Afstudeerverslag

Modernisering van een .NET Windows Applicatie: Onderzoek naar het Opdelen en Vernieuwen van de Monolithische Front-end Architectuur

Lisa Molhoek
477548
06-19420953
Lisa.molhoek@gmail.com

Logic4
Chris van Dam
06-82920298
Chris@logic4.nl

Hogeschool Saxion
Etto Salomons
06-22490777
e.l.salomons@saxion.nl

1 Versie beheer

Versie	Datum	Opmerking
0.0	24-02-2023	Aanmaak document
0.1	04-06-2023	Concept verslag
1.0	18-06-2023	Definitief verslag

2 Voorwoord

U leest nu het afstudeerverslag getiteld “Modernisering van een .NET Windows Applicatie: Onderzoek naar het Opdelen en Vernieuwen van de Monolithische Front-end Architectuur”. Dit verslag is geschreven ter voltooiing van de afstudeerstage van de opleiding HBO-ICT in de richting software engineering aan de hogeschool Saxion. Ik heb van februari 2023 tot en met juli 2023 aan dit afstudeerproject gewerkt bij Logic4.

Ik zocht naar een uitdagend afstudeerproject dat gericht was op technieken die niet direct zichtbaar zijn. Dit project sluit daar perfect op aan. Daarnaast heb ik ervaring kunnen opdoen met technieken waar ik nog helemaal niet bekend mee was. Ik had zelfs nog nooit eerder met desktop applicaties gewerkt. Ook heb ik altijd al een grote interesse gehad in het proces matige aanpak van een project. Tijdens de studie is hierbij vooral aandacht besteed richting teams. Het was zeer interessant om een eigen draai te geven aan een techniek die normaal van toepassing is op teams. Dit project heeft mij hierdoor niet alleen op een vakkundig maar ook op een persoonlijk vlak bevorderd.

Ik wil in het bijzonder Chris van Dam en David La Roi bedanken voor de uitstekende begeleiding en steun tijdens dit project. Jullie hebben mij goed kunnen bijstaan, zowel tijdens de lichte als zware momenten. Daarnaast hebben jullie mij uitgedaagd om mijn onderzoek nog beter te maken. Ook wil ik Etto Salomons bedanken voor zijn waardevolle adviezen gedurende dit project. Uiteraard wil ik ook alle collega's bij Logic4 bedanken voor hun hulp en de plezierige tijd die we samen hebben gehad, waarbij de tosti’s niet vergeten moeten worden.

Tot slot wil ik mijn familie en vrienden bedanken voor hun hulp en luisterend oor gedurende mijn stageperiode.

Ik wens u veel leesplezier.

Lisa Molhoek

Enschede, 18-06-2023

3 Samenvatting

Logic4 heeft een verouderde .NET Framework 4.7.2 front-end applicatie van aanzienlijke omvang. Het is niet haalbaar om deze applicatie over te zetten naar een nieuwere versie van .NET. Om de onderhoudbaarheid van het project te verbeteren, heeft Logic4 besloten om een nieuwe applicatie met een recentere .NET-versie naast de oude applicatie te laten draaien. Dit heeft geleid tot de hoofdvraag: "Wat is voor de front end van het ERP pakket van Logic4 een manier om een aparte applicatie te maken van een scherm dat kan worden gestart vanuit de bestaande ERP applicatie en met de bestaande applicatie kan communiceren?".

Voor dit onderzoek is gebruikgemaakt van beschrijvend onderzoek bij het analyseren van de huidige applicatie. Bij het onderzoeken van mogelijke en beste technieken is bureauonderzoek gedaan en zijn verschillende technieken getest. De resultaten hiervan hebben geleid tot een ontwerp waaruit een Proof of Concept (PoC) is ontwikkeld. Op basis van de PoC zijn conclusies getrokken en adviezen gegeven aan het bedrijf.

Uit het onderzoek is gebleken dat het gebruik van anonymous pipes in combinatie met Protocol Buffers de beste keuze is voor de communicatie tussen de twee .NET-applicaties van Logic4. Deze keuze is gemaakt vanwege de snelheid en veiligheid van deze technieken, hoewel ze complexer zijn. Anonymous pipes is een vorm van Inter-Process Communication die alleen kan worden gebruikt tussen parent-child processen en tussen threads binnen dezelfde applicatie. Een anonymous pipe-verbinding bestaat uit een server en een client, waarbij de client verbinding kan maken met de server door de pipehandle te verkrijgen. De bestaande applicatie moet de nieuwe applicatie starten met behulp van `Process.Start`, omdat zo argumenten kunnen worden doorgegeven aan de opgestarte applicatie. Een van deze argumenten moet de pipehandle zijn om de verbinding tot stand te brengen. Protocol Buffers is een mechanisme voor de serialisatie van gestructureerde gegevens.

Het is gelukt om een .NET 7 applicatie naast de bestaande applicatie te draaien, waarbij de nieuwe applicatie gestart wordt vanuit de bestaande applicatie en kan communiceren met de bestaande applicatie. Het is mogelijk om een scherm uit een andere applicatie te openen en vervolgens te communiceren met dat scherm alsof het een scherm binnen de bestaande applicatie is. Wanneer er gegevens worden gewijzigd in de bestaande applicatie, worden deze wijzigingen doorgegeven aan de nieuwe applicatie, zodat beide applicaties altijd up-to-date zijn. Op dit moment is het echter merkbaar dat het twee afzonderlijke applicaties zijn, aangezien ze beide een eigen plek hebben in de taakbalk. Ook klopt de zichtbaarheid van de schermen niet altijd. Het uitschakelen van de gebruikersinteractiemogelijkheid van een scherm is nog niet bewezen. Als de bestaande applicatie zou worden gesloten, zal dit niet automatisch de nieuwe applicatie sluiten. Bovendien is het op dit moment nog niet mogelijk om gebruikersgegevens over te brengen, wat betekent dat gebruikers opnieuw zou moeten inloggen om calls te maken naar de API. Ook kunnen ontwikkelaars nog enkele uitdagingen ervaren. De nieuwe applicatie moet telkens opnieuw worden gebouwd om wijzigingen te kunnen testen, en de debugger moet handmatig aan de applicatie worden toegevoegd.

Mijn advies aan Logic4 is om verder onderzoek te doen naar de genoemde problemen. Daarnaast raad ik aan om te overwegen om van de nieuwe applicatie geen enkelvoudig scherm te maken, maar een applicatie met meerdere schermen. Hierdoor kunnen grotere delen van de applicatie worden gemigreerd naar een nieuwere versie van .NET. Ik wil ook aanraden om verder te kijken dan alleen .NET. De nieuwe applicatie kan ook worden ontwikkeld met behulp van een ander framework.

Door deze aanbevelingen te onderzoeken, kan Logic4 de beperkingen en uitdagingen die momenteel worden ervaren, verminderen. Het zal hen helpen om het project te realiseren in front end ERP applicatie. Daarbij zouden ze de onderhoudbaarheid van het project verbeteren en een solide basis te leggen voor verdere ontwikkelingen.

