ONDERZOEKSVOORSTEL

Een onderzoek naar de maturiteit van .NET MAUI bij het ontwikkelen van een real-time kassa-applicatie.

Bachelorproef, 2022-2023

Michiel Van Herreweghe

E-mail: michiel.vanherreweghe@student.hogent.be
Co-promotor: K. Van Moorter (Vandapower, kristof.vanmoorter@vandapower.com)

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.

Keuzerichting: Mobile & Enterprise development
Sleutelwoorden: Cross-platform development, .NET MAUI, real-time applicatie, Raspberry Pi

Inhoudsopgave

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.

2. State-of-the-art

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).

2.1.1. 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).

2.1.2. .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).

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.

2.2.1. 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 data-aanvragen 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).

2.2.2. 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).

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.

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.

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.

Referenties

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

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/

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

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