Een onderzoek naar de maturiteit van .NET MAUI bij het vervangen van een real-time POS-systeem.

Optionele ondertitel.

Michiel Van Herreweghe.

Scriptie voorgedragen tot het bekomen van de graad van
Professionele bachelor in de toegepaste informatica

Promotor: Mevr. M. Van Audenrode
Co-promotor: Dhr. K. Van Moorter
Academiejaar: 2022–2023
Eerste examenperiode

Departement IT en Digitale Innovatie .

HO
GENT

Woord vooraf

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

Lijst van figuren

1

Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (**Pollefliet2011**):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1. Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgelijnd zijn. Doelgroepen als “bedrijven,” “KMO’s”, systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2. Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. "welke tools bestaan er voor deze toepassing?") is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specifiëren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3. Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

1.4. Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:
In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.
In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.
In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2

Stand van zaken

Inleiding

In dit hoofdstuk worden een aantal zaken besproken met het oog op de contextualisering van de studie. Allereerst zal kort er in gegaan worden op de programmeertaal COBOL. Daarna wordt het verschil belicht tussen native en cross-platform ontwikkeling, elk met zijn eigen voor- en nadelen. Nadien volgt een bespreking van het .NET MAUI framework. Tenslotte zal ook het onderwerp real time toepassingen worden toegelicht.

2.1. COBOL

2.1.1. Een korte geschiedenis van COBOL

In de jaren vijftig was de informatica vooral gericht op de rekenkracht van computers, die vooral in de wiskunde en de wetenschap kon worden gebruikt. Financiële instellingen zagen echter ook de voordelen van computertoepassingen in het bedrijfsleven. In 1959 pleitte Mary Haws voor de ontwikkeling van een programmeertaal die kon worden gebruikt voor zakelijke doeleinden, zoals salarisadministratie, inventarisatie en analyse van bedrijfsgegevens (National Museum of American History, 2013). In mei 1959 werd het Committee on Data Systems Languages, bekend als CODASYL, opgericht en gefinancierd door het Ministerie van Defensie om een nieuwe programmeertaal te ontwerpen en te ontwikkelen die voldeed aan de criteria van Mary Haws. Deze taal kreeg de naam COmmon Business-Oriented Language, of COBOL (Abby, 2023). De ontwikkeling van COBOL was gebaseerd op drie pijlers: leesbaarheid (de code moet leesbaar zijn voor zowel programmeurs als leken), overdraagbaarheid (toepassingen moeten snel en gemakkelijk kunnen worden overgezet naar een ander systeem) en flexibiliteit (de taal moet modulair genoeg zijn om zich aan te passen aan veranderende behoeften en technologieën) (Abby, 2023). In december 1960 slaagden zij erin COBOL te draaien op zowel een

UNIVAC II systeem als een RCA machine (National Museum of American History, 2013).

2.1.2. De huidige staat van COBOL

Vandaag de dag wordt COBOL nog steeds gebruikt. De studie van Reuters (2017) toont aan dat 43 procent van de bankindustrie nog steeds steunt op COBOL. Sterker nog 95 procent van de pinautomaten steunen nog steeds aan op de programmeertaal. Volgens de studie van Micro Focus (2022) staan er momenteel 800 miljard lijnen COBOL in productie. Daarnaast gaven de bevraagden ook aan dat de hoeveelheid COBOL in productie nog zou verhogen. Voor 92 procent van de bevraagden blijven de COBOL-applicaties een strategische asset, die met der tijd ook gemoderniseerd zullen worden. Een voorbeeld hiervan is het integreren van cloud omgevingen en COBOL-applicaties (Micro Focus, 2022).

2.2. Native VS cross-platform development

2.2.1. Wat is native development?

Van native ontwikkeling is sprake wanneer een programmeur een toepassing ontwikkelt die alleen op een bepaald platform kan draaien, zoals iOS of Android. Hiervoor moet de programmeur de programmeertaal en tools (Marchuk, g.d.) gebruiken die het gekozen platform biedt. Om native Android-toepassingen te schrijven, moet de ontwikkelaar Java of Kotlin gebruiken. Talen die native door het iOS-platform worden ondersteund zijn Objective-C en Swift (Schmitt, 2022).

2.2.2. Wat is cross-platform development

Cross-platform ontwikkeling, ook bekend als multi-platform ontwikkeling, is een manier van ontwikkelen waarbij met één programmeertaal, en dus één codebase, een applicatie kan worden ontwikkeld die op verschillende platforms kan draaien (Kotlin Foundation, 2022).

2.2.3. Een vergelijking tussen native en cross-platform development

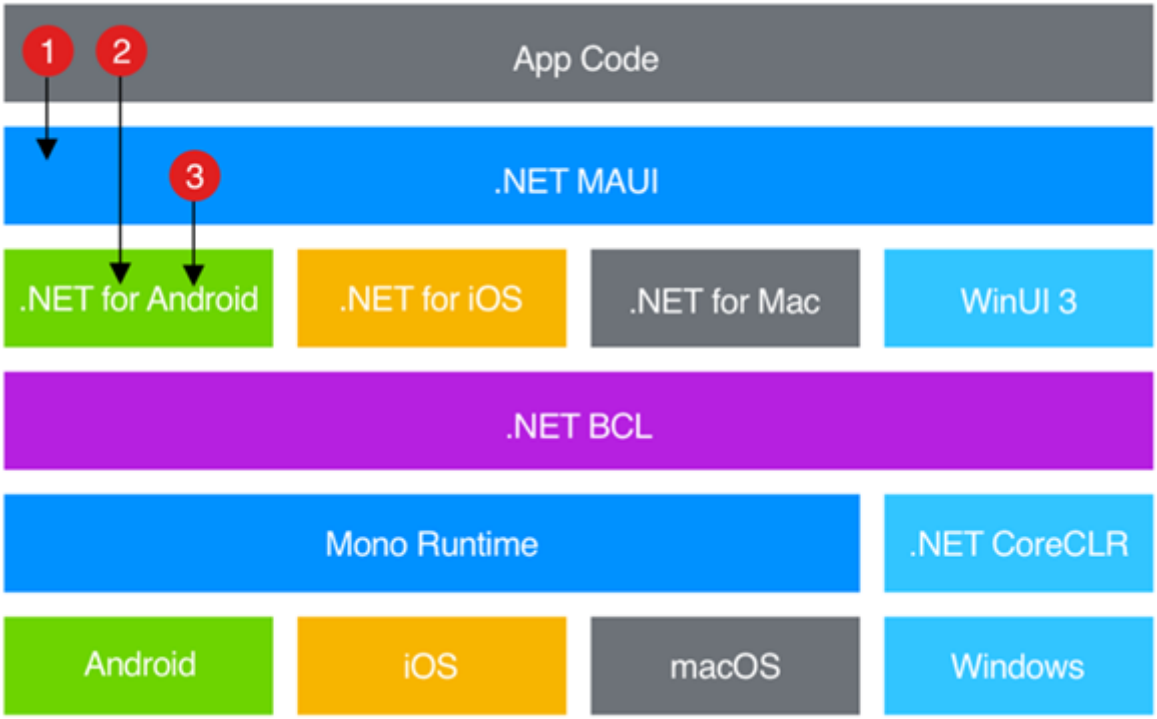
Voordelen	Nadelen
Hoge performantie De programmeertaal en API's van het doelplatform zijn geoptimaliseerd voor maximale prestaties.	Hogere ontwikkelkost Als een applicatie voor zowel iOS als Android moet worden ontwikkeld, moeten er twee verschillende teams van ontwikkelaars aan werken.
Uniforme UX (User eXperience) Elke native applicatie gebruikt dezelfde interface-elementen die door het platform worden geleverd. Hierdoor zien apps er consistenter uit en gedragen ze zich consistenter. Dit maakt een nieuwe applicatie intuïtiever en sneller te gebruiken.	Grotere ontwikkelteams Zoals gezegd zijn er meerdere teams nodig om verschillende applicaties voor de verschillende platforms te ontwikkelen. Binnen deze teams zal een groot aantal experts nodig zijn om hun expertise in te brengen tijdens het ontwikkelingsproces.
Toegang tot de gehele featureset van een platform Zoals vermeld in het eerste voordeel, kan native ontwikkeling gebruik maken van de API's die het platform biedt. Dit kan het gebruik zijn van de camera, GPS, pushmeldingen etc.	Verschillende codebases met risico op meer bugs Aangezien een applicatie meerdere programmeertalen vereist, is het gevolg dat er verschillende codebases zullen zijn. Dit kan leiden tot meer bugs omdat er meer regels code nodig zijn.
	Native applicaties kunnen verschillende logica-implementaties hebben Omdat elk platform een andere native programmeertaal biedt, kan het implementeren van bepaalde logica tot fouten leiden omdat de taal de logica anders interpreteert. Hetzelfde artikel kan bijvoorbeeld een verschillende prijs hebben op de twee platforms omdat de prijsberekening anders is geïmplementeerd.