4 Inhoudsopgave

5 Figuren, tabellen en code

Figuren:

Tabellen:

Code voorbeelden:

6 Begrippenlijst

Begrip	Definitie
.NET	Een software framework voor het ontwikkelen van Windows, Linux en MacOS applicaties. Het is de opvolger van .NET Framework en voor het eerst uitgebracht in 2016.
.NET Framework	Een software framework voor het ontwikkelen van Windows applicaties. Het is voor het eerst uitgebracht in 2002.
.NET remoting	Een mechanisme waarmee objecten op afstand kunnen worden gemaakt en gebruikt door een applicatie.
API	Application Programming Interface, een mechanisme dat er voor zorgt dat twee software componenten met elkaar kunnen communiceren doormiddel van definities en protocollen.
Class coupling	Een meting van de mate waarin een enkele software component afhankelijk is van andere software componenten.
ERP	Enterprise Resource Planning, het management van core business processen door middel van software.
Framework	Een set van tools en functies die ontwikkelaars kunnen gebruiken om applicaties te bouwen en te draaien.
Front-end	Het gedeelte van de applicatie dat een gebruiker kan zien en interacteren. Dit kan communiceren met de backend waar de data vandaan komt.
gRPC	Een framework voor het maken van gedistribueerde applicaties waarbij gebruik wordt gemaakt van Remote Procedure Calls.
IPC	Inter-Process communication, een mechanisme waarmee verschillende processen op een systeem met elkaar kunnen communiceren.
Library (software)	Een verzameling van vooraf geschreven code die ontwikkelaars kunnen gebruiken om bepaalde taken uit te voeren.
Lineaire onafhankelijke paden	Verschillende routes of paden in een programma die geen onderlinge afhankelijkheid hebben.
Microservices backend	Een software structuur dat bestaat uit los verbonden services.
Mock-up	Een prototype of een representatie van een applicatie, meestal op visueel niveau, zonder dat alle functionaliteiten volledig geïmplementeerd zijn.
Monoliet systeem	Applicatie die als een enkele en ondeelbare eenheid is gebouwd.
NIST	National Institute of Standards and Technology, een agentschap van het Amerikaanse ministerie van handel met als missie het bevorderen van Amerikaanse innovatie en industrieel concurrentievermogen.
Onderzoeksrapport	Een verslag waarin het onderzoek is vastgelegd, inclusief de opzet en uitvoering van het onderzoek, de resultaten, conclusies en aanbevelingen.
PoC	Proof of Concept. Een realisatie van een methode of idee dat laat zien hoe haalbaar het uitgewerkte idee is om te maken.
Scherm	Een specifieke weergave of pagina in een applicatie dat aan de gebruiker wordt getoond. Het is een visuele en functionele eenheid waarin bepaalde informatie of functionaliteit wordt gepresenteerd.

Scrum	Een agile project management framework dat veel wordt gebruikt bij software ontwikkeling
Snyk	Een tool die wordt gebruikt voor het scannen van beveiligingsrisico's in de code van softwareprojecten. Het analyseert de code en detecteert mogelijke kwetsbaarheden, bekende beveiligingslekken, verouderde library's of configuratiefouten die kunnen leiden tot potentiële beveiligingsproblemen.
WCF	Windows Communication Foundation, een framework voor het bouwen van gedistribueerde applicaties in .NET.

7 Inleiding

Logic4, een middelgroot softwarebedrijf dat ERP-planning en e-commerce ecosystemen aanbiedt voor verschillende non-foodsectoren binnen de groothandel en retail, staat voor een uitdaging. Ze hebben een grote, monolithische front-end applicatie die is gebouwd met behulp van .NET Framework 4.7.2. Vanwege de omvang van de applicatie is het niet mogelijk om deze in één keer over te zetten naar een nieuwere versie van .NET. Daarom willen ze graag onderzocht hebben wat de mogelijkheden zijn rondom het draaien van een nieuwe .NET applicatie naast de bestaande applicatie, zonder dat de gebruiker dit merkt. Hoewel het niet realistisch is om dit binnen de afstudeerperiode te realiseren, zal er wel een Proof of Concept worden ontwikkeld om de haalbaarheid van dit concept aan te tonen.

Het doel van dit afstudeeronderzoek is om de mogelijkheden te onderzoeken van het draaien van een nieuwe .NET-applicatie naast de bestaande monolithische applicatie van Logic4, met als uiteindelijk doel een geleidelijke overgang naar .NET 7 mogelijk te maken. Een dergelijke overgang zou de onderhoudsmogelijkheden van het project verbeteren en Logic4 in staat stellen gebruik te maken van de nieuwe functionaliteiten van .NET 7.

Het ERP-pakket van Logic4 bestaat uit ongeveer 1.000.000 regels code en 750 schermen, waardoor een directe migratie naar een nieuwere versie van .NET niet haalbaar is. Hoewel er online geen vergelijkbare projecten te vinden zijn, lijkt het concept enigszins op microfrontends. Het overnemen van deze technieken is niet mogelijk, omdat deze gericht is op web toepassingen. De communicatie tussen web front-ends werkt anders dan die van Windows applicaties.

Dit onderzoek maakt gebruik van beschrijvend onderzoek en bureauonderzoek om de mogelijkheden te verkennen. Daarnaast zijn er enkele tests uitgevoerd om de juiste technieken te selecteren voor het ontwikkelen van PoC. Microsoft biedt technieken voor het starten van een applicatie vanuit een .NET applicatie en het communiceren tussen twee processen, wat relevant is voor dit onderzoek waarbij gekeken wordt naar de mogelijkheden van het .NET Framework 4.7.2 en .NET 7.

Het volgende hoofdstuk van dit afstudeerverslag zal verdere verduidelijking geven over de opdracht. Hoofdstuk 9 zal inzicht bieden in de procesmatige aanpak dit is toegepast bij deze opdracht. In Hoofdstuk 10 zal het onderzoek van het huidige systeem en de verschillende communicatie technieken presenteren. Vervolgens zal Hoofdstuk 11 het ontwerp dat zal worden gebruikt voor het PoC omschrijven. Hoofdstuk 12 zal dan in gaan op de uitvoering van het PoC samen met de bijzondere bevindingen van het PoC. In Hoofdstuk 13 wordt de conclusie gegeven waarna in hoofdstuk 14 de adviezen volgen. Tot slot worden in Hoofdstuk 15 de bronnen getoond en in Hoofdstuk 16 de bijlagen aangeboden.

7.1 Organisatie

Softwareblocks BV opereert onder de naam Logic4 en biedt hun ERP-pakket en e-commerce ecosysteem aan als één geïntegreerd softwarepakket.

Het bedrijf werd opgericht op 23 december 2003 door Danny Nijland, die al sinds 1998 de basis legde. Hij was een van de pioniers van webshops in Nederland en ontwikkelde tevens zijn eigen bedrijfssoftware. Sinds 1 april 2020 is Daniel Ruff de nieuwe CEO van Logic4, en sinds 2016 is Chris van Dam betrokken als CTO (Chief Technology Officer).

Logic4 heeft momenteel tussen de 30 en 35 medewerkers, waarvan ongeveer 10 werkzaam zijn binnen de IT-afdeling. Het personeel werkt zowel vanuit het kantoor in Enschede als vanuit huis. Teams en Whatsapp worden gebruikt voor interne communicatie binnen het bedrijf.

Het bedrijf streeft naar voortdurende verbetering en is van mening dat er altijd ruimte is voor verbetering. Dit omvat ook de voortdurende ontwikkeling van hun software.

