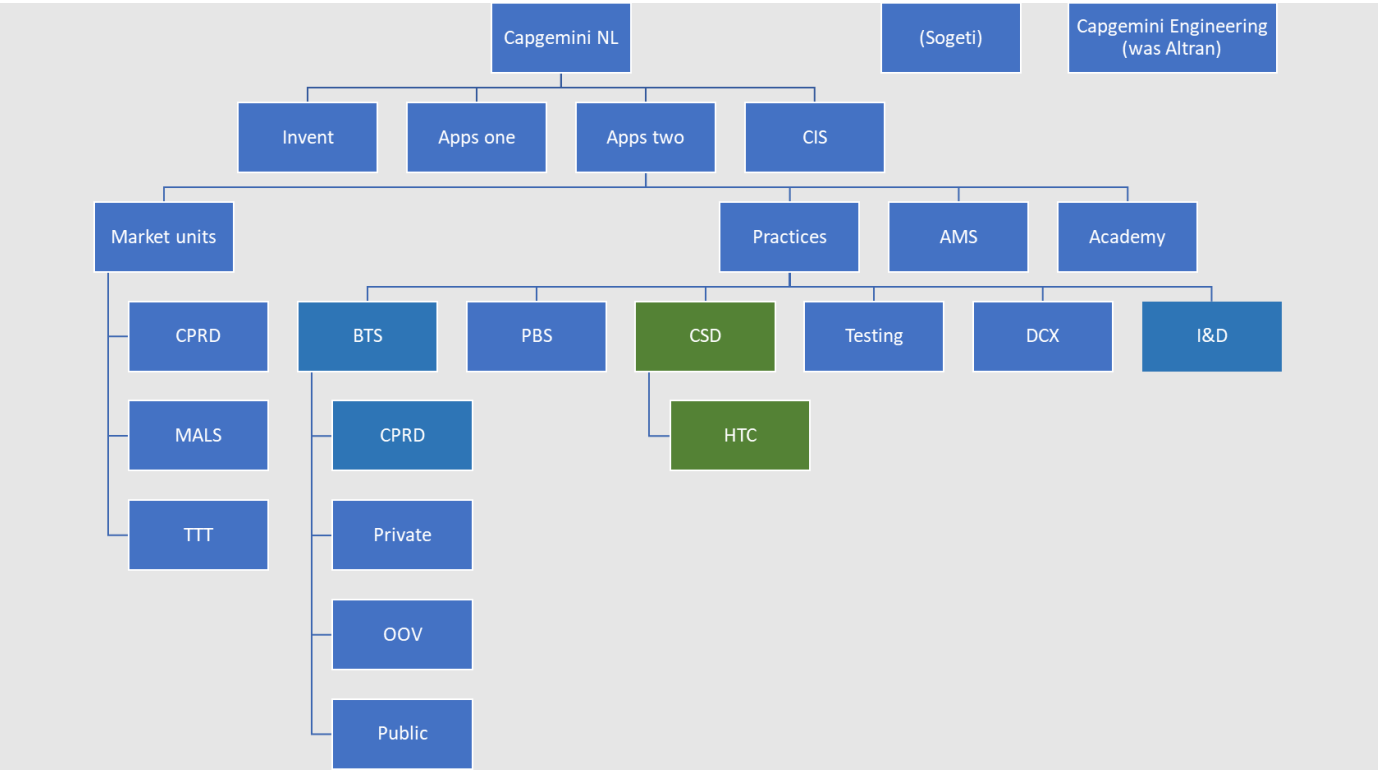
Tijdens het afstuderen is Niels onderdeel geweest van het High Tech Center (HTC). Dit is een team binnen Capgemini van ongeveer tien programmeurs, software architecten en testers. Dit team valt bij Capgemini onder CCA (Cloud and Custom Applications), wat voorheen CSD (Custom Software Development) was, in het Microsoft cluster. Naast CCA bestaat het cluster ook uit specialisten op het gebied van cloud omgevingen van Microsoft waar nauw mee samengewerkt wordt.

Dit team houdt zich voornamelijk bezig met het ontwikkelen van mobiele- en webapplicaties. Dit doen ze voor klanten van buitenaf maar ook voor projecten binnen Capgemini of hun eigen team. Tijdens het afstuderen is Niels actief in deelname bij de standups van het team zodat hij meekrijgt hoe het team opereert. Op deze manier krijgt Niels onder andere een inzicht in waar het team zich mee bezig houdt, hoe het team taken verdeelt en hoe ze met verschillende problemen omgaan.

Deze inzichten heeft Niels gedurende zijn afstuderen mee kunnen nemen in zijn eigen opdracht. De opdracht die Niels heeft gedaan stond volledig los van de opdrachten die lopen bij het team.

2.1 Organogram

Het HTC valt bij Capgemini onder het onderdeel "Custom Software Development" (CSD). Dit onderdeel van Capgemini houdt zich bezig met op maat gemaakte softwareoplossingen.



(Van der Liende & Capgemini, 2021)

2.2 Stakeholders

De stakeholders van deze opdracht zijn de personen werkzaam in het HTC (ongeveer 10 teamleden met specialiteiten in UX, Testen en software ontwerpen en ontwikkelen + HTC-manager). Dit omdat het uiteindelijke advies effect kan hebben op hun manier van werken. Om tot dit advies te komen moet er ook goed gekeken worden naar de huidige situatie.

3 Context

Het HTC (High Tech Center) van Capgemini ontwikkelt voornamelijk mobiele applicaties. Deze mobiele applicaties hebben vaak als eis om multi-platform beschikbaar te zijn (zowel voor iOS als Android). Om aan deze eis te voldoen zijn er drie typen van ontwikkelen *(Valdellon, 2020)*:

1. Native *(Valdellon, 2020, Hoofdstuk 01. Native Apps)*

Bij native ontwikkeling wordt de applicatie ontwikkeld in de taal van het platform waarvoor je ontwikkelt. Voor iOS is dit Swift of Objective-C en voor Android is dat Java of Kotlin. Door het ontwikkelen voor de specifieke platforms maak je gebruik van de user interface elementen van het platform waardoor de applicatie sneller zal opstarten en soepeler loopt. Het nadeel van native ontwikkeling zijn de kosten. De applicatie moet twee keer ontwikkeld en onderhouden worden in twee verschillende talen. Hierdoor kost het meer tijd en geld. Daarnaast is het nadeel van twee verschillende talen dat je ook twee teams nodig hebt.

2. Web *(Valdellon, 2020, Hoofdstuk 02. Web Apps)*

Webontwikkeling is goedkoper en sneller doordat er een enkele codebase gebruikt wordt om te ontwikkelen voor alle platformen. Daarnaast is het mogelijk om de webapplicatie beschikbaar te maken op ieder apparaat dat beschikt over een moderne webbrowser. Het nadeel van webontwikkeling is dat er altijd een internetverbinding nodig is om de applicatie te kunnen gebruiken. Daarnaast heeft een webapplicatie weinig tot geen toegang tot sensoren en API's van het apparaat zelf.

3. Hybrid *(Valdellon, 2020, Hoofdstuk 03. Hybrid Apps)*

Is een combinatie van ontwikkeltechnieken dat ervoor zorgt dat er met een enkele codebase ontwikkeld kan worden voor meerdere platformen. Dit vermindert de ontwikkeltijd en -kosten. Daarnaast is het resultaat een applicatie die op het apparaat geïnstalleerd kan worden net als een native applicatie en die toegang heeft tot sensoren en opslag van het apparaat. Een nadeel is dat de applicaties vaak trager zijn dan native ontwikkelde applicaties.

Het HTC is gespecialiseerd in het ontwikkelen van hybride applicaties. Omdat ze geen verschillende teams nodig hebben voor de verschillende platformen is het mogelijk om met een relatief kleine groep ontwikkelaars, grote projecten aan te kunnen nemen.

3.1 Xamarin

Voor het ontwikkelen van deze applicaties maakt het team gebruik van een framework genaamd Xamarin. Dit is een framework dat gebouwd is in de taal C# en is gebaseerd op het .NET framework. .NET is al een bewezen framework dat gebruikt wordt om web, mobiel, desktop, games en IoT mee te ontwikkelen *(Microsoft, 2022, "Cross Platform" sectie)*. Door de actieve ontwikkeling van Microsoft en de grote community van ontwikkelaars erachter was dit een goede start voor de Xamarin-ontwikkelaars.

Het bedrijf achter Xamarin is in 2011 opgericht door de ontwikkelaars achter een eerder succesvol framework genaamd Mono. Mono was ook een framework voor het ontwikkelen van mobiele applicaties maar de ontwikkelaars wilden met Xamarin een betere service aanbieden door het commerciëler aan te pakken. Door het achter een betaalde licentie te zetten, was het mogelijk om een team van ontwikkelaars fulltime aan het framework te laten werken *(Editor, 2020, Hoofdstuk What Is Xamarin?)*.

Het framework bestaat uit drie producten voor ontwikkeling van mobiele applicaties:

1. Xamarin Android
    a. Een C# framework waarmee Android applicaties ontwikkeld kunnen worden.
2. Xamarin iOS
    a. Een C# framework waarmee iOS applicaties ontwikkeld kunnen worden.
3. Xamarin.Forms

a. Cross platform user interface framework waarmee applicaties ontwikkeld kunnen worden voor iOS-, Android- en Windows-applicaties onder één codebase.

Doordat Xamarin geprogrammeerd is in C#, wat een veel gebruikte programmeertaal is *(Stackoverflow, 2021, Hoofdstuk Most Popular Technologies)*, konden programmeurs die hiermee bekend zijn ook applicaties ontwikkelen voor mobiele apparaten. Het voordeel van het gebruik van Xamarin is dat dezelfde structuur en programmeertaal gebruikt kan worden voor de frontend (Mobiele applicatie) en de backend.

De taal C# en het framework .NET dat op C# gebouwd is, is ontwikkeld door Microsoft. Doordat Xamarin hierop gebouwd is en Xamarin zoveel tractie heeft gekregen, heeft Microsoft in 2016 het bedrijf Xamarin overgenomen *(Editor, 2020, Hoofdstuk What Is Xamarin?)*. Met de overname kwam ook een nieuwe koers van het framework . Microsoft maakte het project open-source en gratis te gebruiken. Door deze koerswijziging werd Xamarin ook toegankelijk voor kleinere bedrijven en meer ontwikkelaars.

Dit zorgde voor een toename van het gebruik van Xamarin. Zo werd er in 2017 door 4.5% van ontwikkelaars actief op StackOverflow aangegeven Xamarin te gebruiken waarna een jaar later dit 7.4% was. *(Stackoverflow, 2018, 2019)*

3.2 Kans

Na de overname van Xamarin door Microsoft hebben beide frameworks altijd naast elkaar geleefd. Hierdoor is een volledige integratie van .NET nooit gelukt.

Microsoft heeft in 2020 aangekondigd bezig te zijn met een nieuw framework voor het ontwikkelen van cross platform applicaties in C#. Het nieuwe framework zal volgens Microsoft vanaf de grond af aan opnieuw opgebouwd worden met performance in gedachten. Daarnaast zal het framework volledig geïntegreerd zijn met het .NET framework. Met deze integratie wil Microsoft dichter bij hun ideale wereld komen van “Build Once run everywhere.”. De naam van het nieuwe framework is .NET Multi-platform App UI, in het kort .NET MAUI *(Microsoft, 2022b, Hoofdstuk What is .NET MAUI?)*.

Microsoft heeft dus bij het ontwikkelen van .NET MAUI gefocust op performance en uitbreidbaarheid. Daarnaast hebben ze de grootste pijnpunten van Xamarin op willen lossen. Een van deze pijnpunten volgens Xamarin ontwikkelaars is het missen van .NET hot reload *(Microsoft, 2022b)*. Dit maakt het mogelijk om aanpassingen in je logica direct te laten reflecteren in je draaiende applicatie. Dit kan in potentie de ontwikkeltijd versnellen *(Lyalin & Microsoft, 2021)*.

.NET MAUI heeft dus potentie om sneller applicaties te ontwikkelen met betere performance. Als dit klopt, is het mogelijk voor het HTC om de ontwikkelkosten voor een applicatie te verlagen. Deze reductie kunnen ze gebruiken om:

- De reductie te laten reflecteren naar de klant waardoor ze een betere concurrentiepositie krijgen, of
- Hogere marges te hanteren, waardoor er meer geld vrijkomt voor R&D (Research and Development)

Daarnaast kunnen bestaande applicaties omgebouwd worden naar .NET MAUI wanneer het blijkt dat .NET MAUI een positief performanceverschil geeft. Dit geeft een pro-actieve houding naar de klant.

3.3 Opdracht

.NET MAUI staat nog in zijn kinderschoenen. Zo is de general availability (GA) versie nog niet uitgerold en kan dus alleen gebruikt worden in de preview (beta) versie. Het HTC ziet door de claims van Microsoft veel potentie in het nieuwe framework.

Nu is het een kwestie van tijd dat .NET MAUI Xamarin zal opvolgen, maar vraagt het HTC zich af of het op dit moment al verstandig is om over te stappen naar .NET MAUI. Daarnaast vragen ze zich af of ze bestaande applicaties moeten ombouwen naar .NET MAUI en of ze hier performanceverbeteringen mee kunnen bereiken.

Vandaar de opdracht vanuit het HTC om dat uit te zoeken en advies te geven over het migreren naar .NET MAUI. Hiervoor zijn een aantal aspecten belangrijk. Hierbij gaat het om ontwikkelgemak en performance.

4 Aanpak

Om het HTC van advies te kunnen voorzien, is er een onderzoek opgezet dat zich heeft verdiept in de vraag “Welke performance- en ontwikkelverschillen zijn er voor het HTC wanneer ze bij de eerste release direct migreren van Xamarin naar .NET MAUI voor ontwikkeling van hun mobiele applicaties?”.

Het onderzoek bestaat uit twee onderdelen:

- **Ontwikkelprocessen**

Tijdens de interviews is er gevraagd aan de participanten welke ontwikkelprocessen volgens hen beter of sneller kunnen. Met de ervaring die het team heeft met andere frameworks is er gekeken naar processen van andere frameworks. Het resultaat hiervan is meegenomen tijdens het ontwikkelproces van beide applicaties. Tijdens het ontwikkelen is er gekeken naar verschillen in het proces, dit is gebruikt voor het uiteindelijke adviesrapport.

- **Performance**

Om de performance te kunnen meten tussen de verschillende frameworks is er een applicatie ontworpen en ontwikkeld in beide frameworks. Deze applicaties zijn qua functionaliteiten en code zo goed als hetzelfde. Op deze manier is de enige variabele het framework zodat de performance verschillen van de frameworks getest kunnen worden.

Onder “performance” van een applicatie zijn de volgende punten gemeten:

1. Opslag
    a. Grootte van het project
    b. Grootte van de resultaatapplicaties (.apk / .ipa)
2. Opstartsnelheid applicatie op de volgende manieren
    a. Cold (Opstarten waarbij geen enkel applicatieproces draait)
    b. Hot/Resume (Opstarten waarbij volledige applicatie nog geladen is in het werkgeheugen)
3. Werkgeheugen gebruik op de volgende manieren
    a. Idle (Applicatie is niet volledig afgesloten en draait op de achtergrond)
    b. Active (Applicatie wordt actief gebruikt)

Het onderzoek begon bij het verkennen van het huidige ontwikkelproces en -ervaringen van het team. Dit is gedaan door middel van het afnemen van semigestructureerde interviews bij de vier softwareontwikkelaars van het team die de meeste ervaring hebben met Xamarin. Tijdens deze interviews is er gekeken naar twee onderdelen. Allereerst naar het ontwikkelproces van Xamarin vergeleken met andere frameworks die ze gebruikt hebben. Daarnaast is er gekeken naar de functionaliteiten die in veel applicaties, waaraan de ontwikkelaar heeft meegewerkt, voor komen om zo een beter beeld te krijgen van hun werkwijzen en eerder gemaakte applicaties.

4.1 Ontwerp

Tijdens de ontwerpfase is er allereerst gekeken naar de werkwijze van ontwerpen bij het team, zodat de ontwerpdocumenten bekend aanvoelen bij het team. Hierin gaf het team aan dat ze de ontwerpfase verschillend aanpakken per opdracht. Dit ligt voornamelijk aan de grootte van de opdracht. Het komt vaak neer op een software-architectuurdocument met daarin alle informatie van functionele tot technische eisen van de applicatie.

In overleg met de begeleiders is er gekozen om een eerder gemaakte software-architectuurdocument van het team als voorbeeld te gebruiken. Dit werd alleen gebruikt als richtlijn voor de inhoud van het document. Niels gaf aan om de functionele eisen en use cases te splitsen in aparte documenten zodat het software architectuur

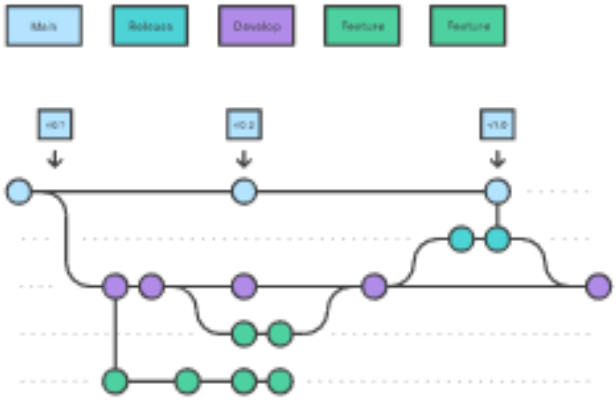
document alleen technische informatie zou bevatten. De bedrijfsbegeleiders waren het eens met deze aanpak van ontwerpen.

Bij de interviews werd er ook doorgevraagd op veel gebruikte functionaliteiten, bibliotheken en apparaat API's. Het resultaat hiervan is uitgewerkt en gecodeerd zodat er een kwantitatief overzicht gemaakt kon worden. Dit overzicht geeft weer welke functionaliteiten het meest voorkomen in de applicaties van de geïnterviewden. Dit heeft geholpen met het opzetten van de functionele eisen voor de applicatie die gebruikt zal worden om de frameworks te benchmarken.

4.2 Opzet ontwikkeling

Het HTC team valt onder het Microsoft cluster waar voornamelijk gebruik gemaakt wordt van producten en services van Microsoft. Voor de ontwikkelomgeving van het team wordt dan ook gebruik gemaakt van Azure DevOps. Dit is een product van Microsoft dat projecten kan helpen met allerlei taken tijdens het ontwikkelen. De services de tijdens het project gebruikt zullen worden zijn *(Microsoft, 2022c, "What is Azure DevOps?" sectie)*:

1. Azure Repos
    a. Git is een versie controle systeem dat ervoor zorgt dat alle veranderingen in de code opgeslagen worden. Hierdoor wordt er een geschiedenis van aanpassingen opgebouwd.
    b. Azure Repos is een service die het beheren van een Git repository eenvoudiger maakt door middel van een grafische interface. Hierin kunnen naast het inzien van wijzigingen ook pull-requests aangemaakt worden. Dit is een verzoek om de aangepaste code door te mogen voeren. Bij het aanmaken van zo'n verzoek kunnen andere ontwikkelaars commentaar leveren op de wijzigingen die voorgesteld zijn. Dit zal tijdens dit project gebruikt worden zodat de teamleden de kwaliteit van de geschreven code kunnen controleren.
    c. Git maakt gebruik van verschillende branches. Dit zijn aftakkingen van de hoofd branch waarin aanpassingen gemaakt kunnen worden. Wanneer de gewenste aanpassingen compleet zijn kan er een pull-request worden gemaakt naar de hoofd branch. Op deze manier kunnen er aan meerdere wijzigingen door meerdere mensen tegelijk gewerkt worden. Het HTC maakt gebruik van de Git Flow.