Tabel 2.1: De voor- en nadelen van native development

Voordelen	Nadelen
Deelbare code Met cross-platform ontwikkeling kunnen ontwikkelaars meerdere platforms bereiken met één enkele codebasis. Dit maakt een stuk code ook overdraagbaar van het ene project naar het andere.	Lagere performantie Omdat de cross-platform taal meerdere platformen moet kunnen aanspreken, kan de taal niet volledig worden geoptimaliseerd voor een bepaald platform, waardoor de prestaties op één platform iets langzamer zijn.
Sneller ontwikkelproces Er hoeven minder regels code te worden geschreven en getest, wat het ontwikkelingsproces aanzienlijk versnelt.	Beperkte toegang tot de gehele featureset van een platform Sommige platformen geven de moedertaal alleen toegang tot bepaalde API's of functies. Een voorbeeld hiervan is het weigeren van toegang tot pushmeldingen.
Hoge kosteneffectiviteit Kleinere ontwikkelingsteams kunnen meerdere platforms met dezelfde programmeertaal aanspreken, waardoor de ontwikkelingskosten dalen.	Beperkte UX-consistentie Cross-platform programmeertalen bieden elk hun eigen interfacecomponenten. Dit betekent dat cross-platform toepassingen vaak verschillende UI's hebben.
Gedeelde logica Het feit dat de toepassing is ontwikkeld met een enkele codebase betekent dat kan worden aangenomen dat de geïmplementeerde logica consistent is voor alle platforms.	

Tabel 2.2: De voor- en nadelen van cross-platform development

Zoals uit Tabel 2.1 Kotlin Foundation (2023) en Tabel 2.2 Kotlin Foundation (2023) op te maken is, is het debat over native VS cross-platform ontwikkeling niet eenvoudig. Elk heeft zijn voor- en nadelen, en voor elk project moet een grondige analyse worden gemaakt om te bepalen welke van deze zaken het belangrijkst zijn voor het project. Als een project hoge prestaties vereist en hoge ontwikkelingskosten minder belangrijk zijn, kunt u ervoor kiezen de toepassingen in de native programmeertaal te ontwikkelen.
Als het project daarentegen door een klein team wordt ontwikkeld en de applicatie zo snel mogelijk op beide platforms beschikbaar moet zijn, kan voor cross-platform ontwikkeling worden gekozen.



Figuur (2.1)
Schematische voorstelling van de bouwstenen van .NET MAUI

2.3. .NET MAUI

2.3.1. Overzicht van wat .NET MAUI is en het doel ervan

.NET MAUI staat voor .NET Multi-platform App UI en is de opvolger van Xamarin.Forms. Terwijl Xamarin.Forms diende als cross-platform programmeertaal die alleen op mobiele platformen kon worden gebruikt, kan .NET MAUI ook worden gebruikt om desktop en Smart TV applicaties te ontwikkelen. NET MAUI is vanaf de grond opgebouwd met uitbreidbaarheid en prestaties in het achterhoofd. Xamarin en .NET MAUI hebben veel overeenkomsten, maar de laatste maakt het ook mogelijk om platform-specifieke code toe te voegen (Brady Gaster, 2020).

2.3.2. De werking van .NET MAUI

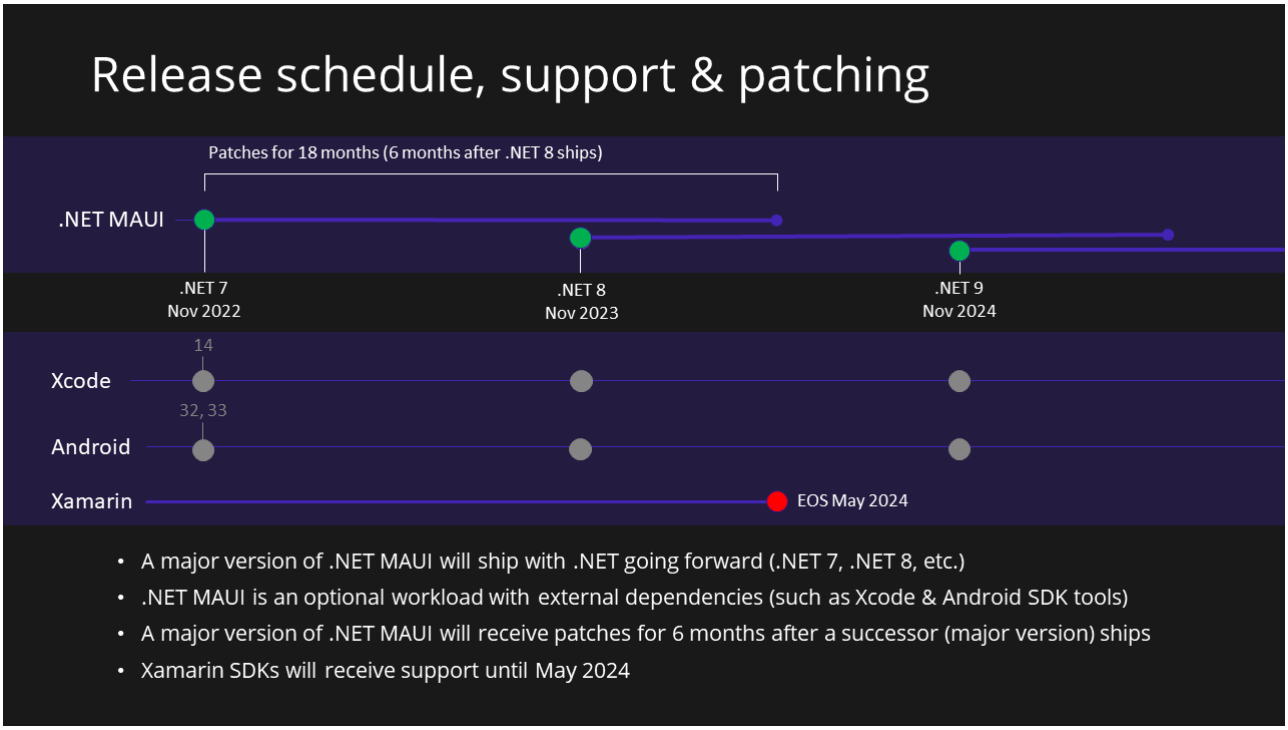
.NET MAUI voorziet in een framework per platform waarmee in C# geschreven code kan worden gecompileerd voor het gewenste platform. Momenteel heeft .NET MAUI vier platformspecifieke frameworks: .NET voor Android, .NET voor iOS, .NET voor Mac en de Windows UI 3 bibliotheek. Elk van deze frameworks gebruikt de .NET Base Class Library, kortweg BCL, die fungeert als abstractie tussen de geschreven broncode en de verschillende platformen. De BCL vertrouwt op de .NET runtime om de toepassing te compileren. Ook hier zijn verschillende platformspecifieke tools voorzien. Zo is er de Mono runtime voor Android, iOS en macOS, en de