De locatie van Logic4 is Nijverheidsstraat 3 in Enschede, een centrale plek tussen het stadscentrum en Saxion. Het kantoor is gemakkelijk bereikbaar per fiets en openbaar vervoer.

8 Opdracht

In dit hoofdstuk wordt de opdracht toegelicht en worden de eindproducten beschreven.

8.1 Probleemstelling

Logic4 heeft een front-end applicatie ontwikkeld voor hun ERP-pakket, bestaande uit één groot monoliet met ongeveer 1.000.000 regels code en 750 schermen. Een scherm is een visuele en functionele eenheid waarin bepaalde informatie of functionaliteit wordt gepresenteerd.

De omvang van de applicatie leidt tot problemen die inherent zijn aan monolithische architecturen, zoals een afnemende begrijpelijkheid van de code en langere doorlooptijden voor het implementeren van nieuwe features. Bovendien maakt het updaten of verwijderen van verouderde libraries of frameworks de applicatie gevoelig voor beveiligingsrisico's en verminderde functionaliteit.

Als dit probleem niet wordt aangepakt, zal de applicatie blijven groeien en kunnen de ontwikkelingskosten voor Logic4 toenemen, terwijl de klanten langer moeten wachten op oplossingen voor hun problemen. Het doel van dit onderzoek is daarom om oplossingen te onderzoeken voor het opdelen van de monolithische front end architectuur van de Logic4 ERP-applicatie en het verminderen van de nadelen die gepaard gaan met deze architectuur.

8.2 Opdracht

Er moet onderzoek worden gedaan naar een manier waarop een scherm in een aparte applicatie kan worden gestart vanuit de bestaande applicatie. De bestaande applicatie is de ERP applicatie die gebouwd is met Windows Forms en .NET Framework 4.7.2.

Er moet in samenwerking met Logic4 een scherm gekozen worden die tot een eigen applicatie gemaakt zal moeten worden. Dit scherm zal moeten worden onderzocht, met name de communicatie die het scherm heeft met de rest van de applicatie.

Er zullen meerdere manieren voor het starten van een applicatie vanuit de bestaande applicatie moeten worden onderzocht samen met de mogelijke manieren van communicatie tussen de nieuwe en bestaande applicatie hieruit moet worden bepaald wat de beste keuze is voor het ERP pakken van Logic4.

De beste manier moet uitgewerkt worden in een PoC (Proof of Concept). Dit moet een PoC worden waarbij vanuit de bestaande applicatie één scherm wordt opgestart, die wordt gehost in een ander uitvoerbaar bestand. Het is hierbij belangrijk dat de communicatie tussen de monoliet en de nieuwe applicatie werkt.

Na het maken van de PoC moet er worden gekeken naar de gevolgen van deze PoC. Denk hierbij aan de struikelblokken of mogelijke gevolgen van het PoC. Ook zal er terug gereflecteerd moeten worden of de gekozen methode nog steeds het best is.

Als deze opdracht succesvol wordt afgerond dan biedt dit Logic4 inzicht in de mogelijkheid om delen van de bestaande EPR applicatie om te bouwen tot een eigen applicatie. Verwachting is dat het dan ook makkelijker wordt om aanpassingen te maken in deze delen.

8.3 Eisen en wensen implementatie opdracht

Logic4 heeft een aantal eisen en wensen bij de mogelijkheden wanneer ze het PoC gaan implementeren bij de bestaande applicatie. Voor dit project moeten er zo veel mogelijk eisen en wensen worden bewezen.

Eisen:

- De gebruiker hoeft niet opnieuw in te loggen.

- De gebruiker moet dezelfde gebruikers-ID en rechten hebben als in de bestaande applicatie.
- De gebruiker moet een scherm kunnen openen vanuit de bestaande applicatie en argumenten kunnen meegeven.
- De gebruiker moet een scherm kunnen openen vanuit de *nieuwe* applicatie en argumenten kunnen meegeven.
- Wanneer een scherm van de nieuwe applicatie wordt gesloten, moet er informatie teruggegeven kunnen worden aan de bestaande applicatie.
- Wanneer een scherm van de *bestaande* applicatie wordt gesloten, moet er informatie teruggegeven kunnen worden aan de nieuwe applicatie.

Wensen:

- Bij het scherm van de bestaande applicatie moet het tijdelijk niet mogelijk zijn om gebruikersinteractie te kunnen uitvoeren tijdens modale dialogen.
- De gebruiker moet beide samenwerkende applicaties als een geheel ervaren.

8.3.1 Eindproducten

Er wordt een ontwerp gemaakt van de relevante onderdelen van de oorspronkelijke applicatie die nodig zijn voor dit project. Er wordt een onderzoeksrapport opgesteld waarin de bevindingen en gemaakte keuzes worden toegelicht. Op basis van dit rapport wordt een ontwerp voor een PoC gemaakt. Dormiddel van dit ontwerp wordt een PoC ontwikkeld. Tevens wordt het PoC geleverd met bijbehorend advies. In het afstudeerverslag worden al deze aspecten uitgewerkt, zodat Logic4 het idee van dit project kan implementeren in hun applicatie.

9 Procesmatige aanpak

Hier zal het proces binnen het project worden besproken. Binnen dit project is er gebruik gemaakt van een globe wekelijkse planning en afspraken. Daarnaast is er een zelf ontwikkeld projectmanagementkader gebruikt, dat sterk is geïnspireerd door scrum. Hoewel scrum bedoeld is voor teams, was dit project een individuele inspanning, waardoor een strikte toepassing van scrum niet geschikt was. Voor de reflectie hierop zie Hoofdstuk 15 Reflectie.

9.1 Globale planning en afspraken

Voor dit project is een globale wekelijkse planning opgesteld. Hierin zijn de verschillende onderzoeksfases, de planning van de documenten, de zitting, de deadlines en andere belangrijke momenten vastgelegd. Deze planning is gebruikt om de voortgang van het project bij te houden. Indien het project niet op schema lag, kon dit besproken worden met de belanghebbenden aan de hand van deze planning. Voor deze planning zie “Bijlage K. Wekelijkse planning”.

De afspraken en opleveringsdata zijn in twee overzichtelijke schema's weergegeven, zodat ze gemakkelijk kunnen worden geraadpleegd. Zie de tabellen hieronder.

Tabel 1 Afspraken opdracht

Afspraak	Moment
Stand-up met David	Elke werkdag
Wekelijkse meeting Chris en David	Elke laatste dag van de werkweek
Maandelijkse meeting afstudeerbegeleider	Elke maand
Evaluatiemoment Plan van Aanpak	Week 4
Zitting	Tussen 3 en 14 juli

Tabel 2 Opeleveringsdata

Oplevering	Datum
Concept Plan van Aanpak	12-03-2023
Definitief Plan van Aanpak	26–03-2023
Maandelijkse oplevering 1	12-03-2023
Maandelijkse oplevering 2	09-04-2023
Maandelijkse oplevering 3	07-05-2023
Concept afstudeerverslag	04-06-2023
Definitief afstudeerverslag	18-06-2023
Evaluatieformulier bedrijfsbegeleider	02-07-2023

9.2 Project management kader

Voor het projectmanagement is gebruikgemaakt van sprints met een duur van één week. Dit betekent dat er elke week een nieuwe set taken moesten worden klaargezet. Door deze wekelijkse planning konden er snel aanpassingen worden gedaan, waardoor zowel de bedrijfswensen als de afstudeervereisten goed konden worden afgestemd op de behoeften. Bovendien waren de taken klein genoeg om meerdere taken in dezelfde week af te ronden.