    Deze flow zal ook gebruikt worden tijden het ontwikkelen van de applicaties.
2. Azure Pipelines
    a. Hiermee kunnen aanpassingen in de code automatisch gebouwd, getest en gedeployed worden. Tijdens het project zal er een pipeline gemaakt worden die automatisch de nieuwe code bouwt en een pakket maakt van de iOS en Android applicatie.
3. Azure Artifacts
    a. Deze service beheert de artifacts die uit de pipeline komen. In dit geval zijn dat de .ipa en .apk pakketten die gebouwd zijn tijdens de pipeline.

4. Azure Boards

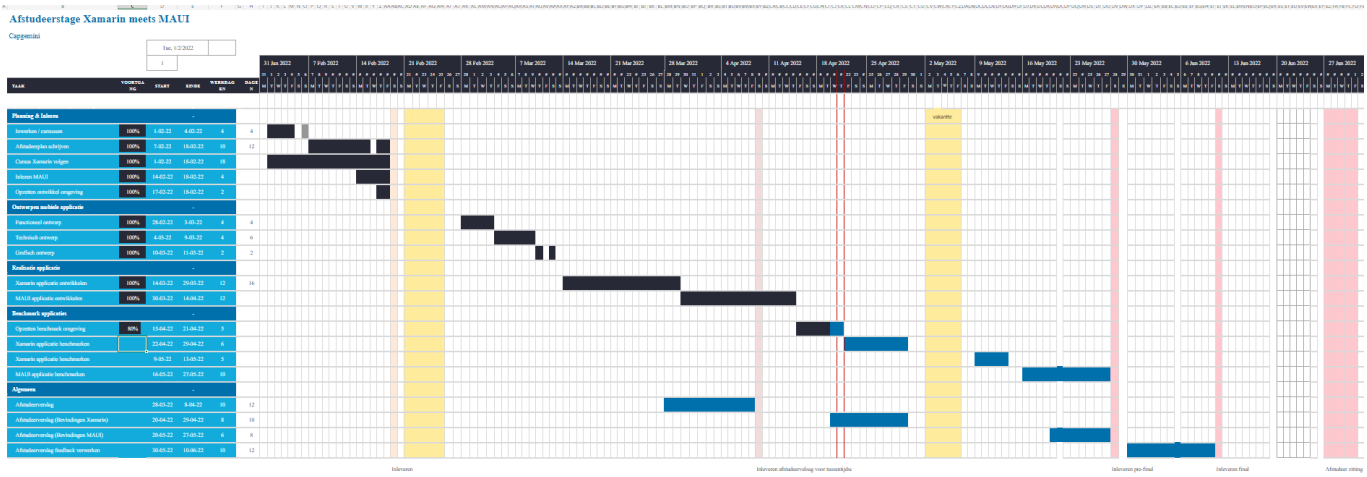
    a. Boards is een service die ontwikkelaars helpt om agile te werken. Agile is een manier van werken die het HTC ook gebruikt in de andere projecten. Dit is een manier van projectmanagement waarmee er gewerkt wordt in kleine iteratieve fases waarin er iedere fase een product wordt opgeleverd. Na iedere fase zal er worden gekeken welke taken er in de volgende fase opgepakt zullen worden. Op deze manier is de ontwikkeling van een applicatie een stuk flexibeler en kunnen aanpassingen eerder worden doorgevoerd.

    b. Hierin zullen de verschillende use cases worden toegevoegd met de onderliggende taken. Door de korte ontwikkeltijd per applicatie zullen de sprints bestaan uit 5 dagen.

5 Proces

5.1 Planning

Om het project tot een succes te leiden is er in de eerste weken een Gantt Chart ontwikkeld met de verschillende deelproducten en de bijhorende verwachte tijdsbesteding. Hierin zijn ook de verschillende deadlines van de Hogeschool Leiden toegevoegd zodat de student en bedrijfsbegeleiders weten wat eraan zit te komen.



Figuur 1 (Planning Gantt Chart)

Daarnaast maakt dit overzicht het ook mogelijk om de voortgang bij te houden, hierdoor wordt een voorsprong of achterstand visueel duidelijk.

Globaal ziet de planning er als volgt uit: Ontwerpfase → Xamarin ontwikkeling → .NET MAUI ontwikkeling → Xamarin benchmarken → .NET MAUI benchmarken.

Hierbij zal er tijdens de ontwikkelfases van de applicaties de Scrum methodiek gebruikt worden. Hierin worden vier achter elkaar volgende sprints opgezet. Per sprint zullen de taken die onder de use cases vallen worden geselecteerd. Hierin lukte het niet om de eerste sprint succesvol af te sluiten. Dit kwam door een inschattingsfout in het werk voor het opzetten van de CI/CD pipeline en Apple development certificaten. Dit was een taak die niet in de sprint was gezet omdat het niet onder een use case valt, maar het moest nog wel gedaan worden. Hierdoor waren al twee dagen van de sprint voorbij zonder enig werk te kunnen verzetten. Bij de Scrum methodiek kan er dan gekozen worden om de onvoltooide taken mee te nemen naar de volgende sprint, dit is toen ook gedaan.

Daarnaast liep Niels tijdens het ontwikkelen van de .NET MAUI applicatie tegen een aantal problemen aan. Deze problemen hadden voornamelijk te maken met de vertraging van Microsoft van het uitgeven van de ‘release candidate’. Dit was gepland in kwartaal vier 2021, alleen dit werd verplaatst naar kwartaal een 2022. Omdat de ontwikkeling van deze applicatie gepland was in april zou dit precies goed uitkomen. Helaas lukte dit niet, Microsoft gaf eind maart aan dat de ‘release candidate’ nog een maand vertraging had. Hierdoor moest er gewerkt worden met een versie van .NET MAUI die nog niet af is waardoor er een aantal problemen zich voordeden. Wat deze zijn en hoe het opgelost is, wordt in hoofdstuk 5.4.1. toegelicht.

Dit heeft uiteindelijk weinig impact gehad op de planning doordat de logica die geschreven was voor Xamarin 1 op 1 overgenomen kon worden naar .Net MAUI. Hierdoor kon een deel van de vertraging worden ingehaald.

5.2 Onderzoek

Om het HTC team een goed advies te kunnen geven, is er een onderzoek opgezet met als hoofdvraag “Welke performance- en ontwikkelverschillen zijn er voor het HTC wanneer ze migreren van Xamarin naar .NET MAUI voor ontwikkeling van hun mobiele applicaties?”

Om een antwoord te kunnen geven op de hoofdvraag zijn de volgende deelvragen opgesteld:

1. Welke technische factoren hebben invloed op de gebruikerservaring van een mobiele applicatie?
2. Hoe kan de performance van een mobiele applicatie gemeten worden?
3. Welke functionaliteiten moet een benchmark applicatie hebben om te lijken op een realistische applicatie van het HTC?
4. Welke ontwikkelprocessen van Xamarin kunnen volgens de teamleden van het HTC sneller, makkelijker of beter?

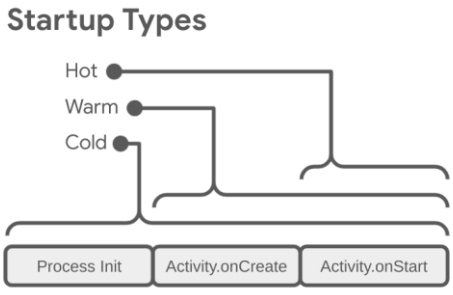
5.2.1 Technische meetpunten voor mobiele applicaties

5.2.1.1 Opstarttijd

Telkens wanneer een gebruiker een applicatie wil gebruiken moet deze eerst opgestart worden. Met de opstarttijd wordt bedoeld de tijd tussen het aanklikken van de applicatie en het tonen van een werkende gebruikersinterface. Wanneer dit te lang duurt kan dit de aandacht van de gebruiker op de proef stellen en bij een lange wachttijd kunnen ze stoppen met hun actie. Daarnaast is het de allereerste ervaring die een gebruiker met een applicatie heeft *(Mo, 2021)*. Naast dat een trage opstarttijd invloed kan hebben op de ervaring van de gebruikers heeft het ook invloed op de waardering van de applicatie in de Android Play store *(Google, 2021, "Recommendations From Google Android Team" sectie)*.

De opstarttijd kan worden gezien vanuit twee verschillende perspectieven *(Google, 2021, "Where to start" sectie)*:

- Time-To-Initial-Display (TTID)
    - De tijd tussen het aanroepen van de applicatie en het eerste beeld dat door de gebruiker gebruikt kan worden (exclusief asynchrone taken).
- Time-To-Full-Display (TTFD)
    - De tijd tussen het aanroepen van de applicatie en het volledig laden van de actieve view. (inclusief het voltooien van alle bijhorende HTTP-verzoeken en asynchrone functies).



De staat waarin de applicatie zich bevindt heeft ook invloed op de opstarttijd. Wanneer de applicatie niet is opgestart zit deze in de COLD staat. Dit betekent dat de applicatie volledig ingeladen moet worden, er bestaat niets meer in het werkgeheugen. Hierdoor kost het opstarten in deze staat vaak het meeste tijd. Wanneer de applicatie eerder al is opgestart zal deze enige tijd op de achtergrond blijven draaien. Wanneer de gebruiker op dat moment terugkeert naar de applicatie start deze vanuit een HOT staat. Wanneer het systeem bepaald om de applicatie vanuit een HOT staat te verwijderen uit het werkgeheugen, beland deze in de WARM staat. Dit resulteert in meer overhead bij het opstarten vergeleken met een HOT start.

5.2.1.2 Werkgeheugen

Het werkgeheugen van een telefoon wordt gebruikt voor het besturingssysteem, het draaien van applicaties, verwerken van de data van de applicaties, cachen en bufferen van data. Wanneer er snel en genoeg werkgeheugen beschikbaar is, kan een applicatie sneller starten en soepeler data verwerken. Wanneer er niet genoeg werkgeheugen beschikbaar is om een applicatie te starten, zal er eerst een applicatie met weinig prioriteit, die op de achtergrond draait, afgesloten worden.

Wanneer de applicatie minder werkgeheugen nodig heeft om dezelfde taken te kunnen vervullen kan dit multitasken bevorderen. Dit komt doordat er op dat moment meer applicaties naast elkaar gedraaid kunnen worden *(Sims, 2022, Hoofdstuk 1)* .

De hoeveelheid werkgeheugen die een applicatie nodig heeft varieert ieder moment. Hierbij zijn twee periodes belangrijk en goed meetbaar. Dit is tijdens het opstarten van de applicatie en wanneer de applicatie stationair meedraait. Dit is het geval wanneer de applicatie opgestart is maar niet als primaire applicatie gebruikt wordt. De applicatie heeft op dit moment nog steeds werkgeheugen nodig om te zorgen dat de staat van de applicatie opgeslagen blijft. Op dit moment kan de applicatie gebruikt worden tijdens het multitasken. Het besturingssysteem zal wanneer nodig de applicatie volledig afsluiten. Dit gebeurt wanneer er werkgeheugen nodig is voor een belangrijker proces of wanneer de applicatie lange tijd niet gebruikt wordt om op deze manier batterij te besparen.

5.2.1.3 Processor

De processor doet al het rekenwerk en dataverwerking voor de verschillende applicaties en het besturingssysteem. Wanneer een applicatie veel van de processor vraagt kan het voorkomen dat de processor langer bezig is met het voltooien van alle taken. Dit komt bij de gebruiker over als een vertraagd systeem. Daarnaast heeft een applicatie die veel vraagt van de processor invloed op de batterijconsumptie van het apparaat *(Chaudhary, 2022)*.

5.2.1.4 Applicatie grootte

De grootte van het resultaat pakket van de applicatie (.ipa voor iOS en .apk voor Android) heeft invloed op het aantal succesvolle installaties en conversieratio. Zo blijkt uit een onderzoek van Zapper, dat een applicatie van 100MB ~30% meer kans heeft om geannuleerd te worden tijdens het downloaden in vergelijking met een applicatie van 10MB. Daarnaast konden ze een correlatie maken tussen de grootte van de applicatie en de conversieratio. Zo zagen ze bij applicaties van onder de 100MB dat er bij iedere 6MB meer, een 1% vermindering is van de conversieratio *(Tolomei, 2022, "Does APK size impact install conversion rate?" sectie)*.

De conversieratio kijkt naar het aantal gebruikers die de applicatie vinden in de app-store en hoeveel mensen de applicatie succesvol voor het eerst gestart hebben.

5.2.2 Functionaliteiten applicatie

De applicatie die gemaakt zal worden moet representatief zijn voor een applicatie die door het HTC team gemaakt zou kunnen zijn. Op deze manier kunnen er functionaliteiten die het team vaak gebruikt vergeleken worden in de benchmarks.

Om deze functionaliteiten die in eerdere applicaties zijn gebruikt op een rij te zetten zijn er semigestructureerde interviews afgenomen met de vier Xamarin ontwikkelaars die op dit moment actief zijn in het HTC. In dit interview zijn de volgende vragen voorgelegd aan de participanten:

1. Welke functionaliteiten zie je veel terugkomen in de applicaties die je ontwikkeld hebt voor het HTC?
    a. Welke apparaat API's worden er vaak gebruikt?
    b. Zijn er bepaalde openbare bibliotheken die veel gebruikt worden?
    c. Maak je wel eens gebruik van hashing of encryptie op het apparaat?

2. Heb je in voorgaande Xamarin applicaties wel eens performance problemen gehad?
    a. Welke functionaliteit was dat?
    b. Hoe is dit opgelost?

De volgende teamleden van het HTC zijn geïnterviewd:

Naam	Functie
Wouter de Bruin	software architect
Bryan Slop	software ontwikkelaar
Iwan Tensen	software ontwikkelaar
Quincy Jacobs	software ontwikkelaar

De resultaten van de interviews zijn getranscribeerd en gecodeerd. Hierdoor is er een overzicht gemaakt van de meest genoemde functionaliteiten, bibliotheken en apparaat API's. Hieruit kwamen een aantal functionaliteiten naar voren die veel herhaald werden. De functionaliteiten die het meeste voorkwamen zijn gebruikt als basis voor de functionele eisen voor de te ontwikkelen applicaties. Ook waren er een aantal bibliotheken die veel zijn gebruikt in eerdere applicaties.

5.2.3 Bibliotheken

Het HTC team gebruikt bij het ontwikkelen een aantal bibliotheken. Sommige van deze bibliotheken komen in ieder project terug. Nu is .NET MAUI de evolutie van Xamarin, maar dit betekent niet dat de bibliotheken direct overgenomen kunnen worden in .NET MAUI. Door de grote aanpassingen in de structuur van .NET MAUI moeten deze bibliotheken aangepast en opnieuw uitgebracht worden voor .NET MAUI *(Matos, 2022, "Refactor or Rewrite or Bifurcate?" sectie).*

Nu kunnen deze bibliotheken een enorme verandering meebrengen in de manier van ontwikkelen. Hierdoor is het belangrijk voor het team of ze de meest gebruikte bibliotheken ook in .NET MAUI kunnen gebruiken. Wanneer dit niet mogelijk is kunnen ze wachten tot dit ondersteund wordt door de bibliotheek maker of overstappen naar een andere vergelijkbare bibliotheek.

Uit de interviews kwam het volgende resultaat op de vraag "zijn er bepaalde openbare bibliotheken die veel gebruikt worden?"

Bibliotheek	Referentie
Newtonsoft JSON	(00:23:44 Quincy Jacobs)
Excalibur	(00:24:17 Quincy Jacobs) (00:16:47 Iwan Tensen)
MvvM cross	(00:17:31 Iwan Tensen) (00:22:10 Quincy Jacobs)
Xamarin.essentials	(00:20:43 Bryan Slop)

	(00:24:02 Quincy Jacobs)

De bovenstaande bibliotheken zijn verder onderzocht op compatibiliteit met .NET MAUI. Dit is belangrijk voor het ontwerp omdat beide applicaties exact hetzelfde gemaakt moeten worden. Alleen wanneer de applicaties qua code en bibliotheken hetzelfde zijn kan de performance van het framework inzichtelijk gemaakt worden.

Bibliotheek	.NET MAUI	Reden / Oplossing
Newtonsoft JSON	Ja	Doordat deze bibliotheek ontwikkeld is op het .NET platform zonder specifieke Xamarin code, is het mogelijk om te gebruiken in alle soorten .NET projecten.
Excalibur	Nee	Dit is een bibliotheek ontwikkeld door een oud teamlid van het HTC. Deze wordt alleen niet meer onderhouden en wordt nu uit projecten gehaald. Hier hoeven we dus niet naar te kijken met .NET MAUI.
MvvM cross	Nee	MvvMCross is de belangrijkste bibliotheek voor het HTC. Dit komt omdat dit, voor het team de basis is van een project. Dit helpt met het MvvM structuur, de vertalingen van teksten, menu's en navigeren. Helaas heeft het team achter MvvM cross laten weten dat de ontwikkeling voor .NET MAUI pas zal beginnen wanneer .NET MAUI officieel uitgerold is en wanneer ze er tijd voor hebben. Dit kan wellicht nog lang duren zo vertelde een ontwikkelaar op de vraag of er een update was "I don't want to promise anything. I think your best bet would be to just go with another framework.."(Raciborski, 2022). Een oplossing hiervoor is, opzoek gaan naar een alternatief die zowel voor Xamarin als .NET MAUI beschikbaar is. Het alternatief waarmee Niels kwam was freshMvvM. Dit is een wat lichtere bibliotheek dat zich vooral focust op de MvvM structuur maar wat minder op de functionaliteiten, zoals vertalingen.
Xamarin.essentials	Ja	Xamarin.essentials is een bibliotheek die het mogelijk maakt om de apparaat API's te gebruiken. Hiermee kunnen sensoren worden uitgelezen, bestanden worden gelezen en geschreven en meer. Dit was in Xamarin een pakket dat toegevoegd moest worden aan het project. In .NET MAUI is dit standaard onderdeel van het project en heet nu MAUI.essentials.

5.2.4 Verbetering ontwikkelprocessen

Het beoordelen van ontwikkelgemak tijdens het ontwikkelen van applicaties is lastig. Dit komt doordat het subjectief is en dus voor iedereen kan verschillen. Om hier wel enigszins advies over te kunnen geven is er gekeken naar bestaande pijnpunten van het ontwikkelen met Xamarin. Deze pijnpunten zijn gehaald uit de codering van de interviews met de teamleden, bij de gestelde vragen:

1. Welke andere frameworks naast Xamarin gebruik je in het HTC?
    a. Welke voor-/nadelen zie je in het ontwikkelproces met deze frameworks vergeleken met Xamarin?
2. Ben je bekend met .NET MAUI, zo ja, welke verbeteringen tijdens het ontwikkelproces kijk je naar uit?