.NET CoreCLR voor Windows desktop applicaties (Britch, 2023).
Met BCL kunt u een toepassing maken die op verschillende platforms draait met dezelfde codebasis. Elk platform heeft echter zijn eigen manier om de gebruikersinterface te definiëren, evenals de manier waarop de verschillende elementen met elkaar communiceren en interageren. Dit is waar .NET MAUI u de keuze geeft om verschillende gebruikersinterfaces voor verschillende platformen te creëren of, omgekeerd, één gebruikersinterface voor alle platformen te creëren (Britch, 2023).

2.3.3. Een vergelijking tussen .NET MAUI en Xamarin

Aangezien .NET MAUI de opvolger is van Xamarin.Forms, zijn er veel overeenkomsten, maar ook veel verschillen. In deze sectie worden de twee frameworks met elkaar vergeleken.
Het eerste grote verschil tussen .NET MAUI en Xamarin is de projectstructuur. In Xamarin wordt bij het aanmaken van een project een projectmap per platform aangemaakt, met als gevolg dat platformspecifieke code en resources (zoals fonts, afbeeldingen...) per project moeten worden onderhouden. Met de .NET MAUI wordt een projectmap aangemaakt die meerdere platformmappen bevat. In deze mappen wordt platformspecifieke code opgeslagen (Koleva, 2023).
Het volgende verschil tussen de twee programmeertalen is de build tooling, .NET MAUI kan de cross-platform .NET CLI (Command Line Interface) gebruiken om projecten te compileren en uit te voeren. Terwijl Xamarin.Forms het .NET framework moet gebruiken om applicaties te bouwen (Kathiresan, 2022).
Een derde belangrijk verschil tussen Xamarin.Forms en de .NET MAUI zijn de platformspecifieke API's. Xamarin biedt een enkele cross-platform API die werkt op Android, iOS en Windows. Met .NET MAUI wordt voor elk platform een API geleverd, waardoor ontwikkelaars kunnen profiteren van de functies van elk platform (UXDivers, g.d.).
Ten slotte kunt u met .NET MAUI kiezen om de GUI (Graphical User Interface) te bouwen met HTML (HyperText Markup Language) of XAML (Extensible Application Markup Language). Terwijl je met Xamarin.Forms alleen met XAML kon werken (Kathiresan, 2022).

2.3.4. Een blik op de toekomst van .NET MAUI



Figuur (2.2)
.NET MAUI release schema

Elke nieuwe versie van .NET zal vanaf nu ook een major versie van .NET MAUI bevatten. Elk van deze major versies zullen 18 maanden ondersteuning krijgen (Ramel, 2022).
Daarnaast zal de support voor Xamarin.Forms stopgezet worden in mei 2024. Concreet wil dit zeggen dat er geen updates of patches meer zullen uitgebracht worden voor Xamarin.Forms (Ramel, 2022).

2.4. Real-time applicaties

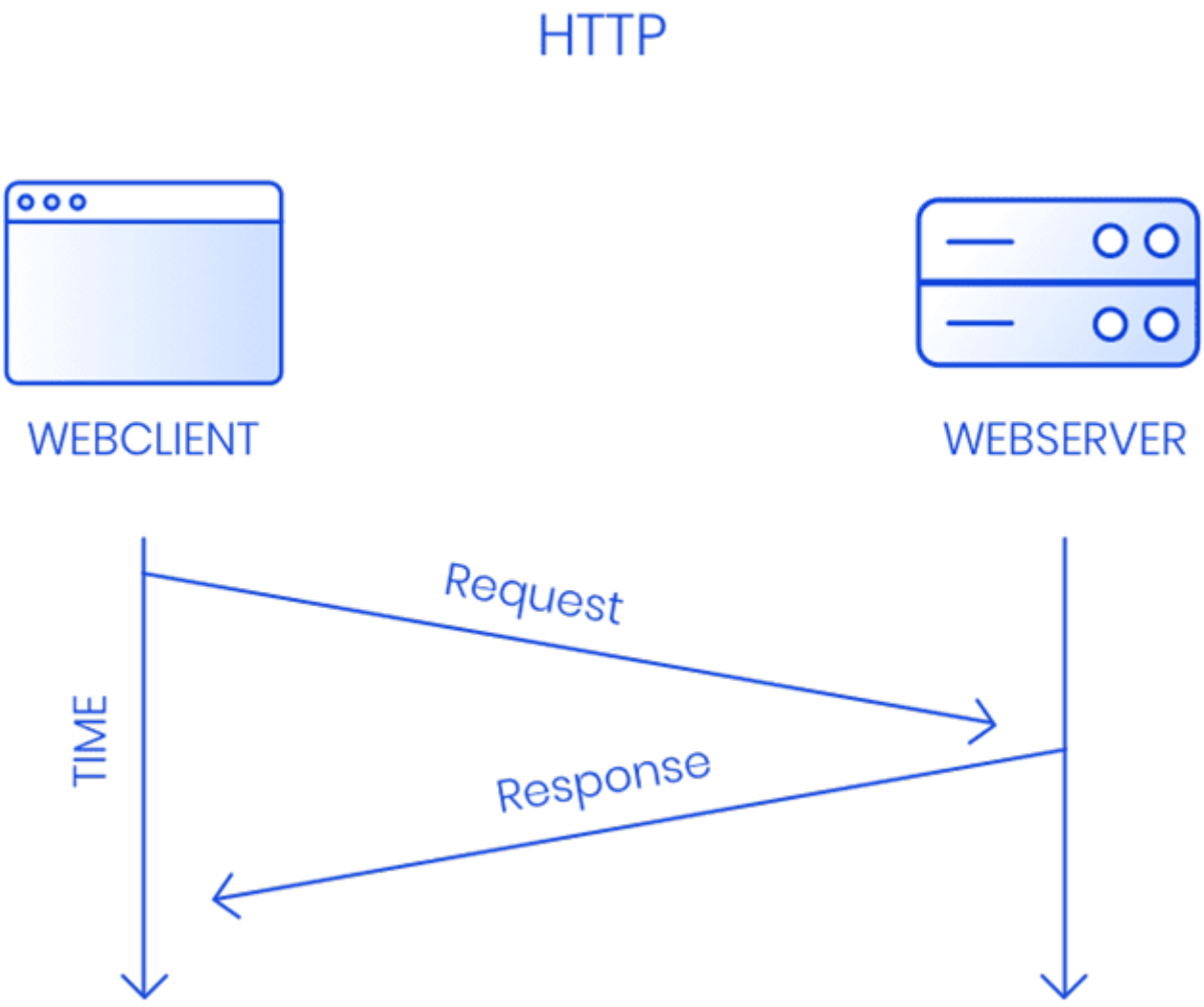
2.4.1. Definitie van een real-time applicatie

Real-time computing verwijst naar computersystemen die zijn ontworpen om gegevensinvoer in real time te verwerken en erop te reageren, doorgaans binnen milliseconden of microseconden nadat gebeurtenissen zich hebben voorgedaan.
De reactietijden moeten onmiddellijk aanvoelen en het systeem moet zonder vertraging of onderbreking doorgaan.
Real-time computing is van cruciaal belang voor toepassingen die onmiddellijke en nauwkeurige gegevensverwerking vereisen, zoals industriële controle, medische toepassingen, luchtvaart, defensie, multimedia, games en financiële handelssystemen.
Real-time computing vereist gespecialiseerde hardware- en softwarecomponen-

ten, waaronder real-time besturingssystemen, real-time netwerken en synchrone programmeertalen, om het systeem binnen het gewenste tijdsbestek te laten functioneren.