De sprintplanning werd besproken tijdens de wekelijkse vergadering, die plaatsvond op de laatste werkdag van de week. De bedrijfsbegeleider en een collega waren hierbij aanwezig. Tijdens deze vergadering werd ook de voortgang besproken en werden belangrijke beslissingen genomen. Deze gesprekken hadden een grote invloed op het project. Zo werd bijvoorbeeld gezamenlijk besloten om een mock-up van de applicatie te maken, omdat het niet haalbaar was om dit direct in de bestaande applicatie te implementeren.

De retrospectives, die binnen scrum worden uitgevoerd om als team terug te blikken op de afgelopen sprint, vonden individueel plaats. Reflectiedocumenten werden bijgehouden waarin de STARR-methode werd toegepast op de afgelopen week. Daarnaast werd gekeken naar het doel van de komende week, zodat dit met de bedrijfsbegeleider kon worden besproken. Ook dit werd wekelijks gedaan om snel aanpassingen te kunnen maken.

Er is gebruikgemaakt van een backlog, waarin de taken voor het project werden bijgehouden. De backlog werd regelmatig aangevuld, omdat nieuwe taken voortkwamen uit de resultaten van andere taken. In eerste instantie was het plan om de taken te sorteren met behulp van MoSCoW. Dit werd echter al snel verlaten, omdat mondeling werd besproken welke taken het belangrijkst waren en de sprint hierop werd aangepast. Aangezien slechts één persoon verantwoordelijk was voor de taken, was het niet nodig om dit volledig op papier vast te leggen.

10 Onderzoek

Dit hoofdstuk behandelt de opzet en resultaten van het onderzoek.

10.1 Hoofdvraag en deelvragen

De hoofdvraag dient als leidraad voor het beantwoorden van het onderzoek. De deelvragen worden gebruikt om de hoofdvraag te beantwoorden.

10.1.1 Hoofdvraag

Wat is voor de front end van het ERP pakket van Logic4 een manier om een aparte applicatie te maken van een scherm dat kan worden gestart vanuit de bestaande ERP applicatie en met de bestaande applicatie kan communiceren?

10.1.2 Deelvragen

Hoe zien de onderdelen en principes van de ERP applicatie van Logic4 die relevant zijn voor deze opdracht eruit?
Doel: Om een scherm uit de bestaande applicatie te kunnen halen zal er eerst gedaan worden naar de huidige situatie.
Aanpak: Bekijken van het systeem, bekijken documentatie, overleg met collega's en bureauonderzoek.
Beroepsproducten: Een onderzoeksrapport met bevindingen, bijvoorbeeld classdiagrammen.
Subvragen:

- Welk scherm wordt uit de huidige applicatie gehaald en hoe ziet deze eruit?
- Met welke onderdelen van de huidige applicatie heeft het scherm te maken en hoe ziet dat eruit?
- Welke communicatie heeft het scherm dat eruit gehaald wordt met de huidige applicatie?

Welke technieken zijn er voor het maken van een aparte applicatie voor een front end?
Doel: Oriënteren binnen de huidige technieken voor het opdelen van de front end, zodat één van de technieken later kan worden uitgewerkt tot een PoC.
Aanpak: Bureauonderzoek en communicatie met experts binnen en/of buiten het bedrijf.
Beroepsproduct: Onderdeel van het onderzoeksrapport.

Hoe moet de aparte applicatie van het gekozen scherm met de best toepasselijke techniek eruitzien?
Doel: Het kiezen van de beste techniek en het maken van een ontwerp voor het PoC, zodat de PoC gemaakt kan worden.
Aanpak: Bureauonderzoek en communicatie met experts binnen en/of buiten het bedrijf.
Beroepsproduct: Functioneel en technisch ontwerp van de aparte applicatie en hoe deze gaat communiceren met de bestaande applicatie.

Wat zijn de gevolgen van het maken van een aparte applicatie van het gekozen scherm met de best toepasselijke techniek uit de bestaande applicatie?
Doel: Advies kunnen geven aan het bedrijf zodat zij keuzes kan maken voor de applicatie.
Aanpak: Het maken van een PoC, een analyse van het PoC en experts binnen en/of buiten het bedrijf raadplegen.
Beroepsproducten: Een PoC met daarbij de benodigde documentatie en een adviesrapport.
Subvragen:

- Wat waren de struikelblokken en of mogelijke verbeteringen van het PoC?

- Denkt de stagiair dat de gekozen techniek nog steeds de beste manier is om een aparte applicatie te maken voor het gekozen scherm.
- Wat zijn de gevolgen voor de bestaande applicatie?

10.2 Onderzoeksmethoden

Er zijn verschillende onderzoeksmethoden gebruikt per deelvraag.

Hoe zien de onderdelen en principes van de ERP applicatie van Logic4 die relevant zijn voor deze opdracht eruit?

Voor deze deelvraag is veel gebruik gemaakt van beschrijvend onderzoek, een onderzoeksmethode om de huidige situatie in kaart te brengen. Het scherm dat onderzocht moest worden is bepaald door logic4. De stagiair heeft onderzocht hoe dit scherm en de schermen die hiermee te maken hebben eruit zien. De relevante onderdelen van de ERP-applicatie zijn in kaart gebracht door te klikken op die delen van de applicatie. Vervolgens zijn de functionaliteiten genoteerd. Als laatste zijn er klassendiagrammen en een flowdiagram gemaakt.

Er is nauwelijks gebruik gemaakt van de documentatie die door Logic4 is geschreven, omdat hier weinig relevante informatie in kon worden gevonden. Er is echter veel gebruik gemaakt van de kennis van collega's. Alle bevindingen zijn nagekeken door iemand die bekend is met de huidige applicatie.

Alles wat hiervoor is onderzocht, is verwerkt in een apart onderzoeksdocument.

Welke technieken zijn er voor het maken van een aparte applicatie voor een front end?

Voor deze deelvraag is er gestart met bureauonderzoek. Hierbij is online gezocht naar vergelijkbare problemen en verschillende benaderingen voor het ontwikkelen van de afzonderlijke onderdelen van het PoC. Vaak zijn bronnen van Microsoft geraadpleegd voor informatie. Ook is ChatGPT gebruikt, hoewel deze AI niet als bron is gebruikt. Het is ingezet om aanvullende zoektermen te genereren, waar vervolgens onderzoek naar gedaan is.

Daarnaast is er contact opgenomen met iemand die eerdere ervaring heeft met het opsplitten van een front-end applicatie van een website, omdat dit aanvankelijk overeenkomsten leek te hebben. Er is gevraagd hoe hij het probleem van communicatie had opgelost. Hieruit bleek dat hij een bestaand framework had gebruikt. Later in mijn onderzoek bleek echter dat dit significant anders is bij een Windows applicatie.

Hoe moet de aparte applicatie van het gekozen scherm met de best toepasselijke techniek eruitzien?