Uit de codering van de interviews blijken twee pijnpunten er bovenuit te stijgen. Dit is het missen van hot-reload tijdens het bewerken van C# bestanden en het toevoegen van afbeeldingen in een applicatie.

5.2.4.1 C# hot-reload

Hot-reload is een techniek die in veel concurrerende frameworks gebruikt wordt en wat de ontwikkelsnelheid kan verbeteren. Dit zie je onder andere terugkomen in React Native *(Khan, 2022)* en Flutter *(Johansen, 2022)*. Hot-reload kan de ontwikkeltijd verkorten door bij aanpassingen in de code het resultaat direct te tonen in een draaiende applicatie. Het verschil in de situatie zal er als volgt uitzien volgens Microsoft *(Lyalin, 2021)*:

Xamarin huidig	Gewenst resultaat
Bij aanpassingen van XAML code (layout / views) worden de wijzigingen direct gereflecteerd in een draaiende applicatie.	Dit werkt op dit moment zoals gewenst.
Bij aanpassingen van C# code (logica) worden de aanpassingen niet direct gereflecteerd. De applicatie dient afgesloten, opnieuw gebouwd en opnieuw gepusht te worden naar het apparaat. De tijd die het kost ligt aan de grootte van de applicatie.	Bij aanpassingen van C# code worden aanpassingen direct gereflecteerd in een draaiende applicatie. Ook de staat van de applicatie zal worden opgeslagen zodat er niet opnieuw ingelogd of genavigeerd hoeft te worden.

Nu heeft Microsoft aangekondigd dat .NET MAUI Hot reload zal gaan ondersteunen. Dit is goed nieuws maar een aantal teamleden vroeg zich af:

- Werkt het altijd of zijn er scenario's waarbij de applicatie toch opnieuw gebouwd moet worden?
- Wordt de staat van de applicatie onthouden wanneer er een aanpassing gepusht wordt?

Tijdens het ontwikkelen van de .NET MAUI applicatie is C# hot-reload en XAML hot-reload continue gebruikt. XAML hot-reload werkte al goed in Xamarin en werkt hetzelfde in .NET MAUI.

De C# hot-reload werkt goed in de Release Candidate (RC) versie van .NET MAUI. Toen er begonnen was met het bouwen van de applicatie was preview 14 de meest recente beschikbare versie. Hierin zaten nog grote fouten, één daarvan was dat het bouwen en hot-reloaden van iOS applicaties niet werkte. Toen is er tijdelijk alleen op Android en Windows UWP ontwikkeld, hierin werkte hot-reload nog naar behoren.

C# hot-reload kan veel tijd besparen, de ervaringen tijdens het ontwikkelen waren heel erg positief. Het grootste voordeel is dat wanneer je een tikfout maakt, aan het debuggen bent of een nieuwe functionaliteit aan het schrijven bent, je niet iedere keer de applicatie hoeft te bouwen, pushen en navigeren naar de juiste view.

Tijdens het ontwikkelen van de Xamarin benchmark applicatie was de bouw en push tijd naar de apparaten als volgt:

iPhone	Android
10.21s + handmatig app starten	13.54s
8.82s + handmatig app starten	11.28s
9.36s + handmatig app starten	11.18s
9.46s	**12s**

Dit is de tijd die er bespaard wordt per wijziging in de C# code. Dit is de minimale tijd die het bespaart per aanpassing en kan tijdens ontwikkeling en onderhoud veel tijd besparen. . Wanneer je aanpassingen doet in een view waar eerst nog naar genavigeerd moet worden kost dit ook nog eens extra tijd. Dit komt omdat je na iedere aanpassing ook weer op de startpagina begint van de applicatie. Dit kost voor de ontwikkelaar veel tijd en kan er bovendien voor zorgen dat de ontwikkelaar uit zijn concentratie wordt gehaald.

5.2.4.2 Toevoegen afbeeldingen

Het toevoegen van afbeeldingen is in Xamarin een grote taak. Dit komt doordat Android en iOS andere type afbeeldingen ondersteunen. Daarnaast heb je ook nog afbeeldingen voor ieder soort scherm, hierbij wordt gekeken naar de pixel ratio van het scherm. Om een afbeelding voor ieder mogelijke scenario van een iOS apparaat te hebben moeten er 47 variaties van de afbeelding worden gemaakt en toegevoegd aan de “Asset catalogus” *(Apple, 2018, "Types Overview" sectie)*.

Nu zal er in de praktijk een aantal van deze variaties weg vallen doordat apparaten zoals Carplay, smartwatches en oudere iPhones niet in de scope van het project vallen. Maar dit kost nog steeds veel tijd volgens de ontwikkelaars van het HTC.

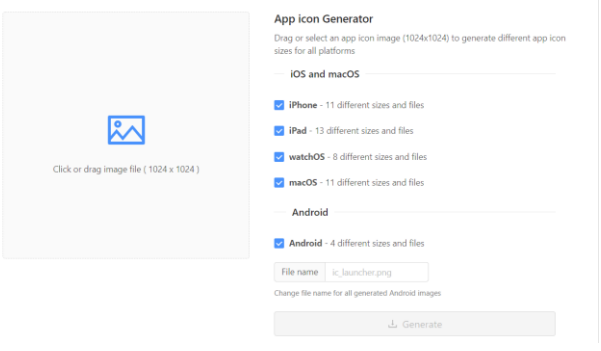
In concurrerende frameworks zijn er makkelijkere manieren om dit te doen. Zo maakt Flutter alleen gebruik van pixel ratio om de afbeeldingen te koppelen. Hiervoor is er slechts een combinatie van vier afbeeldingen die toegevoegd moeten worden. Flutter zal zelf de juiste afbeelding tonen voor de juiste pixel ratio *(Google, Flutter, 2022)*.

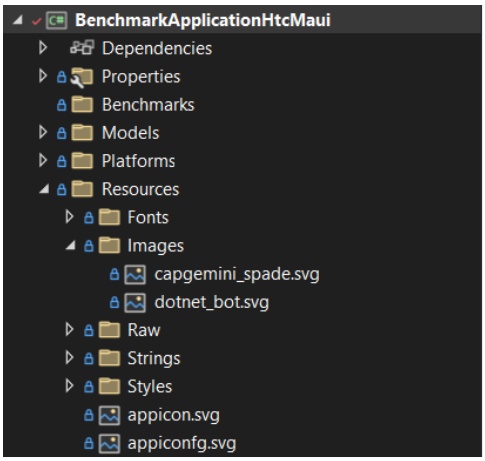
Bij React Native werkt dit op eenzelfde manier alleen moet er zelf logica worden geschreven om de juiste afbeelding te koppelen bij de juiste pixel ratio *(React Native, 2022)*.

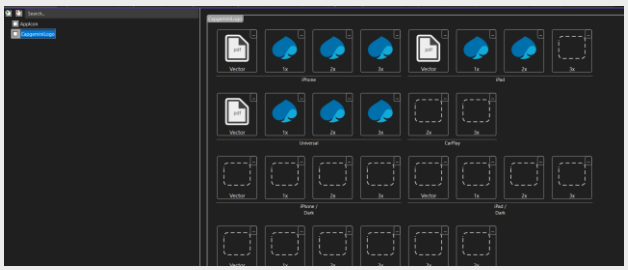
Microsoft heeft bij .NET MAUI een nieuwe manier bedacht van afbeeldingen toevoegen in de applicaties. Dit is door gebruik te maken van Scalable Vector Graphics (SVG). SVG bestanden zijn vector bestanden die veel gebruikt worden voor het ontwerpen van logo’s, icons en illustraties *(Morris, 2020)*. Doordat dit bestandstype gebruik maakt van vectoren kunnen ze onbeperkt worden vergroot zonder kwaliteitsverlies. Dit kenmerk gebruikt .NET MAUI om bij het bouwen van de applicatie de SVG om te zetten naar de verschillende pixel ratio’s. Op deze manier hoeft de ontwikkelaar nog maar één versie van het logo, het icon of de illustratie toe te voegen die geschikt is voor alle varianten van de verschillende apparaten.

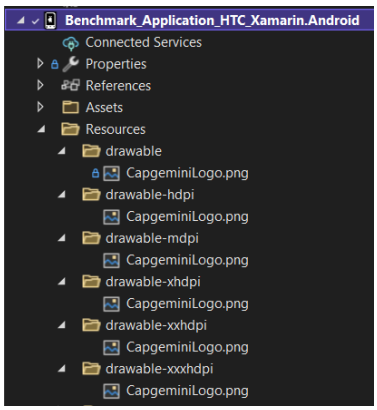
Dit is hoe het zou moeten werken, tijdens het ontwikkelen van de MAUI applicatie bleek de basis van het toevoegen van afbeeldingen goed te werken. Zo kan er een SVG toegevoegd worden en aangeroepen worden via de code.

De verschillen in de processen worden echt duidelijk wanneer deze naast elkaar worden gezet:

#	Xamarin	.NET MAUI
1	Afbeelding in juiste formaten maken. Dit hoeft tegenwoordig niet meer handmatig maar kan gedaan worden door offline of online tools zoals appicon.co: 	Afbeelding in SVG formaat toevoegen in het project:

2	iOS project: Asset aanmaken in asset catalog van iOS. iOS maakt gebruik van een asset catalog, dit is een verzameling formaten van dezelfde afbeelding. Wanneer de asset aangeroepen wordt zal iOS zelf kijken welke versie van de afbeelding gebruikt zal worden.
	 
		Afbeelding aanroepen in de view naar keuze:
		`<Image`
		`Source="capgemini_spade.png"`
		`WidthRequest="180"`
		`HeightRequest="180"`
		`HorizontalOptions="Center"/>`
		Het formaat dat je aanroept is png doordat .NET MAUI de SVG gebruikt om de juiste formaten png te maken. De ingeladen afbeelding is dus niet de SVG, dit is omdat de verschillende platformen verschillende formaten hanteren.

3	Android project: Maak een afbeelding set met de verschillende resoluties: 	

4	Afbeelding aanroepen in de view naar keuze:
	`<Image`
	`HeightRequest="180"`
	`Source="CapgeminiLogo"`
	`WidthRequest="180" />`	

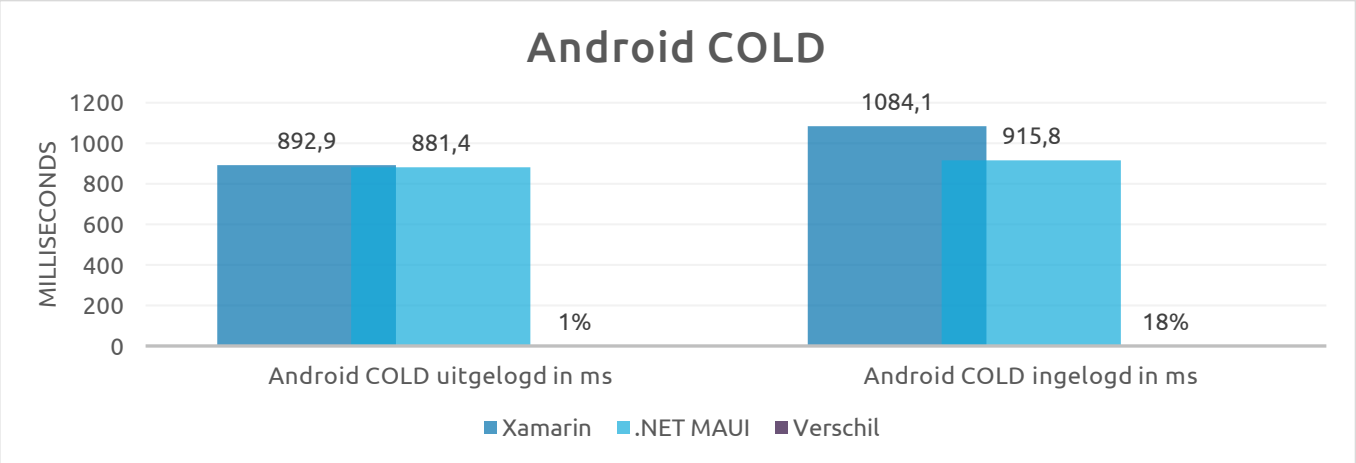
5.2.5 Performance

Om de performanceverschillen tussen Xamarin en .NET MAUI inzichtelijk te maken is er een applicatie ontworpen en ontwikkeld in Xamarin en .NET MAUI. De functionele eisen van deze applicatie komen uit de interviews met de participanten. Om te zorgen dat de applicaties de performance van het framework reflecteren is het belangrijk dat de broncode overeenkomt. Hiervoor is de broncode van de projecten met elkaar vergeleken. Hieruit kwam dat de broncode voor 96% overeenkomt. Het enige verschil hierin is dat de bibliotheek “Essentials” bij Xamarin ingeladen moest worden door middel van een import en bij MAUI standaard beschikbaar is. Hierdoor is het enige verschil tussen beide applicaties het framework waarin ze zijn ontwikkeld.

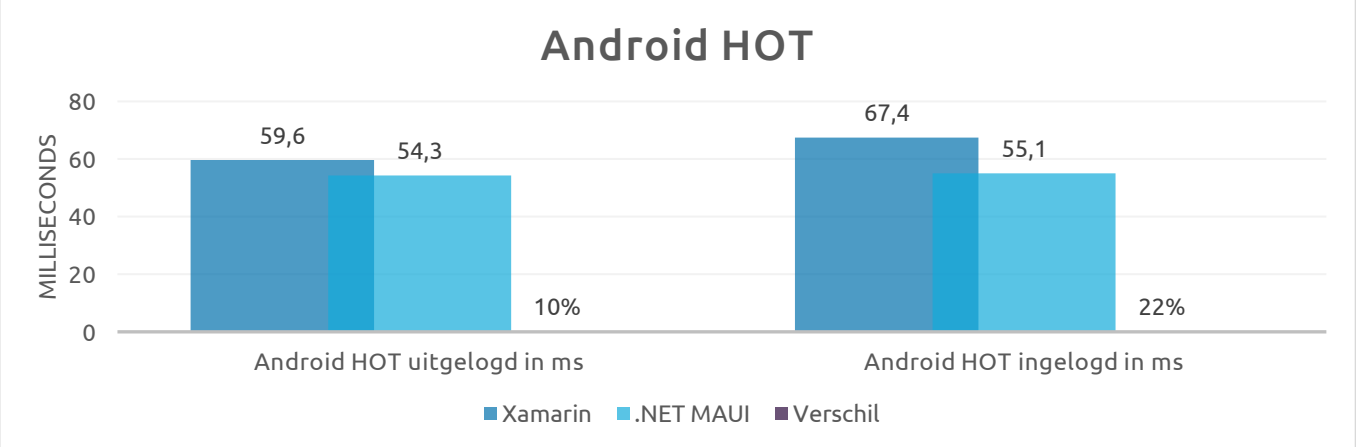
5.2.5.1 Android

5.2.5.1.1 Opstartsnelheid

De applicatie is op twee manieren en in twee staten getest. Dit is via een HOT en COLD start, dat houdt in of de applicatie al in de achtergrond draaide (HOT) of niet (COLD). Daarnaast is de test uitgevoerd wanneer de gebruiker ingelogd is en wanneer niet. Het verschil hierin is dat wanneer een gebruiker ingelogd is de token uit de secure storage van het apparaat gehaald wordt en de gebruiker doorgestuurd wordt naar een nieuwe view.



Met de Android applicatie is er vooral bij de ingelogde gebruiker een verbetering te zien. Hier start de applicatie gebouwd in .NET MAUI 18% sneller op vergeleken met de applicatie gebouwd in Xamarin.

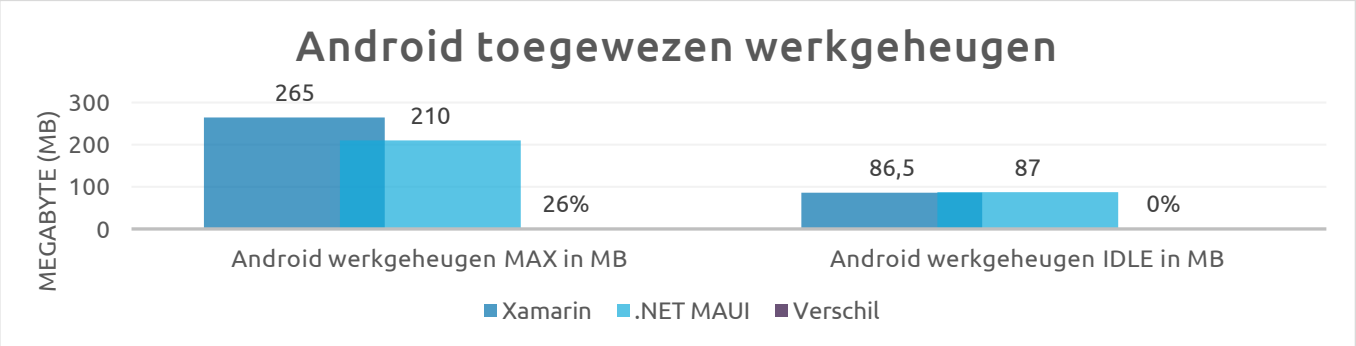


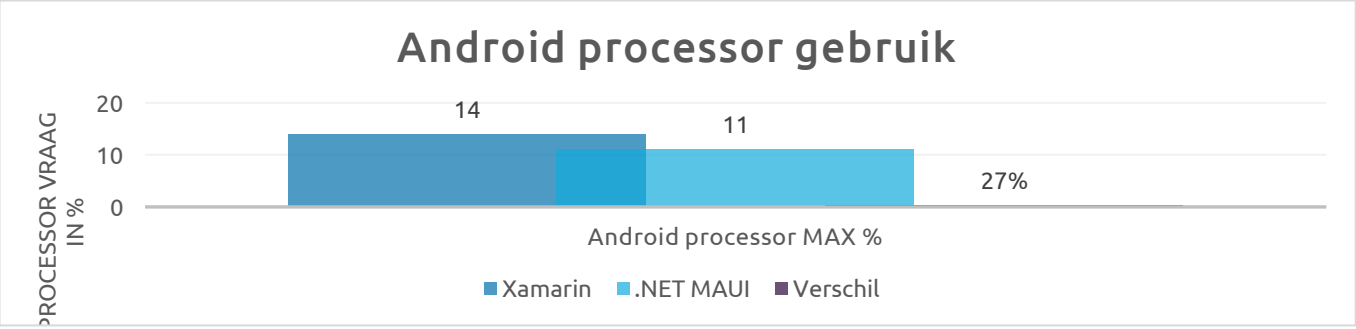
Wanneer de applicatie nog in de achtergrond draait en naar de voorgrond geroepen wordt is er vastgesteld dat zowel bij ingelogde staat als uitgelogde staat de applicatie sneller getoond wordt. Daarnaast waren de resultaten van Xamarin in een grotere spreiding dan .NET MAUI, wat laat zien dat de .NET MAUI applicatie stabielere opstartsnelheden geeft vanuit een HOT staat.

5.2.5.1.2 Werkgeheugen en processorgebruik

Het toegewezen werkgeheugen en de processorkracht namen ook af bij de applicatie die ontwikkeld is in .NET MAUI.