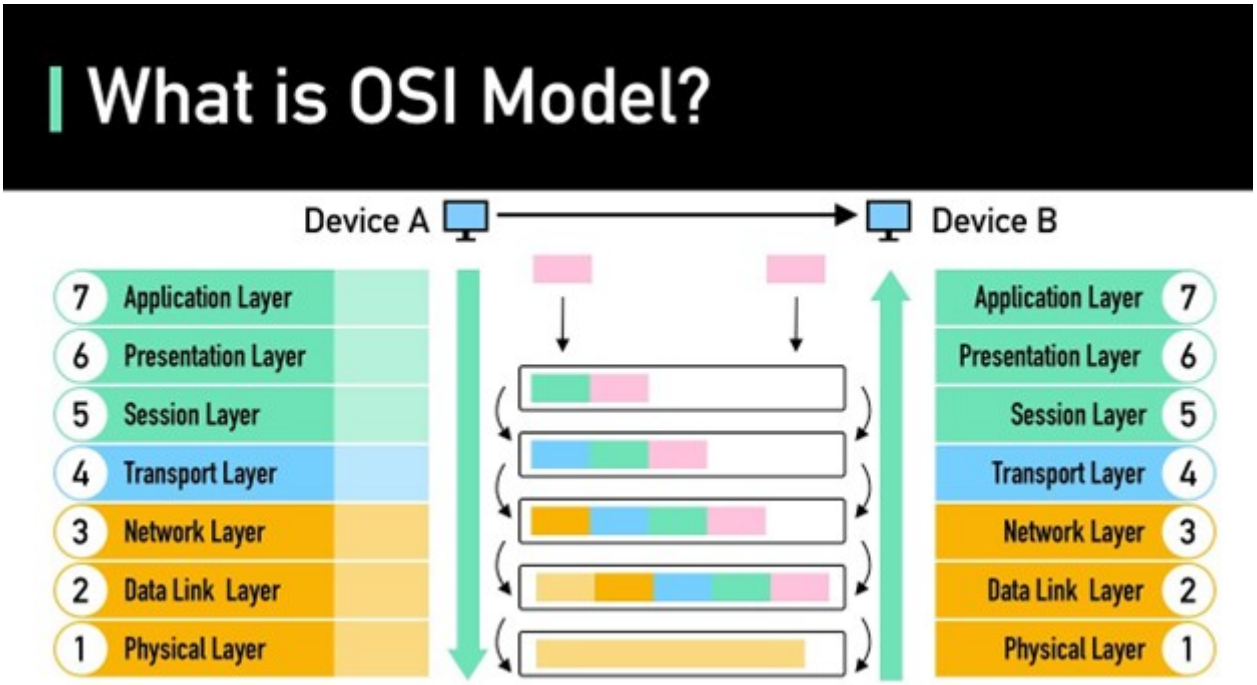
2.4.2. HyperText Transfer Protocol

HTTP is het protocol dat ten grondslag ligt aan het internet. Het protocol wordt gebruikt om gegevensbronnen (zoals HTML, XML, JSON-documenten) op te vragen bij een server (Cloudflare, g.d.).



Figuur (2.3)
Schematische voorstelling van het server-client model

HTTP is een client-serverprotocol, wat betekent dat verzoeken om gegevens altijd door de client (ook bekend als de ontvanger) worden geïnitieerd. Het verzoek wordt naar de server gestuurd, die op zijn beurt het gevraagde document samenstelt en terugstuurt als antwoord op het verzoek (MDN, 2023a)



Figuur (2.4)
Het OSI model

Dit protocol werd oorspronkelijk voorgesteld door Tim Berners-Lee in 1989 en werd gebouwd bovenop de TCP- en IP-protocollen (MDN, 2023b).
Het HTTP-protocol bevindt zich in de zevende laag (ook toepassingslaag genoemd) van het OSI-model (MDN, 2023a)
De filosofie van HTTP rust op vier pijlers:

- Uitbreidbaarheid

  De invoering van HTTP-headers in HTTP versie 1 (HTTP/1) (MDN, 2023b) maakt het mogelijk extra informatie toe te voegen aan verzoeken en antwoorden, zolang client en server deze overeenkomst aanvaarden (Paessler, g.d.).

- Staatloos

  Staatloos betekent dat tijdens en na het verzenden en ontvangen van een verzoek en antwoord geen informatie door de server wordt bijgehouden. Met andere woorden, elk verzoek dat de server ontvangt wordt onafhankelijk gezien en uitgevoerd. Een stateless protocol is ontworpen met het oog op schaalbaarheid. Afhankelijk van de werklast kunnen dus meerdere servers worden opgezet met dezelfde informatie, en elke server in de configuratie kan het verzoek ontvangen en verwerken (Paessler, g.d.).

  Om een toestand bij te houden kunnen HTTP-cookies worden gebruikt. Dit is mogelijk dankzij de bovengenoemde uitbreidbaarheid en maakt het moge-

lijk bepaalde informatie (zoals authenticatietokens) te delen tussen meerdere verzoeken (MDN, 2023a).

- Verbindingsloos

  Het HTTP-protocol wordt om twee redenen als verbindingsloos beschouwd. Omdat verbindingen worden gemaakt door de transportlaag (laag 4 van het OSI-model), valt het niet onder de verantwoordelijkheid van een applicatie laag (laag 7 van het OSI-model) applicatie (MDN, 2023a).

  Bovendien worden voor elk verzoek verbindingen gelegd tussen de client en de server. Zodra het verzoek is verwerkt, sluit de server de verbinding (Paessler, g.d.).

- Media onafhankelijk

  Zolang de client en de server weten hoe zij bepaalde door het MIME-type (Multipurpose Internet Mail Extensions) gespecificeerde gegevens moeten verwerken, kunnen de gegevens via het HTTP-protocol worden getransporteerd (Paessler, g.d.).

De evolutie van het HTTP-protocol

HTTP/0.9

Aanvankelijk had het HTTP-protocol geen versienummer (aangegeven door het cijfer achter het teken /). Om de verschillende versies van elkaar te scheiden, kreeg de allereerste implementatie van het HTTP-protocol echter versienummer 0.9 (MDN, 2023b).

Met verzoeken die bestonden uit een enkele regel bestaande uit het type verzoek en de naam van de gegevensbron, was deze versie van HTTP vrij primitief. Ook kon met deze versie alleen een GET-verzoek worden gedaan. Bijgevolg konden alleen gegevens worden gelezen (Grigorik, g.d.)