Om deze deelvraag te beantwoorden, zijn er tests uitgevoerd voor verschillende communicatiemethoden. Hiervoor is een kleine testomgeving gecreëerd en zijn er tests uitgevoerd om de duur van de communicatie te meten. Daarnaast is er gekeken naar de complexiteit van de code. Ook is er onderzocht hoeveel zoekresultaten er waren voor de verschillende methoden door ernaar te zoeken op het internet.

Bovendien zijn de ontwerpen besproken met een collega die kennis heeft van het onderwerp. Hierbij is ook gelet op de overeenstemming van het ontwerp met de denkwijze van Logic4, zodat zij er later goed gebruik van kunnen maken.

Wat zijn de gevolgen van het maken van een aparte applicatie van het gekozen scherm met de best toepasselijke techniek uit de bestaande applicatie?

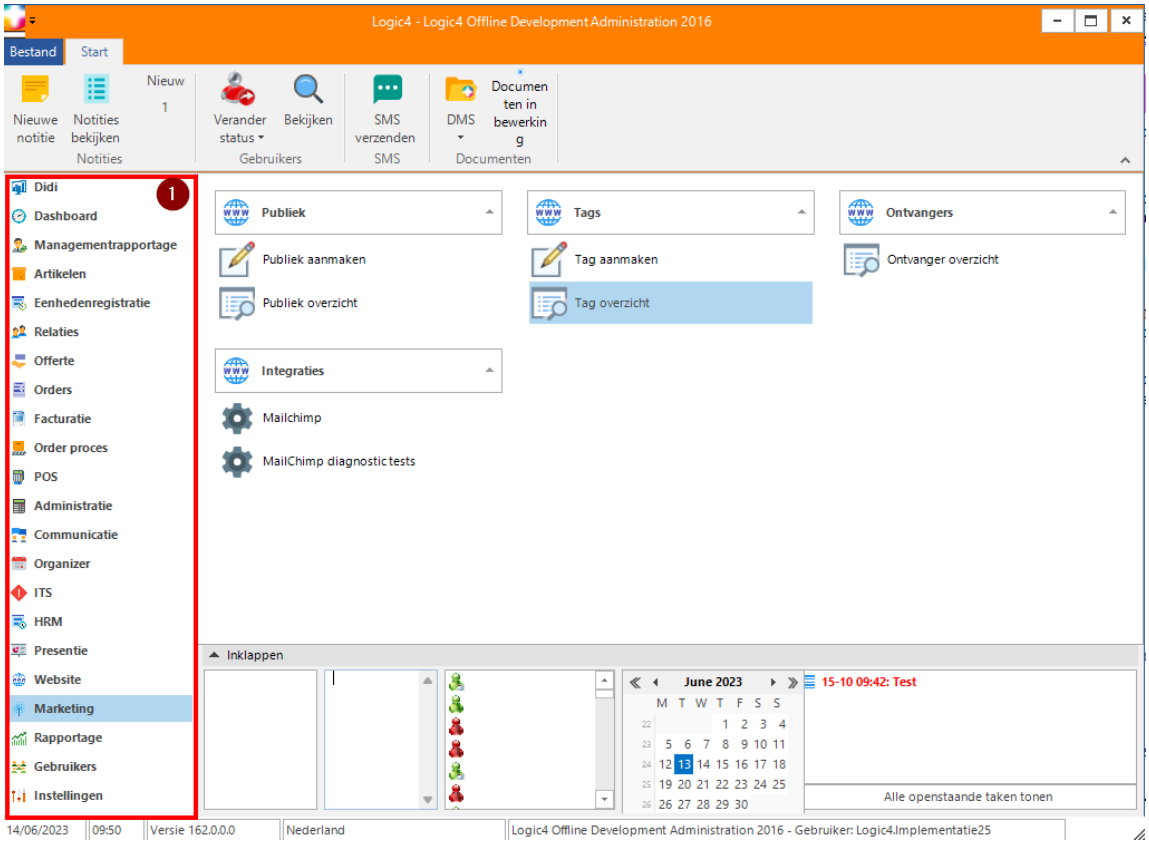
Er is een mock-up gemaakt van de huidige applicatie. Vervolgens is de ontworpen techniek toegepast zodat het minimale kon worden bewezen. Hierna werd het PoC verder uitgebreid om verschillende functionaliteiten die de huidige applicatie heeft te testen.

De resultaten van deze testen zijn besproken met het bedrijf. Hierna zijn er adviezen gegeven over waar vervolg onderzoek handig is en welke ideeën nog meer mogelijk zouden kunnen zijn.

10.3 Onderdelen ERP Applicatie

Hier wordt de onderzoeksvraag “Hoe zien de onderdelen en principes van de ERP applicatie van Logic4 die relevant zijn voor deze opdracht eruit?” beantwoord.

De ERP applicatie is een Windows Forms applicatie dat gebruik maakt van .NET Framework 4.7.2.. Het is een uitgebreide applicatie die bestaat uit verschillende tabbladen. De verschillende tabbladen worden weergegeven in het rode vak met het nummer 1 in de onderstaande afbeelding.

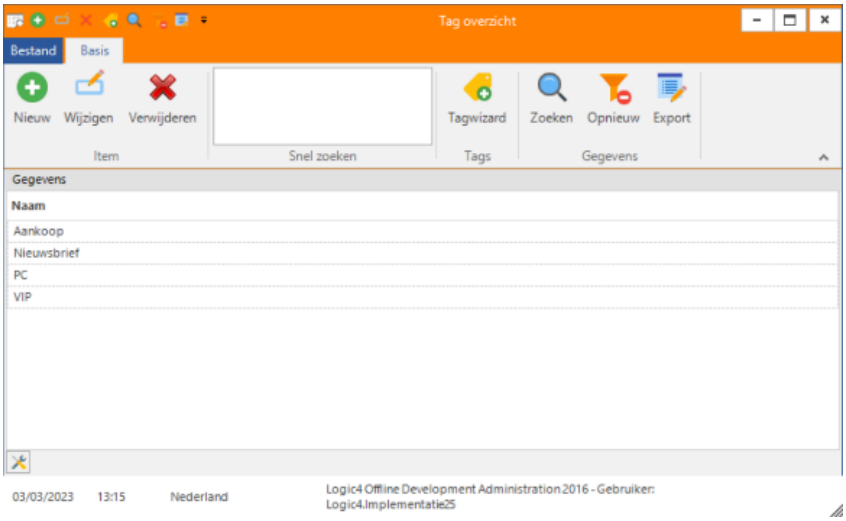


Figuur 1 Marketingtabblad met verwijzing naar alle tabbladen.