Voor het werkgeheugen gebruik is de applicatie in twee staten getest. Dit is tijdens het actief gebruiken van de applicatie, hierbij wordt gekeken naar de maximale vraag naar werkgeheugen. Ook is er gekeken naar de werkgeheugen gebruik terwijl de applicatie op de achtergrond draait, deze staat noemen we "idle".

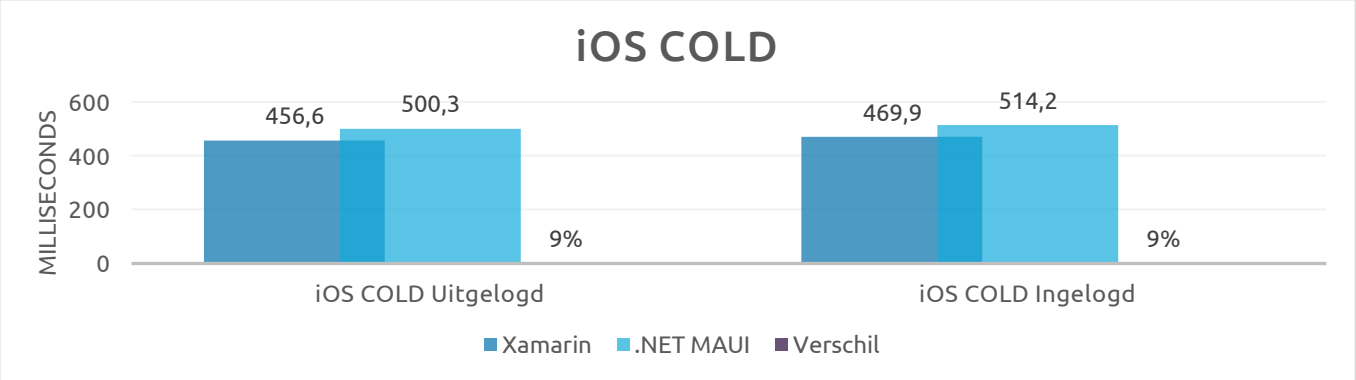




5.2.5.2 iOS

5.2.5.2.1 Opstart snelheid

Bij de performance benchmark van iOS is er alleen gekeken naar het opstarten van een ingelogde en uitgelogde gebruiker met de applicatie in de COLD staat. De reden is hiervoor is dat de gebruikte tool, Xcode Trace, geen metingen voor HOT state kan doen.



De resultaten van de iOS applicatie waren negatief. Hieruit bleek dat de opstartsnelheid in beide gevallen langzamer is dan de applicatie die ontwikkeld is in Xamarin. Wanneer er ingezoomd wordt op de opstartfases

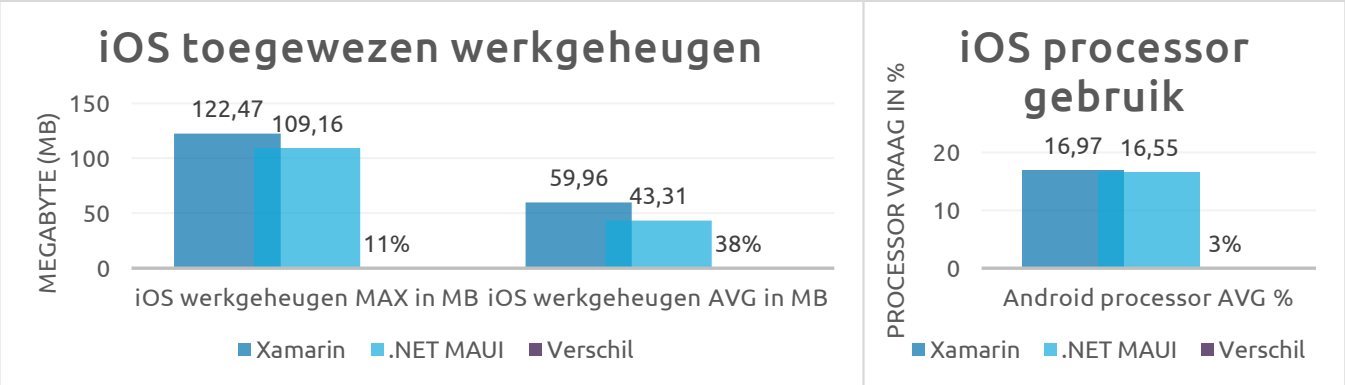
van de applicatie is er te zien dat de applicatie gebouwd in .NET MAUI maar op één punt veel trager is (WillFinishLaunchingWithOptions):

Activiteit	Xamarin seconden	.NET MAUI seconden
Launching – UIKit Initialization	0.0240342s	0.02698242s
Launching – UIKit Scene Creation	0.0003936s	0.00042555s
Launching - WillFinishLaunchingWithOptions	0.0000899s	0.02119327s
Launching - DidFinishLaunchingWithOptions	0.2511294s	0.24910752s
Launching – Initial Frame Rendering	0.0233976s	0.03332989s

Een reden van dit verschil kan zijn dat het .NET MAUI team zich nog niet gefocust heeft op de performance van iOS. Zo is er een project op Github van een Microsoft ontwikkelaar die de performance van Xamarin en .NET MAUI applicaties vergelijkt, maar alleen voor Android *(Peppers, 2022)*. Daarnaast zegt de product manager van .NET MAUI het volgende: *"You have told us how critical it is for your applications to start as quickly as possible, especially on Android." (Ortinau, 2022, "Optimized for Speed" sectie)* en niets over iOS, wat kan betekenen dat ze hier minder moeite in hebben gestopt.

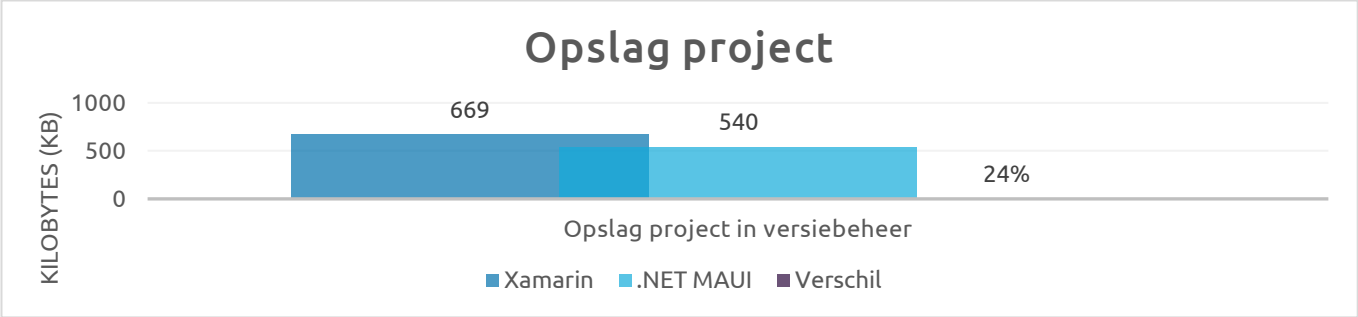
5.2.5.2.2 Werkgeheugen en processorgebruik

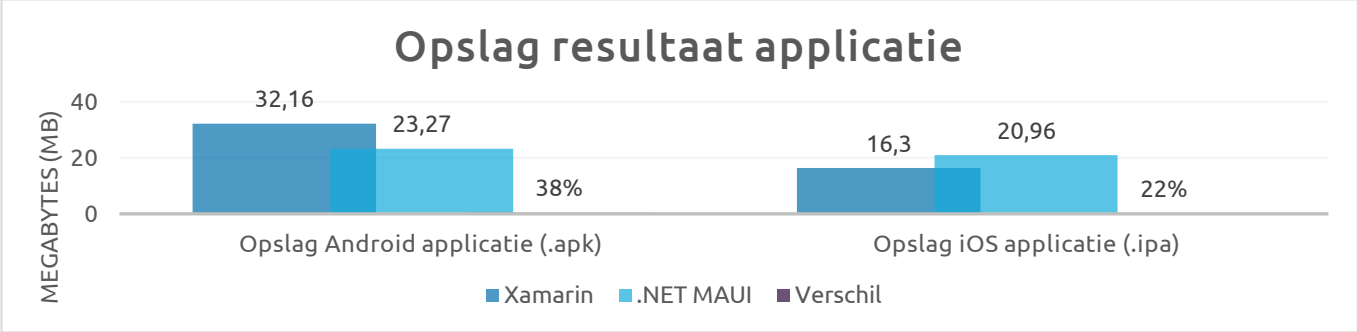
Het werkgeheugen en processorgebruik van de applicatie gebouwd in .NET MAUI is wel positief veranderd. Uit de benchmark is gebleken dat het werkgeheugengebruik met 38% is afgenomen.



5.2.5.3 Opslag

De opslag van de frameworks is vanuit twee perspectieven gemeten. Vanuit het ontwikkelaars perspectief, door te kijken naar de omvang van het project in versiebeheer. En vanuit het gebruikersperspectief dat kijkt naar de omvang van de resultaat applicatie (.ipa of .apk).





Het verschil in de opslag van het project is te verklaren door verschillende verbeterpunten. Allereerst bestaat een .NET MAUI project maar uit één visual studio oplossing. In vergelijking bestaat een Xamarin project uit drie projecten (iOS, Android en een gedeelde oplossing). Daarnaast maakt het gebruik van een vector bestand het project ook een stuk kleine doordat je niet meer alle verschillende variaties hoeft op te slaan. Zo nemen afbeeldingen in het Xamarin project 321kb in beslag waarbij het .NET MAUI project maar 780 bytes in beslag neemt.

In de grootte van de applicatie is er bij Android een verkleining te zien. Dit is door een Microsoft ontwikkelaar te verklaren door verschillende verbeteringen in .NET MAUI *(Peppers, 2022, "App Size Improvements" sectie)*. Zo is er een .NET API verwijderd uit het framework wat resulteert in een reductie van ongeveer 1MB. Ook hebben ze ongebruikte HTTP implementaties weg kunnen halen wat een reductie geeft in applicaties die HTTP verzoeken verwerken. De iOS applicatie is groter geworden bij .NET MAUI. Hiervoor is nog geen verklaring gevonden.

5.2.6 Conclusie

Met dit onderzoek is er gezocht naar een antwoord op de vraag: *"Welke performance- en ontwikkelverschillen zijn er voor het HTC wanneer ze bij de eerste release direct migreren van Xamarin naar .NET MAUI voor ontwikkeling van hun mobiele applicaties?"*. Om hier antwoord op te kunnen geven is er een kwalitatief onderzoek uitgevoerd om de ontwikkelverschillen inzichtelijk te maken en een kwantitatief onderzoek gedaan om de verschillen in performance in kaart te brengen.

Qua proces waren er volgens de participanten van het interview twee onderdelen waar volgens hen verbetering in moet zijn. Dat is het missen van hot-reload voor de C# code en de hoeveelheid werk om een afbeelding toe te voegen voor iOS en Android. Microsoft heeft aangegeven met .NET MAUI ook C# hot-reload te ondersteunen. Hierdoor is het niet meer nodig om een applicatie opnieuw te bouwen en deployen bij een codewijziging. Dit kostte met de benchmarkapplicatie al 10 tot 12 seconden. In plaats daarvan worden de wijzigingen direct getoond op een draaiende applicatie. Naast de 10 tot 12 seconden voordeel van het bouwen hoeft er ook niet meer ingelogd of genavigeerd te worden naar de pagina waar je bezig bent. Tijdens de hot-reload wordt namelijk de staat van de applicatie onthouden. Dit bespaart veel tijd voor de ontwikkelaar.

Daarnaast was het volgens de participanten veel werk om afbeeldingen toe te voegen aan een Xamarin applicatie. Dit waren ook de bevindingen tijdens het ontwikkelen van de benchmarkapplicatie. Om een afbeelding of icoon voor ieder iOS en Android apparaat optimaal te tonen moest er van deze afbeelding 47 variaties worden gemaakt en worden toegevoegd aan het project. In .NET MAUI is dit een stuk eenvoudiger doordat er maar één SVG bestand hoeft te worden toegevoegd aan het project. .NET MAUI zorgt er bij het

bouwen voor dat de variaties van de afbeelding gemaakt worden. Dit bespaart tijd voor de ontwikkelaar en grafisch vormgever.

De performance van .NET MAUI op Android is volgens alle gemeten data verbeterd. Zo is de opstartsnelheid bij een COLD start tot en met 18% versneld en bij een HOT start tot wel 22%. Daarnaast zat er minder spreiding in de opstarttijden van de .NET MAUI applicatie, wat een consistentere opstarttijd aangeeft. Daarnaast vraagt de applicatie gemaakt in .NET MAUI ook minder maximale processorcapaciteit en werkgeheugen van het apparaat. Dit is met 27% voor de processor en 26% voor het werkgeheugen afgenomen.

Bij de iOS applicatie is het een ander verhaal. Hier is de opstarttijd met 9% vertraagd. Ook is de applicatie 22% groter. Dit is waarschijnlijk te verklaren door de focus van het .NET MAUI team. In meerdere artikelen is alleen te lezen over de performance verbeteringen van het Android platform. Ook heeft een Microsoft ontwikkelaar een opstartsnelheid script geschreven om .NET MAUI en Xamarin te vergelijken alleen voor Android. Dit laat zien dat de focus van het team waarschijnlijk op dit moment op Android ligt. Als dit het probleem is, is het een kwestie van tijd voordat de performance gelijkgetrokken wordt en hopelijk verbeterd wordt ten opzichte van Xamarin.

5.3 Ontwerp

Voor de benchmark is er een applicatie ontwikkeld die de functionaliteiten bevat die de applicaties van het HTC ook vaak gebruiken. Deze applicatie is zowel in Xamarin als in .NET MAUI ontwikkeld. Tijdens de ontwerpfase is het belangrijk om dit altijd in het achterhoofd te houden omdat de functionaliteiten en bibliotheken die gebruikt worden voor beide platforms beschikbaar moeten zijn.

Tijdens de ontwerpfase is er in iteraties gewerkt waarin er iedere oplevering samen met de begeleiders gekeken werd naar het ontwerp. Deze iteraties gingen per onderwerp waarin minimaal één feedbackmoment zat:

1. Vision: Doelstelling, Belanghebbenden, Functionele requirements, Niet-functionele requirements, alternatieven.
2. UseCase model: Use cases, UseCase diagram, use case specifications.
3. Testplan: Test scope, strategie, productrisico analyse.
4. Software architectuur document (SAD): Overzicht RUP-views, Architecturale eisen, verantwoordelijkheid deelsystemen, communicatielagen, use case-realisation, structuur project, te gebruiken bibliotheken.

5.3.1 Functionele eisen

De functionele eisen zijn opgenomen in het Vision-document. Deze eisen zijn tot stand gekomen door de requirements die uit de interviews kwamen. De functionaliteiten die in de interviews naar voren kwamen zijn gesorteerd op het aantal keer dat ze voorkwamen. Hier is een selectie van gemaakt en voorgelegd aan de begeleiders Wouter en Bryan. In dit overleg kwam een functionaliteit naar voren die maar één keer in de codering van de interviews zat maar toch vaak gebruikt wordt. Dit was het gebruik van een master-detail overzicht waarbij een lijst getoond wordt waarbij je door kan klikken om de details te zien van het geselecteerde item. Deze functionaliteit wilden de begeleiders graag terugzien in de functionele eisen.

Daarnaast waren ze het eens met de prioritering van de functionele eisen. De prioritering is gedaan door een cijfer toe te wijzen van 1 tot en met 3, waarbij 1 minder prioriteit heeft dan 3.

1	2	3

Code	Functionele requirements	Prioriteit
FR01	Een gebruiker kan zich authentiseren met een geldig account.	3
FR02	Een ingelogde gebruiker kan een overzicht laden met steden.	3
FR03	Een ingelogde gebruiker kan een detailpagina bezoeken bij het klikken op een stad.	3
FR04	Een ingelogde gebruiker kan met de camera van het apparaat een foto maken en toevoegen aan de stad.	3
FR05	Een ingelogde gebruiker kan een afbeelding verwijderen van een stad.	2
FR06	Een ingelogde gebruiker kan een side menu bedienen om te kunnen navigeren.	2
FR07	Een ingelogde gebruiker kan een stad toevoegen.	2
FR08	Een ingelogde gebruiker kan een stad verwijderen	2
FR09	Het systeem moet een push notificatie sturen wanneer dit door het team is aangegeven.	1

Om de functionele requirements heen is een scenario voor de applicatie bedacht. Zo is de applicatie een reisboek waarin de gebruiker steden die hij/zij bezoekt kan aanmaken met een beschrijving over de stad. Hieraan kunnen ze een foto toevoegen via de camera op de mobiele telefoon, om zo het reis moment vast te kunnen leggen en nooit meer te vergeten.

5.3.2 Niet-functionele eisen

Niet-functionele eisen gaan in op de eisen van de applicatie en manier van ontwikkelen die geen functionaliteiten zijn. De belangrijkste eis is dat de applicatie voor Xamarin en .NET MAUI hetzelfde ontwikkeld wordt met zo min mogelijk onderlinge verschillen. Op deze manier is de enige variabele bij de benchmarks het gebruikte framework. De niet-functionele eisen die zijn opgesteld voor het project zijn:

Code	Niet-functionele requirements
NFR01	De applicatie moet uit te breiden zijn zodat andere functionaliteiten later toegevoegd kunnen worden.
NFR02	De applicatie dient ontwikkeld te worden met de platformen Xamarin.Forms & .NET MAUI.
NFR03	De applicaties moeten in beide frameworks hetzelfde worden gemaakt. o.a. gebruik maken van dezelfde bibliotheken. Wanneer dit niet mogelijk is kan er gekeken worden naar een alternatief: 1. Gebruik maken van een alternatieve bibliotheek (moet duidelijk aangegeven worden want kan invloed hebben op performance) 1. Functionaliteit die de bibliotheek gebruikt zelf schrijven 1. Functionaliteit waarvoor de bibliotheek gebruikt is weglaten uit de benchmark applicatie
NFR04	De applicatie, broncode en gebruikte tools en scripts dienen bij oplevering beschikbaar worden gesteld aan het HTC team.
NFR05	De applicatie moet voorzien zijn van een splashscreen zodat de gebruiker feedback krijgt bij het opstarten van de applicatie.

5.3.3 Acceptatie criteria

Voor het ontwikkelen van de benchmarkapplicatie zijn er minder acceptatiecriteria dan er normaalgesproken aan een productie applicatie zou zitten. Dit heeft te maken dat veel acceptatie criteria te maken hebben met gebruikerservaring, dit is iets wat minder effect heeft op een applicatie dat alleen gebruikt wordt in een ontwikkel omgeving.