HTTP/1.0 HTTP-versie 1.0 werd uitgebracht in 1990, vijf jaar nadat versie 0.9 was geïntroduceerd, en voegde nieuwe functies toe om eerdere tekortkomingen te verhelpen:

- HTTP-header

  Zoals hierboven vermeld, bestond een verzoek van versie 0.9 alleen uit de methode en de naam van de gegevensbron die werd opgevraagd (Fulber-Garcia & Fulber-Garcia, 2022). Door de toevoeging van de HTTP-header kunnen nu metadata worden opgenomen in verzoeken en antwoorden, wat deze versie van het protocol flexibel en uitbreidbaar maakt (MDN, 2023b).

- Versiebeheer

  Elk HTTP-verzoek bevat nu de gebruikte HTTP-versie (MDN, 2023b).

- Statuscode

  Aan elk antwoord wordt een statuscode toegevoegd. Dit is een manier voor de cliënt om bij elk antwoord te controleren of het verzoek al dan niet correct is verwerkt (Fulber-Garcia & Fulber-Garcia, 2022).

- Content-type

  Zoals hierboven vermeld, kunnen met de HTTP-header extra metadata aan een verzoek worden toegevoegd. Voor HTTP/1.0 is nu een inhoudstype in de header opgenomen, waardoor ook andere documenten dan HTML kunnen worden verzonden (Fulber-Garcia & Fulber-Garcia, 2022).

- Extra methoden

  HTTP/1.0 heeft ten slotte twee nieuwe methoden toegevoegd, POST en HEAD. Dit zorgt ervoor dat gegevens nu zowel geschreven als gelezen kunnen worden (Fulber-Garcia & Fulber-Garcia, 2022).

HTTP/1.1

HTTP/1.1 werd een jaar na de release van HTTP/1.0 geïmplementeerd. Dit was slechts bedoeld als een verbetering van de 1.0-versie:

- Host header

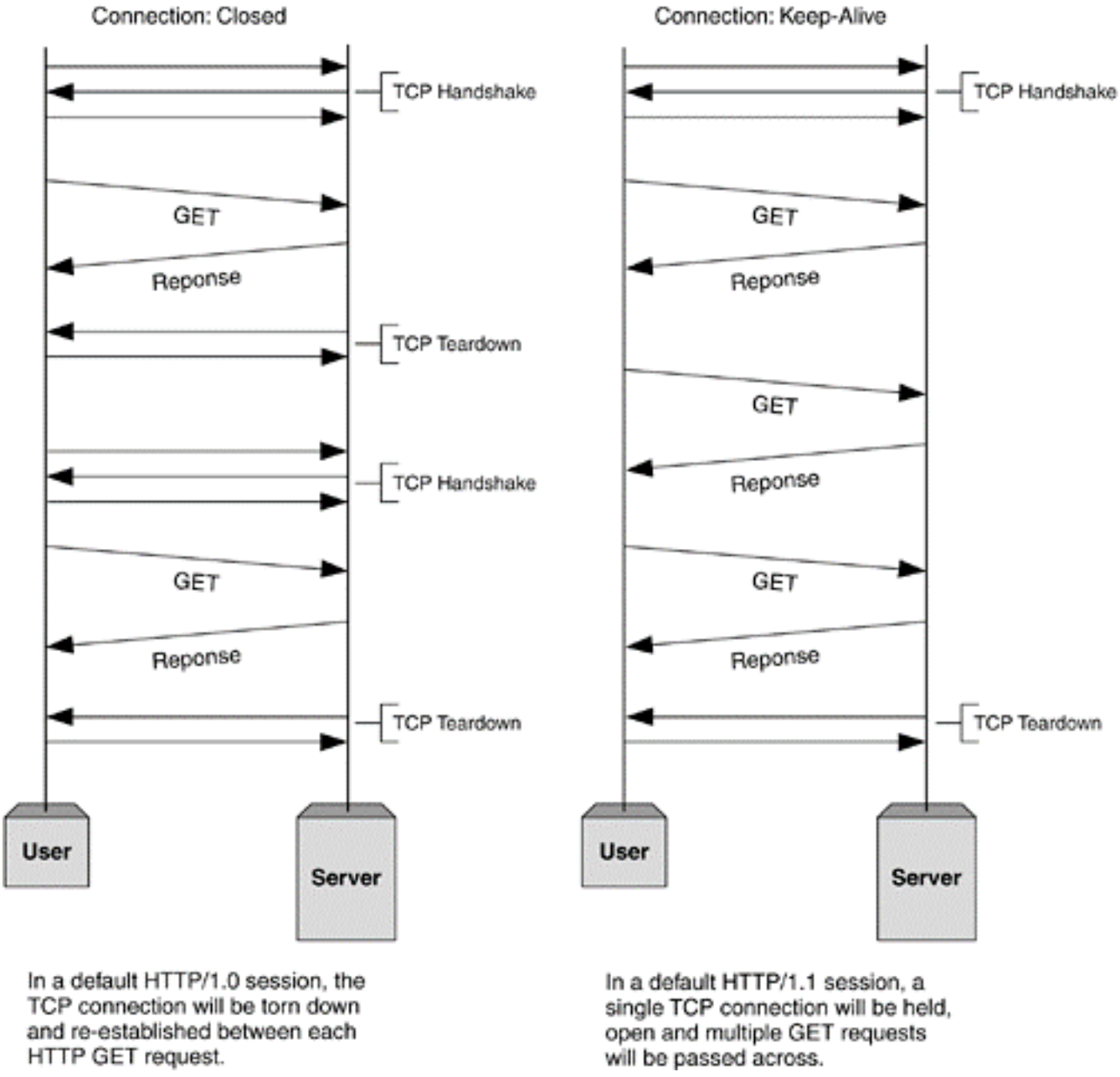
  Met HTTP/1.1 moest de client het host veld toevoegen aan de header van elke aanvraag. Vóór deze versie was dit niet verplicht. Dankzij dit veld kunnen verzoeken via een proxyserver naar de juiste server worden geleid, met andere woorden, als er verschillende domeinen naar dezelfde locatie verwijzen, kan er toch een onderscheid worden gemaakt om ervoor te zorgen dat het verzoek niet naar de verkeerde server gaat (Fulber-Garcia & Fulber-Garcia, 2022).

- Persistente verbinding

  Zoals hierboven vermeld, moest de client in HTTP-versie 1.0 een verbinding maken voor elk verzoek dat hij wilde verzenden. In versie 1.1 kan de verbinding open worden gehouden totdat de client aangeeft dat hij de verbinding wil sluiten (MDN, 2023b)).

- Continue-status

  Vóór de invoering van de continue status konden sommige verzoeken niet door de server worden verwerkt. Als dit het geval was, werd het verzoek gewoon onbeantwoord afgewezen. Vanaf versie 1.1 kan een cliënt eerst de HTTP-headers van het verzoek aan de server doorgeven. Als de server antwoordt met de status continue (statuscode 100), weet de cliënt dat hij het hele verzoek kan doorsturen en een antwoord van de server kan verwachten (Fulber-Garcia & Fulber-Garcia, 2022).



Figuur (2.5)
Schematische voorstelling van het verschil tussen niet-herbruikbare en herbruikbare connecties van respectievelijk HTTP/1.0 en HTTP/1.1

- Aanvullende methoden

  Versie 1.1 introduceert ook een aantal nieuwe methoden. Deze omvatten PUT, PATCH, DELETE, CONNECT, TRACE en OPTIONS (Fulber-Garcia & Fulber-Garcia, 2022).