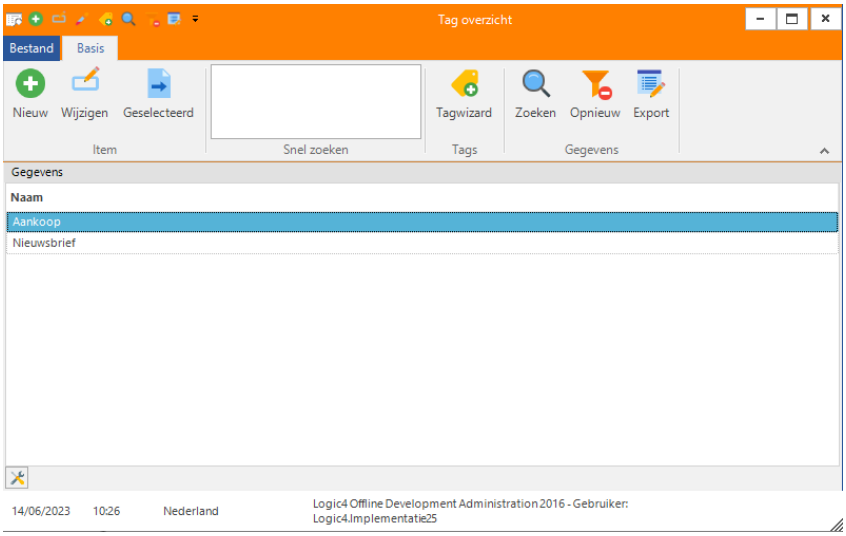
Het tabblad waar aan wordt gewerkt bij dit project is het marketingtabblad. Dit tabblad communiceert met zijn eigen microservice backend.Vanuit dit tabblad kunnen verschillende schermen worden geopend, waaronder het scherm voor het aanmaken en overzicht van het publiek, het scherm voor het aanmaken en overzicht van tags, het ontvangeroverzichtscherm, het MailChimp en MailChimp diagnosescherm.

10.3.1 Het tag overzicht scherm.

Logic4 heeft het tag overzicht scherm gekozen omdat dit scherm moet communiceren met andere schermen. Dit zorgt ervoor dat de communicatie tussen de bestaande en de nieuwe applicatie goed bewezen kan worden. Dit scherm kan worden geopend via het tagoverzicht knop op het marketingtabblad en via het scherm voor het aanmaken van het publiek. Het tagoverzichtscherm ziet er anders uit, afhankelijk van waar het wordt geopend. Dit komt doordat bij het scherm voor het aanmaken van het publiek tags geselecteerd kunnen worden. Zie de afbeeldingen (figuur 2 en 3) op de volgende pagina voor de verschillende versies.



Figuur 3 Tag overzicht scherm geopend vanuit tag overzicht



Figuur 2 Tag overzicht scherm geopend vanuit publiek aanmaken

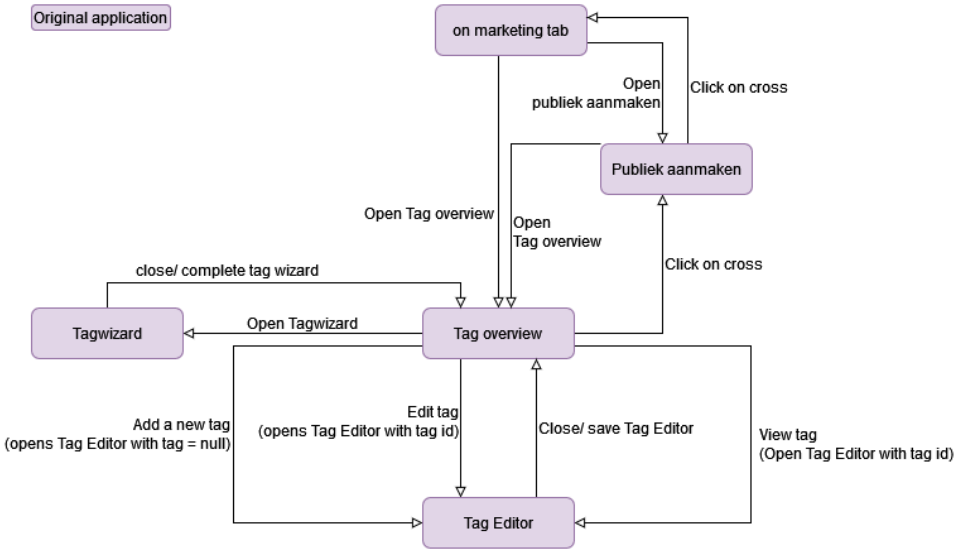
Het tag overzicht scherm stelt de gebruiker in staat om tags te bekijken, toe te voegen, aan te passen en te verwijderen. Een tag kan ook aangemaakt worden met de hulp van een Tagwizard. Daarnaast is het mogelijk om te zoeken naar bepaalde tags. Deze tags kunnen naar een Excel bestand worden geëxporteerd. Voor de class diagram van dit scherm zie “Bijlage A. Class diagram MarketingTagSearcher”. Voor een uitgebreide uitleg van de functies zij “Bijlage B. Overzicht scherm knoppen”.

De gebruiker kan kiezen hoeveel tags die voor zich wil zien op het scherm. Ook kunnen er verschillende scherminstellingen worden gedaan, zoals het herstellen van de schermposities.

Voor het openen van het tag overzicht scherm is er informatie nodig vanuit de front end applicatie. Hierbij worden gebruikers instellingen opgehaald die het juiste icoontje, grootte en plaats ophalen uit de database. Daarnaast wordt via een nieuwe functie opgehaald of het scherm in selectiemodus staat. Dit moet alleen het geval zijn wanneer het scherm wordt gestart vanuit het scherm voor het aanmaken van het publiek. Ten slotte worden de tags en de taginformatie opgehaald via de API.

10.3.2 Communicatie met schermen vanuit het tag overzicht scherm

Het tag overzicht communiceert met meerdere schermen. Zie de onderstaande afbeelding voor een overzicht van deze schermen.



Figuur 4 Tag overzicht diagram andere schermen

Voor het aanmaken, aanpassen en bekijken van een tag worden nieuwe schermen geopend. Deze schermen worden aangemaakt met behulp van dezelfde klasse. Wanneer dit scherm wordt geopend voor het aanmaken van een nieuwe tag, hoeft er geen tagId aan deze klasse te worden doorgegeven. Maar wanneer het scherm wordt geopend voor het aanpassen of bekijken van een tag, moet er wel een tagId worden meegegeven. Een tag kan ook worden aangemaakt via de Tagwizard, maar dit gebeurt via een aparte klasse. Wanneer een tag wordt verwijderd, verschijnt er een pop-upvenster met de vraag of de tag echt verwijderd moet worden. Dit is geen volledig scherm. Voor het class diagram van deze schermen zie “Bijlage C. Classdiagram schermen”.

10.4 Technieken voor een aparte front end applicatie

Hier wordt de onderzoeksvraag “Welke technieken zijn er voor het maken van een aparte applicatie voor een front end?” en gedeeltelijk de onderzoeksvraag “Hoe moet de aparte applicatie van het gekozen scherm met de best toepasselijke techniek eruitzien?” beantwoord.

Eerst zal er worden gekeken naar het starten van een applicatie. Vervolgens zal er worden gekeken naar de verschillende manieren van communicatie tussen de twee front end applicaties. Bij de communicatie tussen de applicaties zullen er een aantal testen worden uitgevoerd waaruit zal blijken welke manier

10.4.1 Starten applicatie

De nieuwe applicatie zal gestart moeten worden vanuit de bestaande applicatie. Dit is zodat de gebruiker maar één applicatie hoeft op te starten. De gebruiker moet niet merken dat het twee verschillende applicaties zijn.

Er zijn twee mogelijkheden voor het starten van de applicatie.