Wel zijn er een aantal acceptatie criteria opgesteld en zijn verwerkt in het software architectuur document. Deze criteria zijn vooral bedoeld om de kwaliteit van de benchmarks te waarborgen. De belangrijkste criteria zijn:

5.3.3.1 Codebase overeenkomst

Omschrijving	Om een eerlijke benchmark te kunnen doen tussen de twee frameworks is het belangrijk dat de twee resultaat applicaties zo goed als hetzelfde zijn. Omdat beide applicaties geschreven zijn in C# is het mogelijk om de codebases met elkaar te vergelijken. Hoe meer deze overeenkomen hoe beter dit de performance van de gebruikte framework reflecteert.
Doel, streefwaarde en toleranties	Het doel is om de broncode van de twee applicaties meer dan 85% overeen te laten komen. Dit geeft genoeg ruimte om eventuele aanpassingen in .NET MAUI te tolereren.
Meetmethode	Er zal handmatig worden gecontroleerd of de zelf geschreven C# en XAML code overeenkomen.
Planning	De meting zal eenmalig plaatsvinden aan het eind van het ontwikkeltraject.
Corrigerende acties	Wanneer de overeenkomst minder dan 85% is, zal de broncode hersteld moeten worden tot dat deze streefwaarde bereikt is.

5.3.3.2 Project instellingen overeenkomst

Omschrijving	Om een eerlijke benchmark te kunnen doen tussen de twee frameworks is het belangrijk dat de twee resultaat applicaties zo goed als hetzelfde zijn. Naast de broncode kunnen project instellingen veel impact hebben op de performance van een applicatie. Daarom is het belangrijk dat deze instellingen gelijk staan voor beide resultaatapplicaties.
Doel, streefwaarde en toleranties	Het doel is om de project instellingen gelijk te krijgen. Waarbij de streefwaarde 95% overeenkomst is. Hier is een kleine speling gehouden zodat eventuele aanpassingen of andere benamingen bij .NET MAUI getolereerd worden.
Meetmethode	Er zal handmatig worden gecontroleerd d.m.v. Visual Studio 2022.
Planning	De meting zal eenmalig plaatsvinden aan het eind van het ontwikkeltraject.
Corrigerende acties	Wanneer de overeenkomst minder dan 95% is, zal de instellingen hersteld moeten worden tot dat deze streefwaarde bereikt is.

5.3.4 Use cases

De functionele eisen zijn gedetailleerder opgeschreven als use cases. Hierin zijn de use cases geprioriteerd en is er een gewicht aangegeven. De prioriteit van een use case is toegewezen volgens de MoSCoW methode. Hierbij worden de usecases verdeeld in vier groepen, die aangeven hoe belangrijk deze zijn voor de applicatie *(Mulder, 2022, "Wat is de MoSCoW methode?" sectie).*

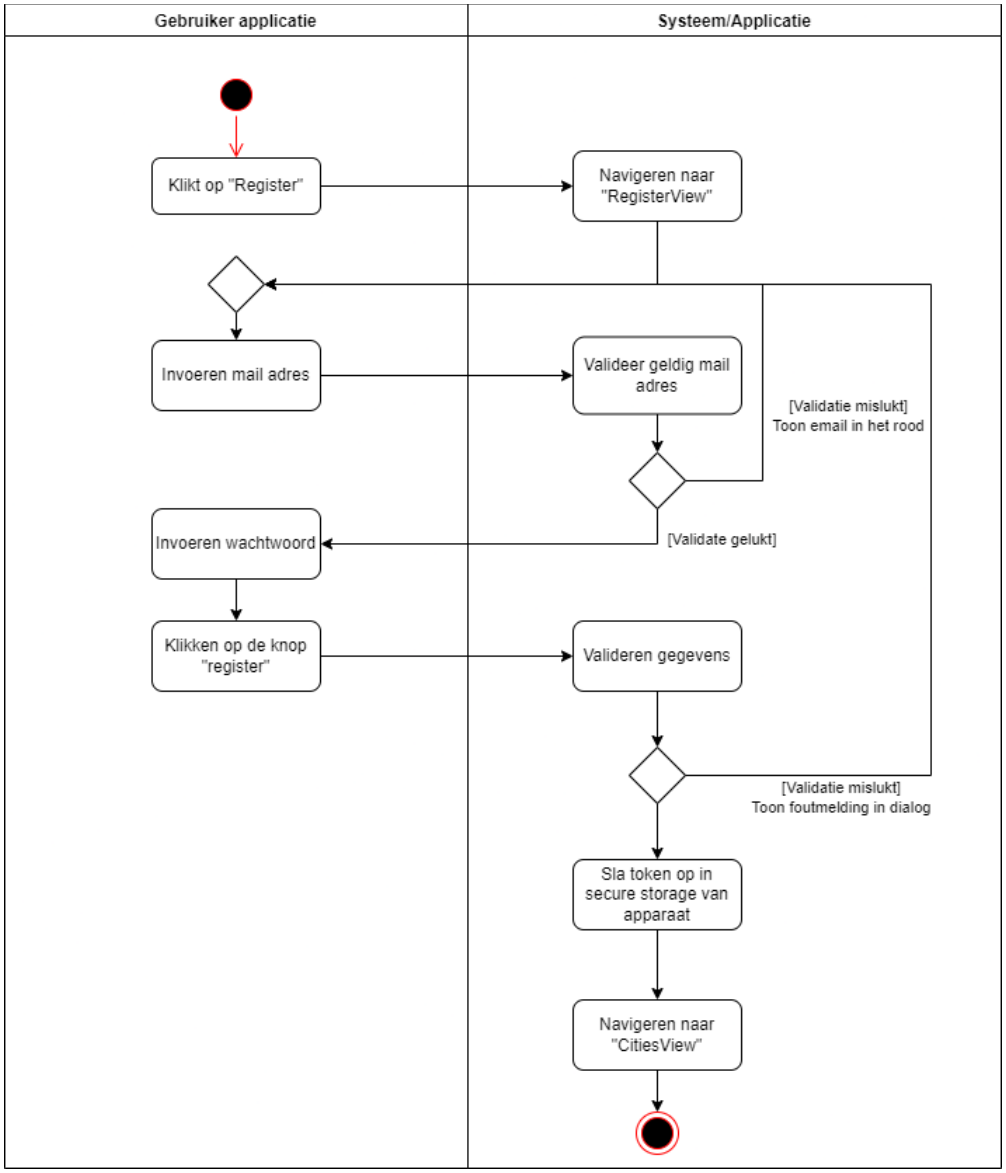
- *W = Won't haves*
- *C = Could haves*
- *S = Should haves*
- *M = Must haves*

Het gewicht geeft aan hoeveel werk het is. Dit is relatief gezien van de andere use cases en kan een gewicht hebben tussen de 1 en 3:

Code	Use case name	Omschrijving	G	P
UC01	Inloggen	Een gebruiker vult een geldige gebruikersnaam en wachtwoord in en drukt op de knop "inloggen" om vervolgens ingelogd te zijn.	3	M
UC02	Registreren	Een gebruiker moet een account aan kunnen maken door middel van een geldig e-mailadres en wachtwoord.	2	M
UC03	Uitloggen	Een ingelogde gebruiker moet kunnen uitloggen.	1	M
UC04	Overzicht laden met steden	Een ingelogde gebruiker kan een overzicht laden met verschillende steden.	3	M
UC05	Details tonen stad	Een ingelogde gebruiker kan vanuit het overzicht een stad aanklikken om hierdoor details te zien van de geselecteerde stad.	2	M
UC06	Foto toevoegen aan stad	Een ingelogde gebruiker kan vanuit een detailpagina een foto toevoegen aan de stad door middel van de camera van het apparaat.	3	M
UC07	Foto verwijderen	Een ingelogde gebruiker kan een eerder toegevoegde afbeelding bij een stad verwijderen.	2	C
UC08	Navigeren tussen views	Een ingelogde gebruiker kan zich door de applicatie navigeren door middel van een side menu.	2	M
UC09	Stad toevoegen	Een ingelogde gebruiker kan een stad toevoegen.	2	C
UC10	Stad verwijderen	Een ingelogde gebruiker kan een stad verwijderen	1	C
UC11	Push notificatie	Gebruiker ontvangt een push notificatie wanneer het ontwikkelteam dit aangeeft.	3	S

Het aanmaken van de use cases is in overleg geweest met de bedrijfsbegeleiders. Hierbij gaven ze aan dat het gebruik van push notificaties een interessante functionaliteit is die veel terugkomt. Hierbij gaven ze wel aan dat dit technisch uitdagend is en veel tijd kan kosten om het te implementeren bij iOS en Android. Hierdoor is ervoor gekozen om de prioriteit op “Should have” te zetten. Op deze manier kon er gekeken of er nog ruimte was in de sprint om deze functionaliteit toe te voegen. Zo niet dan kon er worden gekeken om een nieuwe sprint te starten of om de functionaliteit voor nu achterwege te laten. Uiteindelijk is dit de enige functionaliteit geweest die niet gerealiseerd is. Dit had te maken met de ondersteuning van push notificaties op .NET MAUI. Dit was tijdens de ontwikkeling van de applicatie niet goed ondersteund waardoor het veel tijd zou kosten om het te implementeren voor iOS en Android.

Iedere use case is uitgewerkt in een use case specificatie. Hierin wordt de use case volledig functioneel beschreven. Dit bestaat uit twee onderdelen. Allereerst een activiteitendiagram van de use case. Dit geeft een schematische weergave van de stappen die gezet worden voor die specifieke use case. Hierin wordt er een duidelijk beeld geschetst welke acties de gebruiker onderneemt, wat het systeem doet, welke foutmeldingen er kunnen ontstaan en hoe het einde van een succesvolle flow eruit ziet. Een use case voor het registreren van een gebruiker ziet er als volgt uit:



Figuur 2 (Usecase diagram)

Het use case activiteiten diagram laat zoals hierboven verschillende scenario's zien die doorlopen kunnen worden. Deze scenario's en de stappen die erbij horen zijn uitgeschreven. Hierbij wordt er ook gekeken naar een toepasselijke foutmelding. Het testplan zal deze scenario's gebruiken om volledige dekking te kunnen bieden op de functionaliteiten.

5.3.5 Testplan

Om te zorgen dat de applicaties voldoen aan de functionele eisen en om kwaliteit te bieden is er een testplan opgezet. Het testplan beschrijft alle scenario's van de use cases. Hierbij worden dus alle happy en alternatieve flows doorgelopen. Naast de functionele onderdelen van de applicatie wordt er ook gekeken naar de beveiliging, gebruiksvriendelijkheid en performance. Aangezien de applicatie alleen gebruikt wordt voor de benchmark en geen toekomstvisie heeft om uitgebreid te worden zal er op dit moment nog niet gekeken worden naar test automatisering.

In het testplan is er een teststrategie opgezet met een risicoanalyse. De risicoanalyse geeft per onderdeel van de applicatie aan hoe groot het risico is van het **niet** functioneren van dat onderdeel van de applicatie. Het risico wordt gemeten met een formule: $R = F \cdot S$
Waarbij R = Risico, F = Foutkans en S = mogelijke schade. Hieruit is een overzicht gekomen die de risicoklasse weergeeft van ieder onderdeel van de applicatie met de bijbehorende onderbouwing (Zie Bijlage 1: Productrisico-analyse).

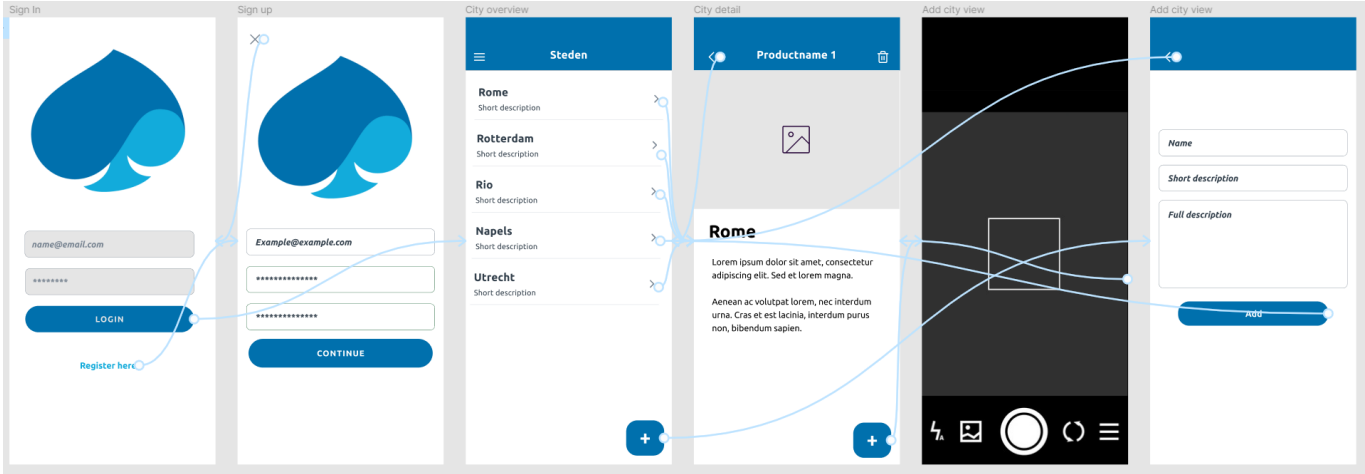
De onderdelen uit het productrisico-analyse zijn getest door middel van drie soorten testen:

Testsoort	Beschrijving	Doel	Technieken
Bouwtest	Bouwtest is een testsoort die uitgevoerd wordt tijdens het ontwikkelen. Hierbij wordt er gekeken of een functionaliteit doet wat er van verwacht wordt.	Verifiëren dat de geschreven code functioneert zoals verwacht en dat aangebrachte wijzigingen geen neveneffecten hebben (regressie).	Functioneel testen
Systeemtest	Systeemtest is een testsoort die wordt uitgevoerd na iedere (sprint)oplevering. Hierbij wordt er aan de hand van de usecase specificaties gekeken of alle flows uit te voeren zijn zoals verwacht.	Verifiëren dat de gerealiseerde software voldoet aan de overeengekomen use cases. Dit maakt het ook inzichtelijk of en wat er iedere sprint op te leveren is.	Black-box testing
Performancetest	Performancetest is een testsoort die gebruikt wordt om inzicht te krijgen in de snelheid en bruikbaarheid van een applicatie.	Het doel bij de benchmark applicatie is om inzicht te krijgen in de performanceverschillen tussen een applicatie die gebouwd is met Xamarin en een applicatie die gebouwd is met .NET MAUI.	Performance benchmarks

Echter zullen de verschillende scenario's handmatig getest worden na iedere sprint. In totaal zijn er vier sprints gestart voor het ontwikkelen van beide applicaties. De resultaten van deze testen zijn opgeleverd in een testrapport. Hierin wordt er per sprint gekeken naar de geslaagde en gefaalde testen.

5.3.6 Prototype

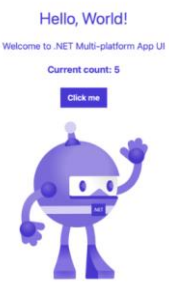
Om het grafisch ontwerp te testen en voor te kunnen leggen aan het team zijn er aantal flows verwerkt in een prototype. Het prototype is gebaseerd op het grafisch ontwerp dat gemaakt is in Figma. Met behulp van deze tool zijn de flows aangemaakt waarna deze flows afgespeeld kunnen worden. Op deze manier was het mogelijk om een gevoel te krijgen bij het grafisch ontwerp.



Figuur 3 (Prototype flows *(Van der Knaap, 2022)*)

5.3.7 Proof of concept

Tijdens het onderzoek wordt er gekeken naar het gebruik van een van de nieuwste technieken, namelijk .NET MAUI. Om hiervoor een valide benchmark op te kunnen zetten is het belangrijk om in ieder geval te weten of .NET MAUI al werkt voor de platformen waarop getest gaat worden.



Om dit te controleren is de template applicatie van .NET MAUI gebruikt om te bouwen naar iOS en Android. Dit is gedaan in de laatst beschikbare preview versie van Visual Studio destijds, dat was 17.2 preview 1. Hierna is de basisfunctionaliteit getest op beide apparaten.

5.3.8 Architectuur

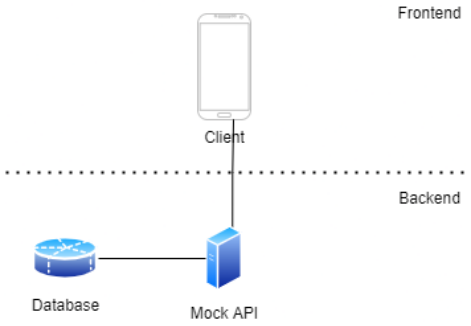
De technische en architecturale keuzes voor de applicaties zijn vastgelegd in het Software Architectuur Document (SAD). Dit is aangepakt door gebruik te maken van het 4+1 view model. Dit model zorgt ervoor dat de verschillende perspectieven van de verschillende belanghebbenden worden uitgelicht.

5.3.8.1 Deelsystemen

De benchmark applicatie zal bestaan uit verschillende deelsystemen. Hierbij is ervoor gekozen om een mock-API als backend API te gebruiken. Dit is een server die een API-server imiteert, waarbij verschillende entiteiten aangemaakt kunnen worden. Voor iedere entiteit worden er automatisch endpoints gemaakt waarmee gecommuniceerd kan worden. Dit zorgt voor een volledige API met alle CRUD (Create, Read, Update, Delete) acties. De keuze voor het gebruik van een mock API heeft te maken met tijd en de benodigdheden voor het project. Zo is het doel van de opdracht om de performance verschillen te meten van tussen Xamarin en .NET MAUI. Dit focust zich op de frontend en is onafhankelijk van de backend. Hierdoor maakt het niet uit of de backend zelf geschreven of een mock-API is.

De applicaties moeten kunnen communiceren met een API maar deze hoeft geen moeilijke logica uitoefenen of afbeeldingen op te slaan. Dit valt namelijk buiten de scope van het benchmarken van de applicatie.

De deelsystemen die gebruikt worden zien er schematisch als volgt uit:



Deelsysteem	Functie
Client	Dit is het mobiele apparaat van de gebruiker. Dit kan zowel een Android als een iOS apparaat zijn. Dit systeem zorgt voor het grafische gedeelte waarmee de gebruiker kan interacteren met de data en logica. Het systeem communiceert via HTTP-verzoeken met de backend API.
API	Dit systeem staat open voor HTTP-verzoeken op bepaalde paden. Het zal het verzoek verwerken, data valideren, data vertalen of herstructureren om vervolgens de data naar de database te sturen. Ook kan dit systeem data ophalen vanuit de database en terugsturen naar de client.
Database	Dit systeem zorgt voor dataopslag en is enkel bereikbaar via de API.

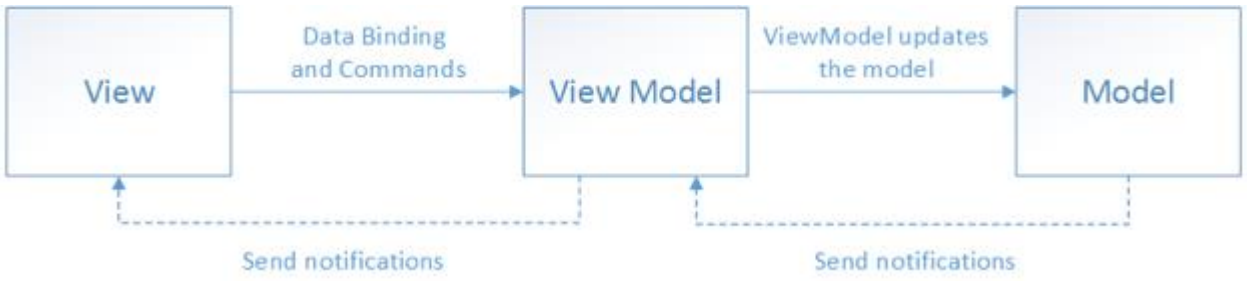
5.3.8.2 Structuur project

Bij het ontwikkelen van Xamarin applicaties gebruikt het HTC team het software ontwikkelpatroon Model-View-ViewModel (MVVM). Dit is een ontwikkelpatroon dat de code structureert. Het scheidt de logica code van de gebruikersinterface. Deze scheiding zet zich door zowel in de folderstructuur als de namespaces waardoor het een overzichtelijk project wordt.