HTTP/2.0

HTTP/2.0 werd uitgebracht in 2015, 18 jaar na 1.1. Het is echter nog niet op grote schaal ingevoerd. Het gebruik van HTTP/2.0 is dan ook een bewuste keuze van de ontwikkelaars (Ab, g.d.). HTTP/2.0 brengt echter een aantal geweldige nieuwe functies die nieuwe mogelijkheden bieden, namelijk:

- Multiplexing

  In versie 1.1 van het HTTP-protocol werden verzoeken om gegevens sequentieel verwerkt. Dat wil zeggen, het eerste HTML-document werd opgevraagd en geladen, dan het tweede document, ... tot alles geladen was. Dit kon echter tot problemen leiden als een bepaalde bron niet kon worden geladen omdat de rest van de bron werd opgehouden. Met HTTP/2.0 kan een enkele TCP-verbinding worden gebruikt om meerdere verzoeken om gegevens tegelijk te doen. Dit zorgt ervoor dat wanneer het bovengenoemde probleem zich voordoet, de resterende gegevens al kunnen worden geladen (Cloudflare, g.d.).

- Server push

  Vóór de invoering van HTTP/2.0 kon de server alleen gegevens naar de client pushen nadat deze een verzoek had verzonden. In versie 2.0 is echter server push geïmplementeerd. Hierdoor kan de server gegevens naar de client sturen nog voordat de client een verzoek naar hem heeft gestuurd. Dit lost prestatieproblemen met lange polling op (Grigorik, 2016).

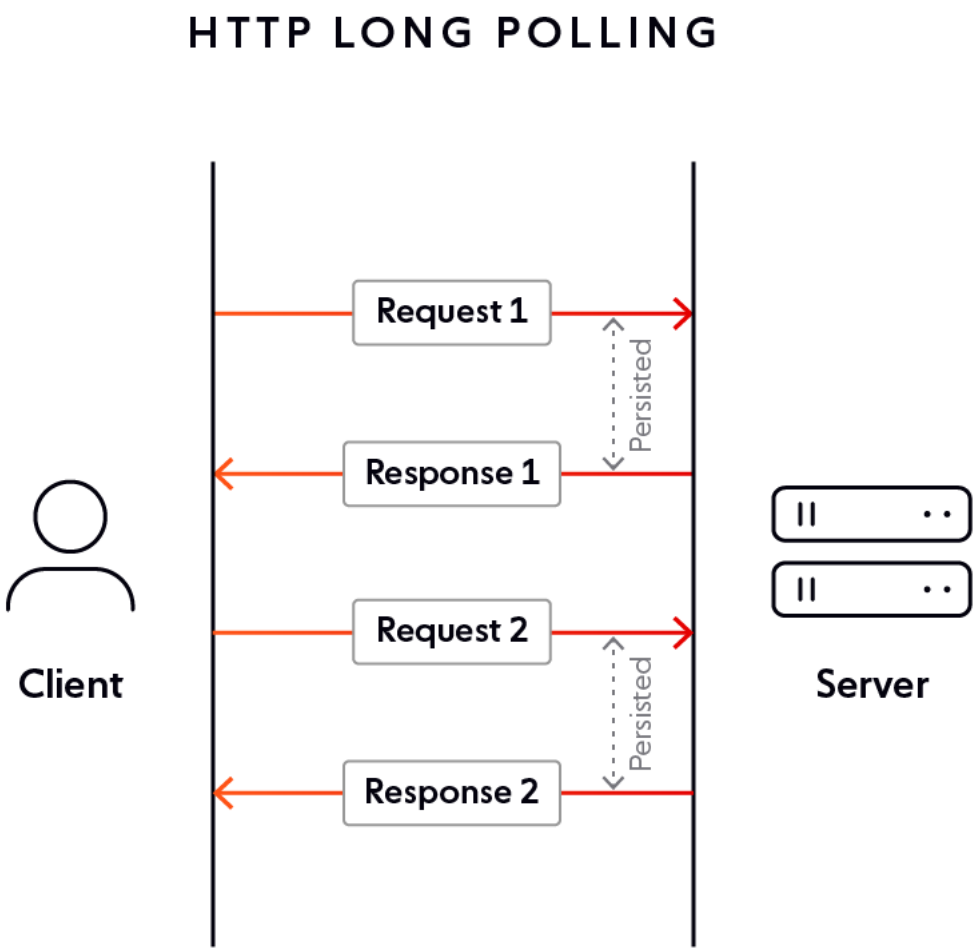
- Compressie van headers

  De toevoeging van HTTP-headers aan een verzoek vergroot de omvang van het verzoek. Dit kan leiden tot het langzamer laden van gegevens en het langzamer verwerken van gegevens door de client. Deze verzoeken kunnen kleiner worden gemaakt en sneller worden verwerkt door de headers te comprimeren. HTTP/1.1 deed dit al, maar versie 2.0 heeft een veel verfijnder compressie-algoritme, het HPACK-algoritme. Dit algoritme verwijdert alle overbodige informatie, waardoor verzoeken kleiner worden (Cloudflare, g.d.).

2.4.3. WebSocket protocol

Het WebSocket protocol is een relatief nieuw protocol. Het protocol werd voor het eerst beschreven in 2008 door Micahel Carter en Ian Hickson en kreeg algemene ondersteuning rond 2010 (Ably, 2020).

In december 2011 publiceerde het IETF (Internet Engineering Task Force) de white-paper die het WebSocket protocol beschreef onder de titel “RFC 6455 - The WebSocket Protocol” (Fette & Melnikov, 2011).
Volgens de paper van Fette en Melnikov (2011) werd het WebSocket protocol ontwikkeld om real-time applicaties mogelijk te maken zonder het zogenaamde “HTTP long polling” te misbruiken.
Bij HTTP long polling stuurt de client een aanvraag uit naar de server. In plaats van direct te antwoorden en daarna de connectie te sluiten, wacht de server met antwoorden totdat er nieuwe data beschikbaar is. Na het uitsturen van het antwoord krijgt de server direct een nieuwe aanvraag van de client (Singh, g.d.).
Het WebSocket protocol is full-duplex en laat bidirectionele communicatie toe. Concreet betekent dit dat in plaats van een aanvraag te sturen die enkel beantwoord wordt wanneer er nieuwe data is, is het nu mogelijk om data door te sturen naar de client zonder dat deze een aanvraag verstuurd heeft. Dit zorgt voor een betere performantie ten opzichte van het HTTP long polling (Fette & Melnikov, 2011)



Figuur (2.6)
Schematische voorstelling van HTTP long polling

2.4.4. Real-time programmeren binnen het .NET ecosysteem

De ASP.NET SignalR bibliotheek is beschikbaar voor real-time applicatieontwikkeling binnen het .NET ecosysteem. Deze bibliotheek maakt het mogelijk een tweerichtingscommunicatie op te zetten tussen een client en een server. De client kan met name bepaalde functies op de server aanroepen, maar de server kan ook functies opvragen bij de client (Brady Gaster, 2020).
In het traditionele client-server model kan alleen de client functies op de server aanroepen. Dit betekent dat als de gegevens zo actueel mogelijk moeten zijn, er extra logica moet worden geïmplementeerd om de server na een bepaald aantal seconden te polsen naar de (mogelijk) gewijzigde gegevens (deze implementatie wordt ook wel ”long polling” genoemd). Dit kan leiden tot slecht presterende toe-

passingen (Brady Gaster, 2020).
Dankzij het gebruik van SignalR is het nu ook mogelijk dat de server functies gaat aanroepen op de client. Dit zorgt ervoor dat het hierboven getoonde voorbeeld op een veel performantere manier kan worden geïmplementeerd. Wanneer gegevens op de server worden gewijzigd, kan de server alle luisterende clients opdragen de nieuwe gegevens op te halen. Dit zorgt ervoor dat gegevens alleen worden opgevraagd wanneer de gegevens veranderen, in plaats van telkens na een bepaalde periode (Brady Gaster, 2020).