De gescheiden lagen zijn verantwoordelijk voor het volgende:

- Model
    - Het model is een representatie van het domein model. Hierin worden de kenmerken van de verschillende modellen gedefinieerd. Daarnaast wordt hierin ook business logica en validatie logica aangegeven.
- View
    - De view is een verzameling met visuele elementen waarmee de gebruiker interacteert. In Xamarin & .NET MAUI wordt hiervoor gebruik gemaakt van XAML. Dit is een markup language zoals HTML.
- ViewModel
    - Het ViewModel zit tussen de View en het Model in. Hierin worden de kenmerken en methodes voor de componenten van de View ondergebracht. Daarnaast wordt er gebruik gemaakt van binding om een verbinding te leggen met de verschillende componenten in de View.

Schematisch ziet dit er als volgt uit:



Figuur 4 (Schematische weergave MVVM)

5.4 Ontwikkeling Benchmark applicatie

Om voor het onderzoek de performance verschillen in kaart te brengen is er eenzelfde applicatie ontwikkeld in zowel Xamarin als .NET MAUI. De ontwikkeling ging door middel van sprints waarin iedere 5 dagen taken van een use case worden geselecteerd. Deze zijn in die sprint ontwikkeld en getest.

In het Microsoft Azure DevOps portaal is per feature/ bug of cleanup een pull request aangemaakt naar de development branch. Wanneer er een pull request aangemaakt was, ging er automatisch een pipeline draaien. De pipeline zal de laatste aanpassingen ophalen en het project proberen te bouwen. Wanneer dit succesvol is zal er een .apk en .ipa opgeslagen worden. Dit zijn de pakketten voor iOS en Android applicaties.

Om een pull request te kunnen accepteren zal het moeten voldoen aan de volgende vastgelegde eisen:

- iOS & Android pipeline moeten succesvol worden doorlopen
- Minimaal één akkoord van een andere ontwikkelaar

5.4.1 .NET MAUI

Tijdens het maken van de planning zou de planning van .NET MAUI precies goed uitkomen. Zo zou de 'release candidate' in het eerste kwartaal uitkomen zodat er in april begonnen kon worden met de ontwikkeling. Helaas was er een vertraging bij Microsoft waardoor de uitrol in april pas zou komen.

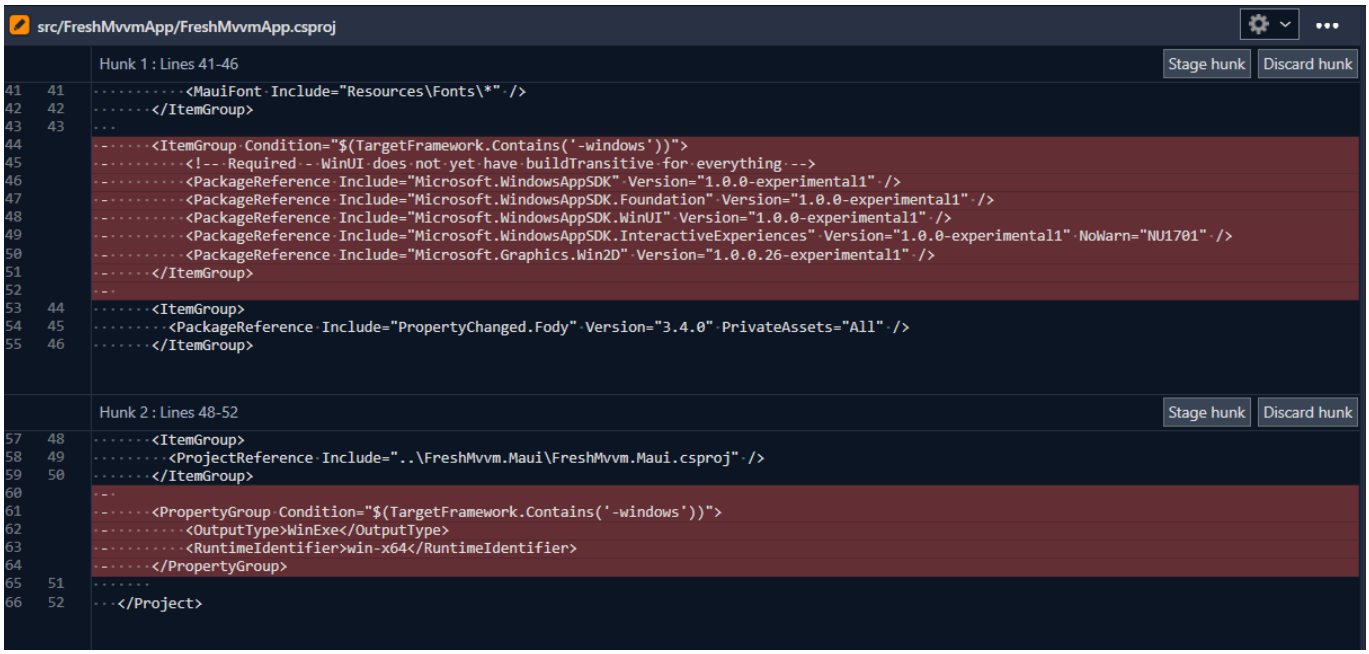
Om hier niet op te wachten is de laatste preview versie gebruikt om alvast te beginnen met de ontwikkeling. Dit had nog wel een aantal problemen waar soms een creatieve oplossing voor bedacht moest worden.

Ten eerste zat er in de laatste preview versie (preview 14) een probleem met de Apple developer certificaten. Hierdoor was het niet meer mogelijk om te ontwikkelen en debuggen op iOS. Dit was niet zelf op te lossen waardoor het bericht vanuit Microsoft was "*We are aware this also happened for Xamarin and working to get it fixed. It should be fixed in the upcoming Visual Studio versions asap.*" *(Versluis, 2022)*. Mijn tijdelijke oplossing was om de applicatie te ontwikkelen en testen op Android. Hierdoor was het mogelijk om de volledige functionaliteiten te ontwikkelen en testen. Toen de update uit kwam kon ik deze functionaliteiten ook testen op iOS.

De update die uitkwam was ook direct de 'release candidate'. Dit betekent dat er wellicht nog kleine bugs in zitten maar dat de structuur niet meer aangepast zal worden en dat alle functionaliteiten gemaakt zijn. Daarnaast is het vanaf deze release mogelijk om op het Microsoft ontwikkel forum vragen te stellen over het framework.

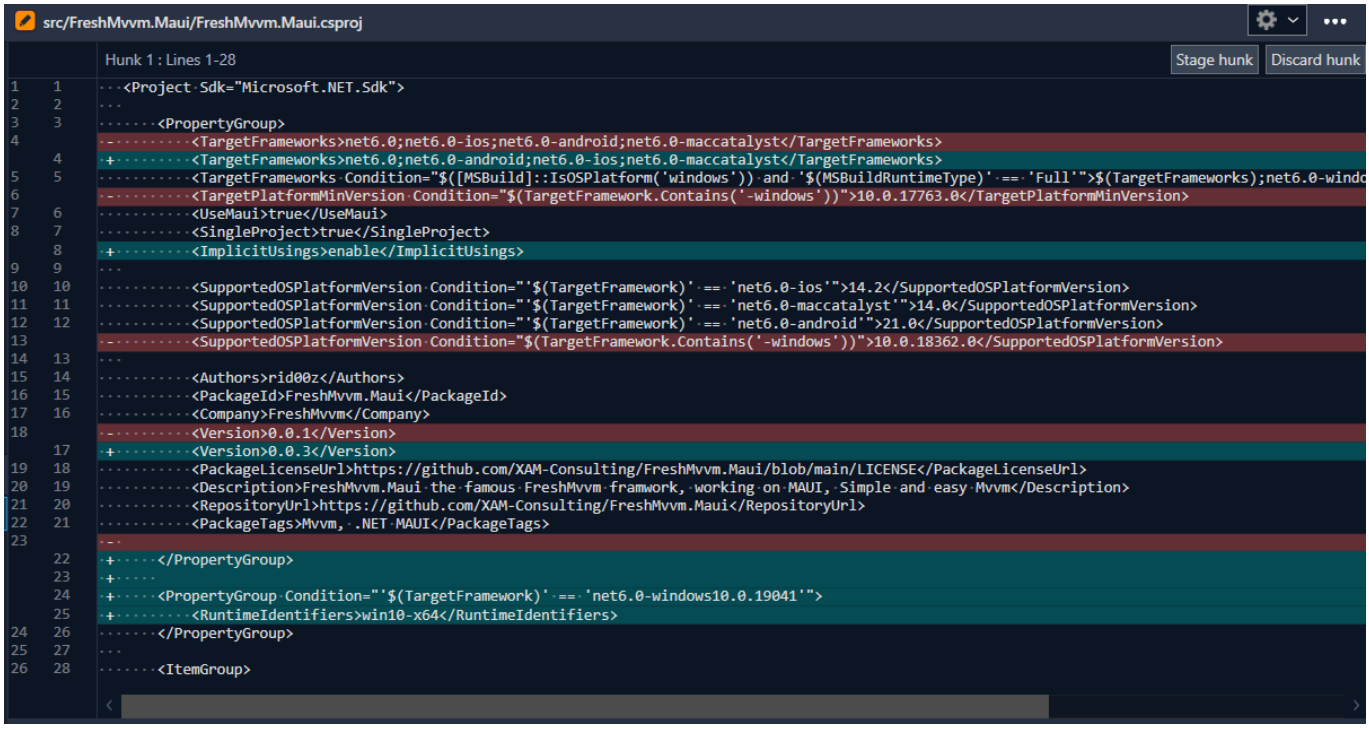
Deze update kwam wel met een volgend probleem. Dit kwam door de bibliotheek FreshMVVM. Alle bibliotheken moesten namelijk opnieuw gebouwd worden voor de nieuwste .NET MAUI versie. Helaas is het bedrijf dat FreshMVVM onderhoudt (nog) niet actief bezig met de .NET MAUI bibliotheek. Hierdoor kwam er maar geen update in de package manager van .NET (NuGet).

Gelukkig was het open-source waardoor de update ook zelf uit te voeren was. Dit is gedaan door de broncode op te halen en te openen in de laatste versie van Visual Studio. Om het project te kunnen bouwen en verpakken naar een NuGet pakket, moest het project aangepast worden zodat het compatibel was met het nieuwe .NET MAUI structuur:



Figuur 5 (Aanpassing code freshMVVM)

Hierboven wordt oude code verwijderd die na de laatste release in .NET MAUI verwerkt zit.



Figuur 6 (Aanpassing code freshMVVM)

Hierboven wordt het versienummer aangepast en wordt aangegeven welke runtime identifier gebruikt moet worden voor Windows.

Na het inpakken van het project in een NuGet pakket kon het pakket ingeladen worden in het benchmark applicatie project. Door deze oplossing heeft Niels ook andere ontwikkelaar kunnen helpen op Github. *(Kelly, 2022)*

5.4.2 Benchmark script

Om de applicaties te kunnen testen op opstartsnelheid, werkgeheugen en processorgebruik kan er gebruik gemaakt worden van een aantal debug tools voor het desbetreffende platform. Bij iOS is dit Xcode tracers en bij Android is dit Android Debug Bridge (ADB).

Hierbij zijn veel handmatige acties nodig en het is dus lastig om precies te reproduceren. Hierom is ervoor gekozen om voor deze acties een script te ontwikkelen. Deze scripts zijn ontwikkeld met behulp van Microsoft Powershell. De keuze voor deze scripttaal is omdat het draait op zowel Windows als MacOS. Omdat de ontwikkelaars van het HTC gebruik maken van beide systemen is handig dat het werkt op deze twee platformen. Alternatieve cross-platform script talen waarnaar gekeken is, zijn Bash en Python. Hieruit kwamen de volgende voor- en nadelen:

Functie	Powershell	Bash	Python
Voor geïnstalleerd Windows	Ja	Nee	Nee
Voor geïnstalleerd macOS	Nee	Ja	Nee
Named command line arguments	Ja	Nee	Niet out of the box, maar mogelijk met een bibliotheek.
Command line arguments validatie	Ja, d.m.v. toevoegen van “flags” aan een argument.	Nee, kan wel zelf geschreven worden.	Niet out of the box, maar mogelijk met een bibliotheek.
XML parser	Ja	Nee	Niet out of the box, maar mogelijk met een bibliotheek.

Hieruit is gebleken dat powershell tussen deze drie scripttalen de meeste functionaliteiten bezit die nodig zijn voor het benchmarken van Android en iOS applicaties. Een van de belangrijkste functionaliteiten is de ingebouwde XML parser die nodig is om voor een iOS benchmark de resultaten uit te lezen.

5.4.2.1 Android

Voor Android wordt er gebruik gemaakt van ADB, dit is een tool waarmee er gecommuniceerd kan worden met het aangesloten apparaat op debug niveau. Hierdoor kunnen ook alle systeem logs uitgelezen worden.

Het script dat geschreven is vraagt de gebruiker om de huidige activity en applicatie id mee te geven. Deze zijn nodig om de applicatie via ADB te kunnen starten. Met deze informatie zal het script de applicatie een aangegeven aantal keer opstarten via een COLD start en een aangegeven aantal keer opstarten via een HOT start.

Via adb logcat worden de logs opgehaald waarin gezocht wordt naar het opstarten van de applicatie. Android geeft hierin namelijk exact mee wat de opstarttijd is van de applicatie. Uit deze tekst wordt het aantal milliseconden gehaald en opgeslagen.

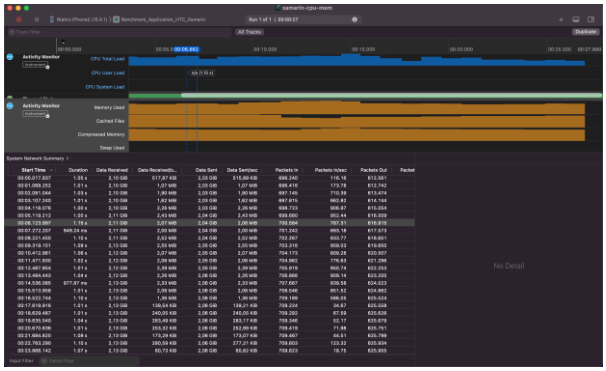
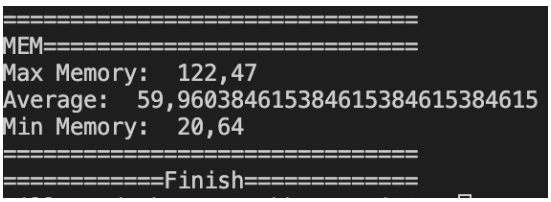
Nadat alle iteraties voltooid zijn zal het script de gemiddelde waardes tonen en de bijbehorende error marge, wat de spreiding aangeeft.

5.4.2.2 iOS

Bij iOS wordt er gebruik gemaakt van Xcode trace. Dit is beschikbaar als een grafische tool (onder Xcode > Instruments) maar bestaat ook als CLI variant. Hierdoor is het mogelijk om de acties die je handmatig doet te automatiseren. Xcode werkt op een andere manier dan ADB. Xcode maakt een opname van alles wat er op de telefoon gebeurt en slaat dit op als een .trace bestand. Dit is een formaat dat Xcode gebruikt om opnames op te slaan en uit te lezen.

Dit maakt het lastig om te automatiseren omdat bijvoorbeeld de opstarttijd van de applicatie ergens verstopt zit in dat bestand. Daarnaast is de CLI tool van Xcode niet tot nauwelijks gedocumenteerd. Wel staat er in de CLI tool dat er een export functie is naar XML.

Deze exportfunctie is gebruikt om de opnames naar XML te exporteren waarna het gerealiseerde script het XML bestand doorloopt en daar de interessante evenementen uithaalt met de bijbehorende tijdstempel. Deze methode is gebruikt om de opstarttijd te automatiseren zodat er eenvoudig een gemiddelde berekend kan worden over meerdere iteraties. Maar ook is dit gebruikt om het werkgeheugen en processorgebruik uit een .trace bestand te halen. Hierbij worden de maximale, minimale en gemiddelde waardes berekend. Dit resulteert in het volgende:

Oude manier, geen mogelijkheid tot berekenen van gemiddeldes of zien wat de maximale of minimale waardes zijn.	**Direct inzichtelijk wat de verschillende waardes zijn, zelfs over meerdere opnames.**
	
Figuur 7 (Xcode instruments)	**Figuur 8 (iOS .trace export script)**

5.5 Adviseren

Het doel van de opdracht was uiteindelijk om een advies te geven aan het HTC team over het gebruik en migreren van Xamarin naar .NET MAUI. Het onderzoek wordt hier gebruikt als onderbouwing van het advies. Het advies is verwerkt in een adviesrapport waarin een breed advies wordt gegeven en het toont de kosten en baten van het migreren naar .NET MAUI.

Daarnaast is er nog advies op maat voor de volgende scenario's, deze zijn elk onderbouwd:

1. Migratie van Bestaande applicaties waarbij de EOL (End Of Life) binnen 2 jaar zal zijn
2. Migratie van Bestaande applicaties die actief worden bijgehouden
3. Nieuwe applicaties

Het advies bevat de volgende kosten en baten:

Kosten	Baten
Mindere performance iOS	Verbeterde performance Android
Geen ondersteuning bestaande Xamarin bibliotheken	Versnelde ontwikkel-/debug tijd door .NET HOT-reload
	Versnelde ontwikkel-/ontwerp tijd door gebruik vectoren
	Langer ondersteund door Microsoft

Hieruit is het volgende advies gepresenteerd:

Nieuwe applicaties	Bestaande applicaties met onderhoud	Bestaande applicatie met end of life (of in het zicht van)
.NET MAUI	Adviseren tot ombouwen naar .NET MAUI	Niet ombouwen

Het advies is eerst gepresenteerd aan het gehele team via een PowerPoint presentatie. Hierin is het team meegenomen in de opdracht, onderzoeksopzet, resultaten en vervolgens het advies. Na de presentatie is er ruimte geweest voor vragen en discussie over het onderzoek en het gegeven advies. Dit is vervolgens verwerkt in een verslag wat opgeleverd is aan de bedrijfsbegeleiders van Niels.

Het HTC team heeft ook aangegeven dat het een interessant onderzoek is en dit verwerkt zou kunnen worden in een blog en motiveerde mij om de resultaten te delen met de wereld. Vooral omdat de manier van iOS benchmarken nieuw is waarmee veel ontwikkelaars geholpen kunnen worden. Niels is op dit moment bezig met het schrijven van de blog en zal dit publiceren op een eigen blog of op de website Medium.com. Daarnaast is Niels bezig met en schrijven van een artikel over de resultaten dat gepubliceerd zal worden in de nieuwsbrief van de Microsoft community binnen Capgemini.

5.6 Reflectie

** Voor de reflectie zal er gebruik gemaakt worden van de ‘ik’ vorm

5.6.1 Planning

In het begin van mijn stage heb ik een planning gemaakt met een Gantt chart. Hierin heb ik de verschillende deelproducten die ik in het begin dacht te maken toegevoegd met de verwachte tijd in dagen. Mijn idee was om dit als globale planning aan te houden en tijdens het ontwikkelen van de applicaties Scrum toe te passen.

In het begin verliep deze planning helemaal volgens plan. Zo had ik na week drie het afstudeerplan af, de cursus Xamarin zo goed als voltooid, toegang tot Azure devops voor de ontwikkelomgeving en het eerste gesprek gehad met mijn afstudeerbegeleider en begeleiders vanuit Capgemini.

Hierna zat er wel een fout in de planning: er stond namelijk geen tijd ingeboekt voor interviews met het team. Tijdens mijn vorige stage had ik hier veel tijd voor ingeboekt omdat ik interviews deed met externe partijen. Waarom ik dit nu vergeten ben weet ik niet, gelukkig viel de schade mee omdat ik dezelfde week nog met de vier participanten een interview kon inplannen. Hierdoor heb ik tijdens de ontwerpfase toch geen vertraging opgelopen.

Na het ontwerpen ben ik volgens planning begonnen met ontwikkelen. Toen ben ik ook begonnen met het toepassen van Scrum. Hierbij had ik een board aangemaakt met de userstories en de daar bijhorende taken. Ook had ik een aantal taken in een sprint gezet en was ik goed van start gegaan. Doordat de sprints maar een week duurden was er weinig ruimte om fouten te maken. Hierbij merkte ik dat ik de 1^{e} sprint te veel hooi op mijn vork nam waardoor ik maar de helft van de sprint kon afronden. Hierdoor had ik moeite met het inplannen van taken voor de volgende sprint. Mijn oplossing was harder en langer doorwerken. Maar ik had hier achteraf beter een goede structuur in kunnen brengen.

Hiervoor had ik eerder hulp kunnen vragen aan mijn begeleiders. Dit merkte ik vooral toen er twee nieuwe young-professionals in het team kwamen. Zij kregen ook een opdracht waarbij ze heel gestructureerde sprints hadden. Ze lieten zo ook hun product zien wanneer er nog niet zoveel van was gerealiseerd, ik liet het pas zien toen het echt wat was. Hierdoor was het team minder betrokken bij mijn voortgang.

Na het ontwikkelen van de applicatie was het benchmarken van de applicatie aan de beurt. De planning hiervoor was om eerst de benchmark van Xamarin af te ronden en daarna de benchmark met .NET MAUI. Dit is uiteindelijk aangepast naar eerst drie weken benchmark scripts schrijven en de week erop de benchmarks doorlopen. Dit is veranderd omdat ik in het begin zat te denken om microbenchmarks te gaan doen. Hiervoor is er een bibliotheek nodig en code die deze benchmarks uit voert. Helaas is dit nog niet mogelijk voor .NET MAUI. Hierdoor is er gekozen voor macrobenchmarks die de resultaatapplicaties testen buiten het framework om. Het voordeel hiervan is dat deze benchmarks ook voor andere frameworks of native applicaties gebruikt kunnen worden. Dit is uiteindelijk helemaal volgens planning gegaan.

Volgende keer zou ik bij zo’n groot project iets meer buffers plaatsen. Nu haalde ik de planning met een krappe marge, maar zou bijvoorbeeld een plotselinge ziekte dit totaal om kunnen hebben gegooid. Dit kreeg ik ook mee van mijn begeleiders bij de tussentijdse beoordeling: *“er lijkt niet veel buffer in de planning te zitten”*.

5.6.2 Proces

Het begin van mijn afstudeeropdracht begon met opzetten en afleggen van interviews. Dit was niet nieuw voor mij omdat ik tijdens mijn vorige stage ook interviews heb gedaan met meerdere bedrijven. Deze interviews waren wel iets anders doordat dit volledig inging op het ontwikkelproces en ervaringen van het team. Doordat ik voor het eerste interview nog niet precies het proces in kaart had, lukte me het eerst niet om gerichte vragen te stellen. Na het eerste interview heb ik mijn vragen herzien en gerichtere vragen kunnen stellen waardoor ik betere resultaten zag. Volgende keer zou ik me eerst iets beter inlezen in het onderwerp zodat ik direct deze vragen kon stellen.

Daarna ben ik begonnen met het ontwerpen van de applicatie. Mijn afstudeerdocent (Heiko) gaf tijdens het kennismakingsgesprek aan dat er goed gekeken moet worden hoe het bedrijf werkt. Hierdoor leek het mij een goed idee om inzicht te vragen in een van de software architectuur documenten die het team eerder gemaakt heeft. Hieruit zag ik veel overeenkomsten van de manier van ontwerpen die ik vanuit school heb geleerd en hoe het team het doet. Wel vond ik het duidelijker wanneer de functionele en technische ontwerpen in aparte documenten stonden. Dit gaf ik aan bij mij begeleider en daar kon hij het ook in vinden.

Nadat deze documenten klaar waren heb ik deze gedeeld met mijn begeleiders. Hier hebben ze toen feedback op gegeven, dit kwam neer op dat het inhoudelijk allemaal klopte alleen qua spelling en grammatica het nog beter kon. Hier baalde ik van omdat dit makkelijk te voorkomen was, hierna heb ik alle documenten eerst goed gecontroleerd op deze punten. Ook heb ik hiervoor hulp gevraagd in mijn omgeving.

Veel van het proces dat ik doorlopen heb was voor mij nieuw. Xamarin, .NET MAUI en het benchmarken van mobiele applicaties. Hierdoor heb ik veel obstakels moeten overwinnen om uiteindelijk een mooi resultaat neer te zetten. De grootste verandering die ik zou doorvoeren wanneer ik dit traject opnieuw zou gaan doen is beginnen met kleine proof of concepts.

Zo had ik voordat ik de planning maakte al kunnen testen of microbenchmarken mogelijk was voor het .NET MAUI platform met een proof of concept. Wanneer ik dit had gedaan zou ik meteen mijn planning en plan kunnen aanpassen aan wat er nodig is. Dit was vooral nodig omdat het een nieuwe techniek is waar nog weinig over te vinden is op het internet. Het idee van dit soort functionaliteiten testen in een proof of concept kreeg ik mee vanuit het team. Zij kregen van een klant een wens die technisch uitdagend was. Om hier onderbouwd op in te kunnen gaan hebben ze dit eerst getest met een proof of concept.

Tijdens het ontwikkelen van de .NET MAUI applicatie zijn er drie grote updates geweest. Vanaf de laatste preview, naar release candidate 1 t/m 3. Dit ging niet zonder slag of stoot. Zo moesten bij iedere update de packages ook bijgewerkt worden. Hierdoor kon ik of wachten op de beheerder van de package, of zelf het stuur in hand nemen. Dat laatste heb ik gedaan waardoor ik veel geleerd heb en anderen heb kunnen helpen op Github. Iets wat ik wel geleerd heb met het doorlopen van deze updates is om het altijd eerst te proberen, en een update is niet altijd beter. Zo had Microsoft bij release candidate 2 een probleem met de iOS certificaten waardoor er niet meer gebouwd kon worden voor iOS. Dit is uiteindelijk pas bij de volgende release opgelost, en doordat er geen makkelijke manier was om een versie terug te gaan heeft dit mij veel tijd gekost.

5.6.3 Doelen

Mijn grootste doel voor deze periode was het verder verkennen van de mogelijkheden in mobiele applicatie ontwikkeling. Ongeveer een jaar geleden kwam ik voor het eerst in aanraking met het ontwikkelen van mobiele applicaties door bij mijn minor sensortechniek een Android applicatie te ontwikkelen. Dit vond ik erg interessant en ik heb dit meegenomen voor mijn 3^{e} jaars stage. Toen heb ik een mobiele applicatie voor het eerst cross-platform ontwikkeld. Dit deed ik destijds in Flutter, een redelijk nieuw framework ontwikkeld door Google.

Dat je zo snel applicaties kon maken voor Android en iOS tegelijkertijd was voor mij op dat moment ongelofelijk. Dit zorgde ervoor dat ik mijn afstuderen ook in de mobiele applicatie ontwikkeling wilde gaan doen. Deze opdracht sloot perfect hierbij aan, doordat het een verbreding en verdieping was in de mobiele wereld.

Ik ben erg tevreden met wat ik heb neergezet, hiervoor moest ik veel leren. Van de frameworks Xamarin en .NET MAUI tot welke eigenschappen van een applicatie invloed hebben op gebruikerservaring en conversieratio van de installaties. Waar ik ook erg blij mee ben is het uitzoeken en ontwikkelen van de benchmark scripts. Hiermee is het mogelijk om mobiele applicaties te testen op opstarttijd, werkgeheugen en processorgebruik. Wat hieraan zo mooi is, is dat het zowel voor Android als iOS werkt en framework onafhankelijk. Hierdoor kan het ook gebruikt worden om andere frameworks zoals MAUI met de vergelijking mee te nemen of om performance van applicaties tussen sprint releases te vergelijken.

5.6.4 A-Competenties

5.6.4.1 Onderzoek

Om het onderzoek tot een succes te maken zijn er interviews en een experiment uitgevoerd. Daarnaast is er gebruikt gemaakt van deskresearch voor een aantal beschrijvende deelvragen die in het theoretisch kader al beantwoord konden worden.

Het experiment heeft geresulteerd in statistische onderbouwing van de hoofdvraag en er is een conclusie getrokken die als basis geldt voor het advies rapport.

Volgende keer zou ik het onderzoek splitsen in twee documenten. Doordat de deelvraag “Welke functionaliteiten moet een benchmark applicatie hebben om te lijken op een realistische applicatie van het HTC?” ook gebruikt wordt voor het ontwerp van de benchmark applicatie kan het verwarrend worden.

5.6.4.2 Leren leren

Tijdens en na mijn afstudeerperiode heb ik vanuit school één evaluatiemoment gehad en vanuit Capgemini twee. Hierin hebben mijn begeleiders mij goede feedback gegeven. Bij het eerste moment gaven ze aan dat ik volgende keer extra buffer moest inplannen en dat ik documenten eerst goed moet doornemen op spellingsfouten voordat deze ingeleverd kunnen worden. Ik heb de feedback serieus genomen en met resultaat. Zo kwam er uit de laatste evaluatie: *“Punten uit tussentijdse beoordeling goed opgepakt (zoals niet te vroeg dingen laten reviewen)” (Zie Bijlage 3: Eindevaluatie).*

Tijdens mijn school, werk en privéleven ben ik zelfstandig. Dit heeft mij veel geholpen maar kan ook zo nu en dan tegen mij werken. Dit merkte ik bij het implementeren van Scrum deze periode. Hierbij had ik beter wat advies kunnen vragen. Maar voor de rest heeft dit goed uitgepakt, ook mijn begeleiders gaven dit aan tijdens de tussentijdse beoordeling en eindevaluatie, prettig te ervaren: *“Vanaf begin zelfstandig in organiseren van werk en gestructureerd z'n onderzoeksplan aflopen.” (Zie Bijlage 3: Eindevaluatie)*

Met deze opdracht en de begeleiders heb ik alle vrijheid gekregen voor mijn eigen invulling. Dit maakte mij enthousiast om uit te zoeken wat er allemaal mogelijk was. Dit heeft uiteindelijk geresulteerd in een benchmark script dat het team eenvoudig kan gebruiken voor nieuwe en bestaande applicaties. Hier ben ik erg tevreden mee en ik hoop dat deze scripts nog veel gebruikt gaan worden.

5.6.4.3 Professioneel werken

Zoals aangegeven in 7.6.1. is de globale planning die aan het begin gemaakt is, zo goed als gevolgd. Hierin zijn kleine wijzigingen aangebracht door de bevindingen op dat moment maar ik heb dit altijd bij de hand gehouden. Dit reflecteert zich ook in de eindevaluatie van mijn begeleiders: *“Je hebt je eigen planning goed gevolgd en komt volgens mij goed uit aan het einde, dus dat gaat goed.” (Zie Bijlage 3: Eindevaluatie)*

Naast de algehele planning heb ik tijdens het ontwikkelen gebruik gemaakt van de Scrum methodiek. Hierbij had ik 4 sprints van een week, waarin zowel de ontwikkeling van de Xamarin als de .NET MAUI applicatie gedaan werd. Iedere sprint gaf ik aan welke userstories opgepakt werden vanuit de backlog. Onder de userstories vallen losse taken die gedaan moeten worden. Ook stond er in iedere userstory een “definition of done” die aangeeft wanneer de userstory succesvol is afgerond.

De communicatie met mijn begeleiders ging de gehele periode goed. Dit kwam voornamelijk doordat we dagelijkse standups hadden met het gehele team. Hier spreekt iedereen uit wat ze gedaan hebben, wat ze gaan doen en of ze ergens tegenaan lopen. Hierdoor wist het team altijd waar ik mee bezig was en konden ze mij ook wel eens helpen. Naast de formele communicatie viel ik voor mijn gevoel ook goed in de groep. Dit gevoel groeide in deze periode alleen maar meer en ik zie het team nu echt als collega’s. Dit gevoel bevestigden mijn begeleiders ook in mijn tussentijdse evaluatie: “*Luisteren en formuleren van gedachten gaat goed. Ligt goed in het team, bij medewerkers en leiding.*” (Zie bijlage 2: Tussentijdse evaluatie)

Het geven van feedback deed ik vooral tijdens de interventie van Hogeschool Leiden. Hier ging ik het werk van een medestudent beoordelen. In dit geval beoordeelde ik het verslag van Melissa Basgol. Op moment van de interventie had ze twee competenties aan te tonen. Die heb ik beoordeeld vanuit mijn perspectief. Ook gaf ik haar advies over de structuur van het verslag, hierbij gaf ik aan dat het niet lekker las dat er eerst diep ingegaan werd in de huidige situatie van het proces en daarna pas verteld werd over de organisatie waar ze gewerkt heeft. Wanneer je dit omdraait leert de lezer eerst iets over de organisatie en daarna duik je meer het proces in. Dit is in mijn ogen een stuk beter te volgen.

Daarnaast heb ik ook feedback kunnen geven aan twee junior ontwikkelaars die tijdens mijn afstudeer periode in het team kwamen. Hun kregen een interne opdracht om ervaring op te kunnen doen met .NET. Door mijn eerdere ervaringen met .NET had ik hun kunnen adviseren om een andere structuur aan te houden in de database. Deze aanpassing resulteerde in minder NULL-waardes bij entiteiten die overerven.

5.6.4.4 Innovatie

Vooral in het benchmarken van de applicaties kon ik mijn creativiteit kwijt. Hierbij werd ik volledig vrijgelaten. Dit heeft uiteindelijk geresulteerd in het gebruik maken van de debug tools van Android en iOS. Deze tools heb ik zo gebruikt dat ik de handmatige testen kon automatiseren. Hierdoor zijn vele handmatige stappen die moeilijk te herproduceren zijn, vereenvoudigd in een script.

Dit project had ik aangenomen met een risico dat .NET MAUI nog niet klaar zou zijn voor de benchmarks. Dit was in februari (start van het afstuderen) gepland in kwartaal één, dus binnen twee maanden. Helaas werd dit uitgesteld. Gelukkig kwam de release candidate uit precies toen ik begon met ontwikkelen. Wanneer dit niet het geval was geweest had ik verder moeten gaan met de preview versie, wat wellicht niet correcte resultaten zou hebben gegeven van het platform.

5.6.5 B-Competenties

5.6.5.1 Software analyseren

Het onderzoek is begonnen met het in kaart brengen van het ontwikkelproces van applicaties van het HTC team. Hiervoor is een interview uitgevoerd waarbij vier ontwikkelaars van het team zijn ondervraagd (zie 7.2.2.). Het resultaat hiervan is gecodeerd en gebruikt als basis voor de functionele eisen in samenspraak met de begeleiders. Naast de functionele eisen is er ook gekeken naar de technische eisen. De bibliotheken die gebruikt worden moeten namelijk ook te gebruiken zijn in .NET MAUI. Hiervoor is er allereerst gevraagd aan de participanten welke bibliotheken ze gebruikt hebben in de eerder gebouwde applicaties. De bibliotheken die hieruit kwamen zijn verder onderzocht op compatibiliteit met .NET MAUI.

5.6.5.2 Software adviseren

Voor het adviseren is er een onderzoek gedaan naar de performance en ontwikkel verschillen tussen Xamarin en .NET MAUI. Om dit inzichtelijk te maken is er gekeken naar de ontwikkelprocessen die volgens de ontwikkelaars van het HTC verbeterd kunnen worden. Tijdens het ontwikkelen van de applicaties zijn deze processen gedocumenteerd en is er gekeken naar de verschillen. Hieruit is gebleken dat .NET MAUI deze processen een stuk efficiënter kan uitvoeren wat resulteert in minder ontwikkel en ontwerp tijd (Zie 5.2.4.).

Daarnaast is er een experiment uitgevoerd in de vorm van een benchmark. Met deze benchmark is er gekeken naar de performanceverschillen tussen Xamarin en .NET MAUI. Hiervoor zijn twee applicaties ontwikkeld waarbij het enige verschil het framework is waarin het is ontwikkeld. Hierdoor zullen de verschillen in performance de performance van het framework reflecteren.

Het resultaat van het onderzoek is gebruikt als onderbouwing voor het advies. Dit advies heb ik gepresenteerd aan het gehele team, waarna ik het verwerkt heb in een rapport en dit gedeeld heb met het team. Hierin staan de kosten en baten aangegeven (zie 5.5.).

5.6.5.3 Software ontwerpen

Bij het ontwerpen van de benchmarkapplicatie zijn alle aspecten van software ontwerpen aan bod gekomen. Dit heb ik verwerkt in een functioneel en technisch ontwerp wat ik verder toegelicht heb (zie 5.3.). Bij het ontwerpen van de applicatie staat het HTC centraal. Tijdens het onderzoek was de focus om te kijken welke verschillen er zijn voor het HTC bij de overstap naar .NET MAUI. Hierbij hoorde ook dat de benchmarkapplicatie gebouwd werd op de standaarden van het team. Zo heb ik toegang gekregen tot een van de software architectuur documenten van de grootste klant van het team. Hier heb ik veel geleerd over de manier van werken, architecturale keuzes en gebruikte technieken.

Dit voorbeeld gaf mij een goed beeld van hoe het HTC de ontwerpfase aanpakt. Ik zag ook veel van het document terug in wat we op school hebben geleerd. Ik gaf wel aan dat het voor mij beter gestructureerd is wanneer de functionele keuzes in een ander document staan dan de technische en grafische keuzes. Hierin konden mijn begeleiders zich vinden waardoor ik dit in mijn project heb doorgevoerd.