3

Methodologie

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetuer quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at

lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

4

Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem. Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A

Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

Samenvatting

Native applicatie development is een tijdrovend en duur proces waarbij een applicatie ontwikkeld wordt voor verschillende platformen, zoals iOS en Android. Hier kan cross-platform development een goede oplossing bieden. Cross-platform laat toe om met één codebase in één en dezelfde programmeertaal een applicatie te ontwikkelen die op de meeste gebruikte platformen kan draaien. In deze bachelorproef zal er worden nagegaan of .NET MAUI al voldoende ontwikkeld is om een real-time kassa-applicatie, geschreven in COBOL, te vervangen met een productiewaardige cross-platform applicatie ontwikkeld in .NET. In de eerste fase van het onderzoek zal er nagegaan worden of het analoge signaal van een printer opgevangen en omgevormd kan worden met een Raspberry Pi. In de tweede fase zal er een real-time .NET MAUI-applicatie ontwikkeld worden die steeds kan worden geüpdate wanneer er een signaal van een printer binnenkomt. Tenslotte zal de .NET MAUI-applicatie gecompileerd worden naar een mobiel platform. Hier zal nagegaan worden of dit zonder enige probleem kan. Er wordt verwacht dat de .NET MAUI-applicatie sneller zal reageren dan de huidige COBOL-applicatie. Daarnaast wordt er ook aangenomen dat de user experience al zodanig ontwikkeld is dat er slechts kleine problemen zouden voorkomen bij het compileren naar een mobiel platform. De beoogde doelgroep voor dit onderzoek is elke ontwikkelaar die de overweging maakt om .NET MAUI te gebruiken om een cross-platform applicatie te ontwikkelen.

A.1. Introductie

Een bedrijf (Goossens NV) heeft enkele copycentra, waar ze momenteel nog steeds werken met een kassasysteem dat geschreven is in COBOL. Op het einde van de dag wanneer de winkel sluit, moet steeds een dagrapport worden gemaakt en deze worden dan op een floppydisk geplaatst. Aan het begin van afgelopen zomer stoot het bedrijf op het probleem dat er geen floppydisks meer geproduceerd worden. Dit heeft als gevolg dat de dagrapporten niet meer bewaard kunnen worden. In dit onderzoek zal worden nagegaan of .NET MAUI, een cross-platform programmeertaal, al voldoende ontwikkeld is om een nieuw real-time kassasysteem op te zetten.

A.2. State-of-the-art

A.2.1. Cross-platform development

Cross-platform development, ook wel multi-platform development genoemd, is een manier van ontwikkelen dat toelaat om met één programmeertaal en bijgevolg één codebase een applicatie te ontwikkelen die op verschillende platformen kan draaien (Kotlin Foundation, 2022).

Native development VS cross-platform development

De term “native development” geeft aan dat een applicatie uitsluitend ontwikkeld wordt voor één bepaald platform, bijvoorbeeld iOS. De applicatie wordt geprogrammeerd met de door dat platform voorziene programmeertaal en hulpmiddelen (Marchuk, g.d.).
Om native Android-applicaties te schrijven, moet de developer gebruik maken van Java of Kotlin. De talen, die dan weer native ondersteund worden door het iOS platform, zijn Objective-C en Swift (Schmitt, 2022).

.NET MAUI

.NET MAUI is de uitgebreide evolutie van Xamarin.Forms om niet alleen mobiele applicaties te ontwikkelen, maar ook desktopapplicaties. .NET MAUI verenigt Android, iOS, macOS en Windows API’s in een enkele API die het toelaat om met één codebase applicaties te ontwikkelen voor de verschillende platformen (Britch & Gechev, 2022).

A.2.2. Real-time applicaties

De term real-time applicatie wordt door Lutkevich (2022) als volgt gedefinieerd: “Real-time-toepassingen zijn toepassingen die binnen een onmiddellijk tijdskader werken; zij detecteren, analyseren en handelen op streaming gegevens terwijl die zich voordoen. Dit in tegenstelling tot een database-gerichte toepassing waarbij informatie wordt opgenomen en opgeslagen in een database (in de cloud of op locatie) voor toekomstige analyse.”

Om de werking van real-time applicaties te kunnen verduidelijken, moet het websocket protocol uitgelegd worden.

HyperText Transfer Protocol

HyperText Transfer Protocol, ook wel gekend als HTTP, is het standaard protocol dat webbrowsers gebruiken om informatie op te vragen bij de server. HTTP werkt volgens het client-server principe. Dit houdt in dat de client, in de meeste gevallen de webbrowser, steeds een dataverzoek stuurt naar de server. Deze zal op zijn beurt dan de gewenste data verzamelen en terugsturen als een document van een bepaald type. Dit kan bijvoorbeeld gaan over HTML-documenten voor webpagina's of JSON-documenten om data te transfereren (MDN, 2022).
Zoals hierboven beschreven is HTTP unidirectioneel. Dit wil zeggen dat de dataaanvragen enkel en alleen kunnen worden opgestart door een client en kunnen alleen beantwoord worden door de server. Na versturen van het antwoord, wordt de verbinding verbroken (MDN, 2022).

WebSocket

WebSocket is een bidirectioneel protocol die dezelfde noden vervult als HTTP, maar in tegenstelling tot HTTP is WebSocket een stateful protocol. Hiermee wordt bedoeld dat de verbinding tussen de client en de server in stand wordt gehouden ook al is de data aanvraag verwerkt. De enige manier om de verbinding te verbreken is wanneer één van de twee partijen besluit om de verbinding stop te zetten. Een groot verschil met HTTP is ook dat de communicatie bij WebSocket bidirectioneel is, wat wil zeggen dat de client de server kan aanspreken, maar ook dat de server de client kan aanspreken (GeeksforGeeks, 2022).

A.3. Methodologie

Het huidige kassasysteem werkt als volgt: een klant komt binnen in één van de copy centra en krijgt een printer aangewezen door de verko(o)p(st)er die op dat moment in de winkel staat. Eens de klant het printproces gestart heeft, wordt er per geprinte pagina een klik doorgestuurd naar een elektronisch bord. Dit bord weet op welke interface de klik binnengekomen is, welke printer het signaal verstuurd heeft en vertaalt dit dan naar een digitaal signaal dat verstuurd wordt naar het kassasysteem. Eens het signaal is aangekomen, wordt de GUI geüpdatet zodat op het einde van het printproces de verkoopster exact weet hoeveel pagina's er geprint zijn en hoeveel het te betalen totaal is.
Hieruit kunnen volgende onderzoeksvragen naar voor geschoven worden:

1. Is .NET MAUI voldoende ontwikkeld om het bestaande kassasysteem te vervangen?
2. Is het mogelijk om de desktop applicatie te vertalen naar een mobiel platform?

3. Is de ontwikkelde .NET MAUI-applicatie sneller in het verwerken van de printerklik dan de huidige applicatie.

Om na te gaan of .NET MAUI al voldoende ontwikkeld is voor deze toepassing zal er een proof-of-concept opgezet worden. In de eerste fase zal er uitgezocht worden of het analoge signaal, de zogenaamde klik, kan worden opgevangen door een Raspberry Pi en kan worden vertaald naar een digitaal signaal.
In het tweede luik van het onderzoek zal er een .NET MAUI-applicatie worden ontwikkeld die het huidige COBOL-kassasysteem moet vervangen. Deze applicatie moet zichzelf steeds kunnen updaten wanneer er een klik binnenkomt van de printer. Om dit te kunnen verwezenlijken zal er ook een API (Application Programming Interface) opgezet worden. Deze API moet het mogelijk maken om door de Raspberry Pi aangesproken te worden elke keer dat er een klik van een printer binnenkomt. Daarnaast moet de API ook de client kunnen updaten na elke klik. Deze feature zal dan ook geïmplementeerd worden met behulp van SignalR. Een .NET library die het toelaat om real-time applicatie op te zetten. SignalR maakt het mogelijk om op basis van websockets bidirectioneel te communiceren tussen de client en de server. Met andere woorden de client kan data opvragen aan de server indien nodig, maar de server kan ook zonder aanvraag van de client data doorgeven. De bidirectionele communicatie moet verhelpen dat de client steeds na een aantal seconden aan de server moet vragen of er al nieuwe kliks zijn binnengekomen.
In de laatste fase zal er worden nagegaan of de applicaties vertaald kan worden naar een Android-applicatie. Deze zou het mogelijk moeten maken om op het einde van de dag een pushmelding te sturen zodat de dagrapporten van de copycentra opgehaald en bekeken kunnen worden.
Om te weten of .NET MAUI voldoende ontwikkeld is om de huidige applicatie te vervangen, zal er een meting gebeuren. De meting in kwestie zal starten vanaf het moment dat de printer één klik verstuurt en stopt op het moment dat de grafische interface zichzelf update. Daarnaast zal er ook gekeken worden naar developer experience bij het ontwikkelen in .NET MAUI. Hier wordt er nagegaan of de ”intellisense”, een helper tool die suggesties geeft tijdens het programmeren, goed werkt. Met andere woorden; worden de juiste suggesties gegeven tijdens het programmeren. Ook zal er gekeken worden of er nog integratiebugs in Visual Studio zitten: krijg je foutmeldingen van Visual Studio die niets te maken hebben met de code, maar met de .NET MAUI integratie etc.

A.4. Verwacht resultaten

De metingen zouden moeten uitwijzen dat de .NET MAUI een grote stap vooruit is in vergelijking met de huidige kassa-applicatie geschreven in COBOL. De .NET applicatie zou sneller moeten reageren dan de huidige applicatie.
Daarnaast zou de developer experience al zodanig gevorderd zijn dat er slechts

kleine bugs overblijven in de integratie van .NET MAUI en Visual Studio. Het zou mogelijk moeten zijn om .NET MAUI te gebruiken om een productiewaardige applicatie te ontwikkelen.

A.5. Verwachte conclusie

.NET MAUI is een product dat nog steeds volop in ontwikkeling is. Dit kan er dan ook voor zorgen dat de developer experience nog niet optimaal is: de “intellisense” geeft geen of foute suggesties, Visual Studio geeft foutmeldingen die niets te maken hebben met de geschreven code, de applicatie crasht soms vanzelf etc. Als men toch wenst .NET MAUI reeds te gebruiken, zou dit echter wel mogelijk moeten zijn. Men moet enkel opletten bij het gebruik van bepaalde features en packages. Desondanks de kleine ongemakken zou .NET MAUI toch de toekomst kunnen zijn van cross-platform development waar kleinere development-teams, die een applicatie moeten ontwikkelen voor meerdere platformen, gretig gebruik van kunnen maken.

Bibliografie

Ab, S. (g.d.). *The current HTTP/2 adoption and the quite recent history*. https://www.shimmercat.com/blog/http2adoption/

Abby. (2023, februari). *COBOL Guide: History, Origin, and More*. https://history-computer.com/cobol-guide/

Ably. (2020, juli). *WebSockets - A Conceptual Deep Dive | Ably Realtime*. Ably. https://ably.com/topic/websockets

Brady Gaster. (2020, september). *Introduction to SignalR*. Microsoft. https://learn.microsoft.com/en-us/aspnet/signalr/overview/getting-started/introduction-to-signalr

Britch, D. (2023, januari). *What is .NET MAUI?* https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0

Britch, D., & Gechev, I. (2022, november 8). *What is .NET MAUI?* https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui?view=net-maui-7.0

Cloudflare. (g.d.). *What is HTTP?* Cloudflare. https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/

Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. *Internet Engineering Task Force*. https://doi.org/10.17487/rfc6455

Fulber-Garcia, V., & Fulber-Garcia, V. (2022, november). *HTTP: 1.0 vs. 1.1 vs 2.0 vs. 3.0*. https://www.baeldung.com/cs/http-versions

GeeksforGeeks. (2022, februari 21). *What is web socket and how it is different from the HTTP?* https://www.geeksforgeeks.org/what-is-web-socket-and-how-it-is-different-from-the-http/

Grigorik, I. (g.d.). *High Performance Browser Networking*. https://www.oreilly.com/library/view/high-performance-browser/9781449344757/ch09.html

Grigorik, I. (2016, september). *Introduction to HTTP/2*. https://web.dev/performance-http2

Kathiresan, S. G. (2022, november). *Xamarin Versus .NET MAUI*. https://www.syncfusion.com/blogs/post/xamarin-versus-net-maui.aspx

Koleva, P. (2023, februari). *From Xamarin to MAUI, What has changed?* https://flatrocktech.com/xamarin-forms-maui-migration/

Kotlin Foundation. (2022, september 6). *What is cross-platform mobile development?* Kotlin Foundation. https://kotlinlang.org/docs/cross-platform-mobile-development.html

Kotlin Foundation. (2023, maart 29). *Native and cross-platform app development: how to choose?* Kotlin Foundation. https://kotlinlang.org/docs/native-and-cross-platform.html

Lutkevich, B. (2022, mei 9). *real-time application (RTA)*. https://www.techtarget.com/searchunifiedcommunications/definition/real-time-application-RTA

Marchuk, A. (g.d.). *Native vs Cross-Platform Development: Pros and Cons Revealed*. https://www.uptech.team/blog/native-vs-cross-platform-app-development

MDN. (2022, november 22). *An overview of HTTP*. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

MDN. (2023a, maart). *An overview of HTTP*. MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

MDN. (2023b, maart). *Evolution of HTTP*. MDN. https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/Evolution_of_HTTP

Micro Focus. (2022, maart). *COBOL Market Shown to be Three Times Larger than Previously Estimated in New Independent Survey*. Micro Focus. https://www.microfocus.com/en-us/press-room/press-releases/2022/cobol-market-shown-to-be-three-times-larger-than-previously-estimated-in-new-independent-survey

National Museum of American History. (2013, september). *Proposing COBOL*. National Museum of American History. https://americanhistory.si.edu/cobol/proposing-cobol

Paessler. (g.d.). *What is HTTP?* Paessler. https://www.paessler.com/it-explained/http

Ramel, D. (2022, augustus). *What's Next for .NET MAUI? Roadmap & Xamarin Sunset Unveiled*. https://visualstudiomagazine.com/articles/2022/08/22/net-maui-roadmap.aspx

Reuters. (2017). *COBOL blues*. http://fingfx.thomsonreuters.com/gfx/rngs/USA-BANKS-COBOL/010040KH18J/index.html

Schmitt, J. (2022, augustus 24). *Native vs cross-platform mobile app development*. https://circleci.com/blog/native-vs-cross-platform-mobile-dev/?utm_source=google

Singh, S. (g.d.). *What is HTTP Long Polling ?* https://www.educative.io/answers/what-is-http-long-polling

UXDivers. (g.d.). *.NET MAUI vs Xamarin.Forms: A Comparison of Cross-Platform Frameworks*. https://grialkit.com/blog/learn-the-key-differences-between-net-maui-vs-xamarin-forms-for-cross-platform-mobile-and-desktop-development