In het technische ontwerp is er ook veel input gekomen vanuit het team. Zo moet het gebruik maken van een MVVM architectuur. Dit is een architectuur die veel gebruikt wordt in .NET frontend applicaties en ook altijd gebruikt wordt door het HTC.

6 Bronnen

Apple. (2018, 9 april). *Asset Catalog Format Reference: Types Overview*. Developer.Apple.Com. Geraadpleegd op 14 juni 2022, van https://developer.apple.com/library/archive/documentation/Xcode/Reference/xcode_ref-Asset_Catalog_Format/AssetTypes.html

Capgemini. (2017). *Capgemini history*. Geraadpleegd op 14 juni 2022, van https://www.capgemini.com/wp-content/uploads/2017/07/capgemini_history.pdf

Capgemini. (2022, 9 juni). *About Us - Strategic Partner To Companies Around The World*. Geraadpleegd op 14 juni 2022, van https://www.capgemini.com/about-us/

Chaudhary, S. (2022, 13 februari). *13 Feb Battery and CPU utilization testing for Android*. Fleek IT Solutions. Geraadpleegd op 14 juni 2022, van https://www.fleekitsolutions.com/battery-and-cpu-utilization-testing-for-android/#:%7E:text=If%20an%20application%20consumes%20more,is%20used%20by%20your%20applic

Editor. (2020, 27 februari). *The Good and The Bad of Xamarin Mobile Development*. AltexSoft. Geraadpleegd op 21 april 2022, van https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/

Google. (2021, 16 november). *Improving App Startup: Lessons from the Facebook App*. Android Developers Blog. Geraadpleegd op 14 juni 2022, van https://android-developers.googleblog.com/2021/11/improving-app-startup-facebook-app.html

Google, Flutter. (2022). *Adding assets and images*. Flutter. Geraadpleegd op 21 april 2022, van https://docs.flutter.dev/development/ui/assets-and-images

Johansen, J. (2022, 5 januari). *Flutter Hot Reload - Flutter*. Medium. Geraadpleegd op 21 april 2022, van https://medium.com/flutter/flutter-hot-reload-f3c5994e2cee

Kelly, V. (2022). *Error adding FreshMvvm.Maui to RC1 Maui project · Issue #5 · XAM-Consulting/FreshMvvm.Maui*. GitHub. Geraadpleegd op 21 april 2022, van https://github.com/XAM-Consulting/FreshMvvm.Maui/issues/5

Khan, R. H. (2022, 14 januari). *Hot Reloading in React Native*. Delft Stack. Geraadpleegd op 21 april 2022, van https://www.delftstack.com/howto/react/hot-reloading-in-react-native/

Lyalin, D. (2021, 23 oktober). *Update on .NET Hot Reload progress and Visual Studio 2022 Highlights*. ..NET Blog. Geraadpleegd op 21 april 2022, van https://devblogs.microsoft.com/dotnet/update-on-net-hot-reload-progress-and-visual-studio-2022-highlights/

Lyalin, D. & Microsoft. (2021, 25 mei). *Introducing the .NET Hot Reload experience for editing code at runtime*. ..NET Blog. Geraadpleegd op 28 februari 2022, van https://devblogs.microsoft.com/dotnet/introducing-net-hot-reload/

Matos, L. (2022, 11 mei). *Tips for Moving Your Xamarin Library to .NET MAUI*. Xamarin Blog. Geraadpleegd op 14 juni 2022, van https://devblogs.microsoft.com/xamarin/tips-for-porting-your-xamarin-library-to-dotnet-maui/

Microsoft. (2022a, april). *What is .NET? An open-source developer platform.* Geraadpleegd op 21 april 2022, van https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

Microsoft. (2022b, april 21). *What is .NET MAUI? - .NET MAUI*. Microsoft Docs. Geraadpleegd op 21 april 2022, van https://docs.microsoft.com/en-us/dotnet/maui/what-is-maui

Microsoft. (2022c, mei 23). *What is Azure DevOps? - Azure DevOps*. Microsoft Docs. Geraadpleegd op 14 juni 2022, van https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops

Mo, C. (2021, 21 mei). *Why Should You Care About Your Mobile App's Startup Time?* Embrace Blog and Events. Geraadpleegd op 14 juni 2022, van https://blog.embrace.io/why-should-you-care-about-your-mobile-apps-startup-time/

Morris, W. (2020, 28 augustus). *What is an SVG File (And How Do You Use it)? - Elegant Themes Blog*. Elegant Themes. Geraadpleegd op 21 april 2022, van https://www.elegantthemes.com/blog/wordpress/what-is-an-svg-file-and-how-do-you-use-it

Mulder, P. (2022, 25 januari). *MoSCoW Methode voor het stellen van prioriteiten - project tool | toolshero*. Toolshero. Geraadpleegd op 11 juni 2022, van https://www.toolshero.nl/project-management/moscow-methode/

Ortinau, D. (2022, 25 mei). *Introducing .NET MAUI - One Codebase, Many Platforms*. ..NET Blog. Geraadpleegd op 1 juni 2022, van https://devblogs.microsoft.com/dotnet/introducing-dotnet-maui-one-codebase-many-platforms/

Peppers, J. (2022, 8 juni). *Performance Improvements in .NET MAUI*. ..NET Blog. Geraadpleegd op 13 juni 2022, van https://devblogs.microsoft.com/dotnet/performance-improvements-in-dotnet-maui/

Raciborski, P. (2022). *Future of MVVMCross with .NET 6.0 and MAUI · Issue #4138 · MvvmCross/MvvmCross*. GitHub. Geraadpleegd op 14 juni 2022, van https://github.com/MvvmCross/MvvmCross/issues/4138

React Native. (2022, 30 maart). *PixelRatio · React Native*. Geraadpleegd op 21 april 2022, van https://reactnative.dev/docs/pixelratio

Sims, G. (2022, 18 februari). *Apple vs Android RAM management: Who does it better?* Android Authority. Geraadpleegd op 14 juni 2022, van https://www.androidauthority.com/apple-vs-android-ram-management-3100032/

Stackoverflow. (2018). *Stack Overflow Developer Survey 2017*. Geraadpleegd op 21 april 2022, van https://insights.stackoverflow.com/survey/2017

Stackoverflow. (2019). *Stack Overflow Developer Survey 2018*. Geraadpleegd op 21 april 2022, van https://insights.stackoverflow.com/survey/2018

Stackoverflow. (2021, 28 februari). *Stack Overflow Developer Survey 2020*. Geraadpleegd op 21 april 2022, van https://insights.stackoverflow.com/survey/2020#technology

Tolomei, S. (2022, 5 mei). *Shrinking APKs, growing installs - Google Play Apps & Games*. Medium. Geraadpleegd op 14 juni 2022, van https://medium.com/googleplaydev/shrinking-apks-growing-installs-5d3fcba23ce2

Valdellon, L. (2020, 2 november). *What Are the Different Types of Mobile Apps? And How Do You Choose?* CleverTap. Geraadpleegd op 21 april 2022, van https://clevertap.com/blog/types-of-mobile-apps/

Van der Knaap, N. (2022, 1 april). *HTC design document*. Figma. Geraadpleegd op 14 juni 2022, van https://www.figma.com/file/xbM2ptXYKnL92aGss2RDaL/HTC-design-document?node-id=128%3A1334

Versluis, G. (2022). *[iOS] Could not load file or assembly "Xamarin.iOS.Windows.Client, Version=1.0.0.0" A strongly-named assembly is required. · Issue #5629 · dotnet/maui*. GitHub. Geraadpleegd op 21 april 2022, van https://github.com/dotnet/maui/issues/5629

7 Bijlagen

7.1 Bijlage 1: Productrisico-analyse

Kenmerk / Onderdeel	Risicoklasse	Argumentatie
Functionaliteit		
- Inloggen & Registreren (UC01 & UC02)	**Hoog**	De gebruiker van de applicatie moet een account aan kunnen maken en kunnen inloggen om de andere functionaliteiten van de applicatie te kunnen gebruiken.
- Overzicht laden steden (UC04)	**Hoog**	Het hoofdscherm na het inloggen zal het scherm zijn dat de steden opvraagt vanuit de backend API en toont aan de gebruiker. Alleen wanneer de steden correct laden, is het mogelijk om te navigeren naar de detailpagina van een stad. Als dit faalt leidt dit ertoe dat meerdere must have usecases niet meer uitgevoerd kunnen worden en dus grote schade geeft aan de applicatie.
- Details tonen stad (UC05)	**Hoog**	Op de detailpagina wordt de informatie getoond van de geselecteerde stad. Vanaf hier kan ook een afbeelding gemaakt worden, die gekoppeld wordt aan de stad. Ook hierdoor zijn een aantal must have usecases niet meer uit te voeren, wat zich reflecteert in schade.
- Foto toevoegen aan stad (UC06)	**Hoog**	Bij het toevoegen van een afbeelding aan een stad heeft de applicatie toegang nodig tot een camera. iOS en Android hebben beide een eigen API hiervoor, waardoor de foutkans groter is. Daarnaast is dit een must usecase voor de applicatie.
- Navigeren tussen views (UC08)	**Hoog**	Wanneer het navigeren tussen views niet werkt, is de gehele applicatie onbruikbaar. Zo moet er na het inloggen genavigeerd kunnen worden naar de steden overzicht. Hierdoor is de mogelijke schade groot.
- Stad toevoegen (UC09)	**Midden**	Alleen wanneer de gebruiker niet eerder een stad heeft toegevoegd vallen er functionaliteiten zoals: navigeren naar detail pagina & afbeelding toevoegen weg. Dit is dus een risico voor nieuwe gebruikers die must have functionaliteiten kan beïnvloeden.
- Uitloggen (UC03)	**Laag**	Wanneer een ingelogde gebruiker niet meer kan uitloggen, zal de volledige applicatie nog te gebruiken zijn. Om uit te kunnen loggen is er alleen toegang nodig tot de secure storage wat al bevestigd is bij het inloggen. Hierdoor is de foutkans klein.
- Foto verwijderen (UC07)	**Laag**	Wanneer een bestaande foto niet verwijderd kan worden zal maar één usecase met een lage prioriteit niet meer uit te voeren zijn.
- Stad verwijderen (UC10)	**Laag**	Wanneer een bestaande stad niet verwijderd kan worden zal maar één usecase met een lage prioriteit niet meer uit te

		voeren zijn.
Beveiliging		
- Rechten (bezoeken beveiligde view)	**Hoog**	Dit zorgt ervoor dat de applicatie de juiste werking heeft. Wanneer de gebruiker verder kan navigeren zonder juiste inlog zal de applicatie niet werken naar behoren.
- Toon alleen eigen steden	**Hoog**	Het is belangrijk voor de privacy van de gebruiker dat de applicatie alleen steden en afbeeldingen kan laden en toevoegen die onder de ingelogde gebruiker bestaan.
Gebruiksvriendelijkheid		
- Opslag	**Midden**	Een van de meetpunten van de applicatie is de opslag gebruik. Dit kijkt naar de grootte van de applicatie op het apparaat. Onderzoek heeft laten blijken dat voor iedere 6MB dat een applicatie groter is, dit de conversieratio met 1% vermindert *(Tolomei, 2022, "Does APK size impact install conversion rate?" sectie).*
Performance		
- Opstartsnelheid	**Hoog**	De opstartsnelheid van de applicatie is de eerste interactie tussen de gebruiker en de applicatie. Deze actie wordt meerdere keren herhaald en kan impact hebben op de aandacht van de gebruiker. Wanneer dit te lang duurt kan de gebruiker de actie stoppen en zo de applicatie niet meer gebruiken.
- Werkgeheugengebruik	**Hoog**	Werkgeheugen is beperkt op een telefoon en moet dus gedeeld worden tussen het systeem en alle applicaties. Hierdoor heeft het werkgeheugengebruik van een applicatie impact op de snelheid van multitasken. Daarnaast is er een relatie tussen het gebruik van werkgeheugen en batterijconsumptie.
- Processorgebruik	**Hoog**	Ook processorgebruik heeft invloed op het multitasken op het apparaat. Daarnaast is hier ook een relatie te zien met processorgebruik en batterijconsumptie.
Betrouwbaarheid		
- Toegang tot lokale opslag	**Hoog**	De toegang tot lokale opslag is nodig voor het lokaal opslaan van de afbeeldingen. Deze afbeeldingen kunnen worden toegevoegd bij een detailpagina van een stad. Omdat deze use case een must is, is de mogelijke schade groot. Daarnaast is er ook een verhoogde foutkans doordat iOS en Android dit met hun eigen API's afhandelen, waardoor er twee of meer methodes onderhouden moeten worden.
- Toegang tot beveiligde opslag	**Hoog**	De toegang tot het beveiligde gedeelte van de opslag wordt gebruikt om de login token in op te slaan. Wanneer hier geen toegang toe is kan er niet ingelogd worden en kan de token ook niet meegestuurd worden naar de backend. Dit resulteert in een volledig niet werkende applicatie. Daarnaast is er ook een verhoogde foutkans doordat iOS en Android dit met hun eigen API's afhandelen, waardoor er twee of meer methodes onderhouden moeten worden.

- Toegang tot camera	**Hoog**	Toegang tot de camera van het apparaat is nodig om een afbeelding onder een stad toe te voegen. Doordat dit een must have usecase is, is de mogelijke schade groot. Daarnaast is er ook een verhoogde foutkans doordat iOS en Android dit met hun eigen API's afhandelen, waardoor er twee of meer methodes onderhouden moeten worden.
- Bereikbaarheid	**Laag**	De applicatie zal niet in een productie omgeving gaan draaien. Hierdoor hoeft de applicatie alleen bereikbaar te zijn wanneer de benchmarks worden uitgevoerd.

7.2 Bijlage 2: Tussentijdse evaluatie

Onderdeel	O	V	G	Opmerkingen
Inzicht • kan snel inwerken • plannen en organiseren • besluiten nemen		X		Besluiten over te gebruiken libraries genomen. Inwerken in Xamarin/MAUI deed je snel. Liep een paar dagen achter op planning, er lijkt niet veel buffer in de planning te zitten. MAUI preview kapot, hoe ga je daarmee om?
Communicatie • luisteren • gedachten formuleren • contact met medewerkers • contact met de leiding • samenwerken • omgaan met kritiek • schriftelijke vaardigheden • documenteren			X	Documenten soms te snel ter review aangeboden, met nog te veel spelfouten erin. Inhoudelijk zijn documenten goed. Luisteren en formuleren van gedachten gaat goed. Ligt goed in het team, bij medewerkers en leiding. Ging goed om met feedback op pull requests.
Eigenheid • kunnen motiveren / overtuigen • ontplooien van initiatieven • eigen inbreng in besprekingen • creatief • behulpzaam • zelfstandig		X		Gaat zeer zelfstandig te werk. Neemt initiatief voor keuzes als resource files, libraries en kan overtuigen in waarom de keuze juist is. Creativiteit zien we graag terugkomen in hoe je gaat benchmarken.

Stiptheid • afspraken nakomen • analytisch vermogen • inzet / werktempo • kritisch t.a.v. eigen functioneren		X		Uitspreken verwachting wanneer iets af is heb je al verbeterd, dat gaat nu goed. Analytisch vermogen zien we graag terugkomen in het benchmarken. Werktempo is goed, voorbeeldapp zat snel in elkaar en Pull Request review opmerkingen snel verwerkt. Kritisch op eigen functioneren ben je ook, je geeft zelf aan wat je "fails" zijn in de stagiair pitches.
Overig (vakinhoudelijke kennis en vaardigheden)			X	Gaat heel goed. Een aantal punten kunnen we later pas goed inschatten als je verder bent met je onderzoek.

7.3 Bijlage 3: Eindevaluatie

Onderwerp: Xamarin meets MAUI

Legenda

1. Onvoldoende	Resultaat / houding zijn ondermaats
2. Voldoende	Resultaat / houding zijn op het randje, maar persoon laat ruimte voor verbetering zien
3. Goed	Resultaat / houding is vergelijkbaar met dat van mijn collega's
4. Uitmuntend	Pakt extra taken op, resultaat is boven verwachting.

Accountability

Is op tijd aanwezig of meldt zich af bij afwezigheid	4	*Ook structureel aanwezig op kantoordagen*
Werkt zelfstandig	4	*Vanaf begin zelfstandig in organiseren van werk en gestructureerd z'n onderzoeksplan oflopen.*
Neemt initiatief	4	*Komt met oplossingen voor issues waar die tegenaan loopt.*
Stressbestendig en flexibel	4	*Tegenslagen met MAUI releases waarbij dingen kapot gingen enzo, daar goed mee omgegaan.*

Inzet

Aanwezig bij pitches	4	*Staat er elke keer.*
Aanwezig bij (verplichte) activiteiten	4	*Aanwezig*
Neemt deel aan extra activiteiten		*Weet ik niet*
Zichtbaar gegroeid tijdens de stageperiode	3	*Begon al met professionele werkhouding. Veel geleerd over mobile development.*

Competenties

Communicatie, *duidelijk en doelgericht*	4	*Heel helder in wat hij doet, gaat doen en welke uitdagingen er overwonnen zijn of op tafel liggen.*
Communicatie, *Nederlandse spelling en grammatica*	3	*Schrijven is volgens mij niet je favoriet.*
Communicatie, *mbt verwachtingsmanagement*	4	*Geeft duidelijk aan wat we kunnen verwachten.*
Time management	3	*Lang doorgegaan met techniek, lijkt alsof je intensief werk aan documenten een beetje vooruitgeschoven hebt.*
Presentatie skills	3	*Presenteren gaat goed.*
Omgang met Feedback, *positief en negatief*	3	*Punten uit tussentijdse beoordeling goed opgepakt (zoals niet te vroeg dingen laten reviewen)*
Effective Writing, *doelgericht en duidelijk*	3	*Schrijven is volgens mij niet je favoriet.*
Big Picture Thinking, *hebben van een helicopterview en gevolge*	3	*Je hebt je eigen planning goed gevolgd en komt volgens mij goed uit aan het einde, dus dat gaat goed.*

Aanbeveling

Ik ben er enthousiast over dat je onze collega wordt. Vanaf het begin heb je gedegen gewerkt, je gaf telkens aan waar je je op focuste en wat de tussentijdse resultaten en bevindingen waren. Het eindresultaat mag er zijn, herbruikbare tools om performance tussen verschillende apps te kunnen benchmarken. Waarbij we dus hergebruik kunnen toepassen voor andere vergelijkingen dan MAUI - Xamarin. Documenten schrijven is volgens mij niet je favoriete werkzaamheid maar dat is niet ongebruikelijk voor developers. Als tip zou ik je meegeven om op tijd te beginnen met documenten schrijven om tijdgebrek aan het einde te voorkomen .

About Capgemini

Capgemini is a global leader in partnering with companies to transform and manage their business by harnessing the power of technology. The Group is guided everyday by its purpose of unleashing human energy through technology for an inclusive and sustainable future. It is a responsible and diverse organization of over 325,000 team members more than 50 countries. With its strong 55-year heritage and deep industry expertise, Capgemini is trusted by its clients to address the entire breadth of their business needs, from strategy and design to operations, fueled by the fast evolving and innovative world of cloud, data, AI, connectivity, software, digital engineering and platforms. The Group reported in 2021 global revenues of €18 billion.

Get the Future You Want | www.capgemini.com