1. `Interaction.Shell` is een simpele functie waar weinig extra aspecten aan zitten [1]. Deze functie kan een applicatie starten door aan de functie de locatie van de applicatie mee te geven. Daarnaast kan de AppWinStyle [2], een boolean om aan te geven of het programma moet wachten en het aantal seconden

dat het programma moet wachten worden meegeven. De functie geeft een int terug met het gestarte programma's process ID.

2. `Process.Start` is een complexere functie omdat hier veel mogelijkheden bij zitten [3]. Dit kan een handig zijn in de toekomst wanneer er andere applicaties gestart moeten worden. Een interessant voorbeeld van deze mogelijkheden is arguments. Er kunnen doormiddel van een string argumenten worden meegeven. Dit kan via de parameters maar ook via een class "ProcessStartInfo" [4]. Via die class kunnen er nog meer specificaties worden meegeven.

10.4.2 Communicatie applicaties

Voor de communicatie tussen applicaties zijn er diverse mogelijkheden beschikbaar. Sommige opties worden echter snel uitgesloten vanwege incompatibiliteit met de vereiste versie van het .NET Framework of .NET. Voorbeelden hiervan zijn WCF, .NET Remoting, Message Queuing en gRPC, hierover is meer te vinden in het onderzoeksdocument frontend-technieken.pdf [5] [6] [7] [8] [9] [10].

Het is mogelijk om te communiceren via bestanden. In dat geval wordt een bestand of meerdere bestanden gebruikt om data te schrijven, en moeten de applicaties deze bestanden periodiek uitlezen. Dit is de meest simpele oplossing, maar heeft ook veel nadelen. Het periodiek bekijken van het bestand maakt real-time communicatie onmogelijk en kan prestatieproblemen veroorzaken. Bovendien zijn er weinig beveiligingsmogelijkheden beschikbaar bij deze methode. Deze optie kan wel nuttig zijn als er grote hoeveelheden niet-beveiligde data moeten worden gedeeld die niet vaak aangepast hoeven worden.

Via anonymous pipes, named pipes en socket-programmering kunnen berichten worden verstuurd en kunnen specifieke berichten worden gebruikt om bepaalde acties te starten, zoals het openen van een scherm.

10.4.2.1 Anonymous pipes

Anonymous pipes is een vorm van Inter-Process Communication (IPC) die alleen gebruikt kan worden tussen processen op dezelfde computer, dus niet over een netwerk [11] [12] [13] [14] [15] . Het kan alleen worden gebruikt tussen parent-child processen of threads binnen dezelfde applicatie. Een anonymous pipe bestaat uit een server en een client, waarbij slechts eenrichtingsverkeer mogelijk is. Het is echter mogelijk om met twee anonymous pipes toch tweerichtingsverkeer op te zetten, door gebruik te maken van de PipeDirection. Dit geeft aan welke richting de communicatie in de pipe op gaat.

10.4.2.2 Named pipes

Named pipes biedt de mogelijkheid om meerdere clients met één server te verbinden [16]. Het kan worden gebruikt voor eenzijdige of dubbelzijdige communicatie. Het ondersteunt impersonisatie, dit betekend dat de server de acces token van de client kan gebruiken. Hiermee kunnen clients hun eigen machtigingen gebruiken bij de server. Named pipes is ook mogelijk te gebruiken over een netwerk.

10.4.2.3 Socket- programmering

Socket-programmering wordt vaak gebruikt voor communicatie over het netwerk, maar kan ook gebruikt worden om te communiceren tussen twee processen op dezelfde computer [17] [18] [19]. Hierbij wordt gebruik gemaakt van een server en een client, waarbij een IP-adres en een poortnummer wordt gebruikt om een connectie op te zetten. Er moet een keuze worden gemaakt voor het soort protocol dat gebruikt wordt bij het versturen van de data. Hierbij kan worden gekozen tussen TCP/IP en UDP. Waarvan UDP sneller is en TCP/IP betrouwbaarder is. Er zijn veel mogelijkheden om extra functionaliteit toe te voegen aan de communicatie via sockets.

10.4.3 Gegevensuitwisselingsformaten

Voor het overdragen van gegevens van de ene naar de andere applicatie wordt een gegevensformaat gebruikt. Dit zorgt ervoor dat de gegevens correct worden overgenomen in de ontvangende applicatie.

10.4.3.1 JSON

Javascript Object Notation (JSON) is een lichtgewicht gegevensuitwisselingsformaat [20]. Het kan onafhankelijk van de programmeertaal worden gebruikt. Binnen JSON worden key/value-paren en lijsten met waarden gebruikt. Het is leesbaar voor mensen en eenvoudig voor machines om te analyseren en genereren. Het is een veelgebruikt gegevensuitwisselingsformaat waar veel kennis over beschikbaar is.

10.4.3.2 Protocol Buffers

Protocol Buffers is een mechanisme voor de serialisatie van gestructureerde gegevens [21] [22]. Bij Protocol Buffers wordt een schema gebruikt om de structuur van de gegevens te beschrijven. Dit schema wordt gebruikt om code te genereren voor verschillende programmeertalen. Zodra gegevens zijn geserialiseerd naar Protocol Buffers, zijn ze niet meer leesbaar voor mensen. Protocol Buffers staat bekend om zijn snelheid en compactheid. Het wordt niet aanbevolen voor gebruik met niet-objectgeoriënteerde programmeertalen. Het is mogelijk om een klasse te maken die gebaseerd is op andere klassen bij Protocol Buffers. Op deze manier kan de ontvanger zelf zien welk type bericht er is verzonden.

Voor Protocol Buffers is er een bestaande bibliotheek genaamd protobuf-net voor eenvoudig gebruik in .NET [23]. Deze ondersteunt .NET Framework 4.7.2, .NET 7 en .NET Standard 2.0, waardoor het geschikt is voor dit project.

Bij het maken van een Protocol Buffer van een klasse moeten de klasses worden gemarkeerd als een ProtoContract en de objecten met een ProtoMember. Elke ProtoMember moet een unieke integer hebben binnen het ProtoContract. Zie onderstaand voorbeeld.

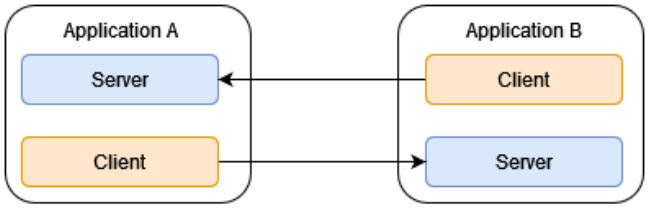
```
[ProtoContract]
public class Message
{
    [ProtoMember(1)]
    public int objectOne { get; set;}

    [ProtoMember(2)]
    public string objectTwo { get; set; }

}
```

Code voorbeeld 1 Protocol bericht

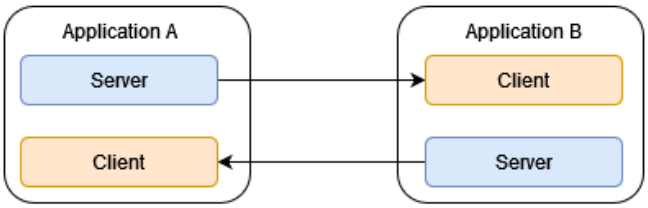
10.4.4 Full-duplex

Het is noodzakelijk dat de communicatie plaats vind in full-duplex modus omdat het in delen van de applicatie nodig is dat de applicaties tegelijkertijd met elkaar zouden kunnen communiceren [24]. Als er slechts één stream is waarbij communicatie in beide richtingen mogelijk is, wordt dit half-duplex genoemd. Dit komt doordat er per keer slechts één bericht door de stream kan worden verzonden. Om full-duplex communicatie te realiseren, zijn er daarom twee streams nodig. Bij het gebruik van named pipes en sockets zou de communicatie er als volgt uit kunnen zien: de

client communiceert met de server, waarbij de applicatie zowel een server als een client bevat, of de server communiceert met de client, waarbij de applicatie ook een server en een client bevat. Zie de afbeeldingen hieronder:

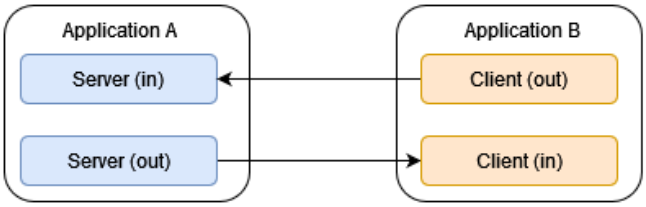


Figuur 6 Full-duplex client naar server



Figuur 5 Full-duplex server naar client

Bij Anonymous Pipes is full-duplex ook mogelijk, maar werkt dit op een andere manier. De applicatie waarmee gecommuniceerd wordt, moet worden gestart vanuit de andere applicatie. Dit is omdat anders de server handle niet meegeven kan worden aan de client. In het geval van dit project start de .NET Framework 4.7.2 applicatie de .NET 7 applicatie op. De .NET Framework applicatie moet dan twee servers bevatten: één server voor de in-richting en één server voor de uit-richting. De .NET 7 applicatie heeft twee clients nodig: één client voor de in-richting en één client voor de uit-richting. Bij het starten van de .NET 7 applicatie wordt van iedere server een client handle meegegeven. Deze handle wordt door de client gebruikt om een connectie te maken met de server.



Figuur 7 Full-duplex anonymous pipes

10.4.5 Communicatie testen

Om te bepalen welke communicatie middelen het best zijn voor Logic4 zijn er drie soorten tests uitgevoerd: prestatietesten, tests op codecomplexiteit en beveiligingstests. Hiervan vind Logic4 prestatie en beveiliging belangrijker dan codecomplexiteit omdat er bij prestatie en beveiliging tijdens elk gebruik profijt van is te zien, bij complexiteit kost vooral het bouwen en onderhoud extra geld en moeite.

10.4.5.1 Test applicatie

Om deze tests uit te voeren zijn twee kleine applicaties gemaakt, één in .NET Framework 4.7.2 en één in .NET 7. Deze applicaties hebben een cijfer dat op 0 staat en een knop. Telkens wanneer er op de knop van de ene applicatie wordt gedrukt, wordt het cijfer van de andere applicatie met 1 verhoogd.

10.4.5.2 Prestatie

Er zijn drie prestatietesten uitgevoerd om de duur van acties te meten. Hiervoor is een stopwatch in de code zelf gebruikt, die de duur tot op milliseconde nauwkeurig weergaf. Voor het testen van het starten en verbinden van de server werd de stopwatch gestart vóór het uitvoeren van de bijbehorende code-opdrachten en gestopt nadat deze opdrachten waren verzonden. Daarna kon de rest van de code worden uitgevoerd. Voor de flow van deze test zie “Bijlage I. Flow test start + connect”.

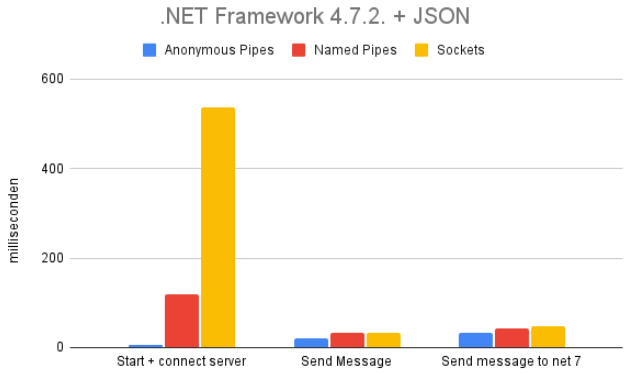
Daarnaast is de tijd gemeten die nodig is om een bericht te verzenden. Hierbij werd een stopwatch gestart in de code zodra er op de verzendknop werd geklikt. Deze werd gestopt zodra de code in de knopmethode was voltooid. Hierbij werd vooral gekeken naar de serialisatie van een bericht en verzendfunctie van de client of server.

Ten slotte is de tijd gemeten vanaf het moment van het klikken op de verzendknop tot aan het ontvangen van het bericht in de andere applicatie. Omdat dit zich in twee verschillende applicaties bevindt, kon er geen stopwatch in de code worden gebruikt. In plaats daarvan werd de huidige tijd afgedrukt. Door het verschil te berekenen tussen

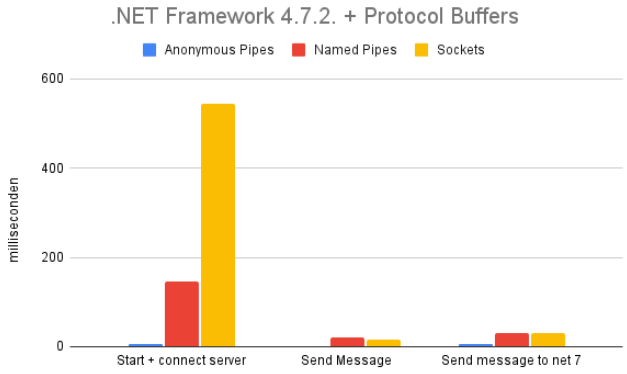
de nieuwere tijd en de oudere tijd, werd de duur van het verzenden van een bericht van de ene naar de andere applicatie gemeten. Dit werd tot op milliseconde nauwkeurig uitgevoerd. Voor de flow van deze testen zie “Bijlage J. Flow test message tijd”.

Alle drie deze testen zijn tien maal uitgevoerd. Hierbij werd er elke keer opnieuw de applicaties gestart en vervolgens één keer bij beide kanten op de knop gedrukt. In de resultaten worden de gemiddelden hiervan behandeld.

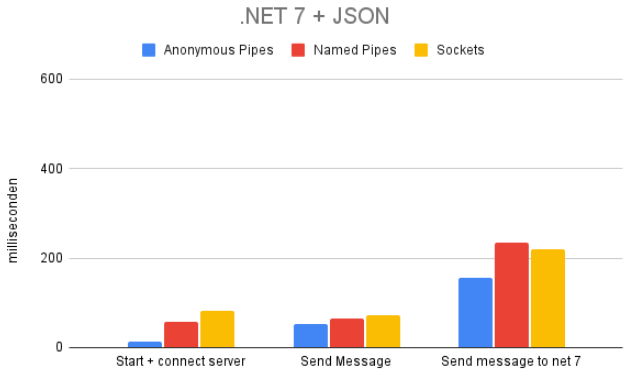
In de onderstaande grafiek worden de gemiddelde resultaten weergegeven van deze testen, voor de gehele resultaten zie Bijlage D. Test resultaten communicatie en Bijlage E. Test resultaten data format.



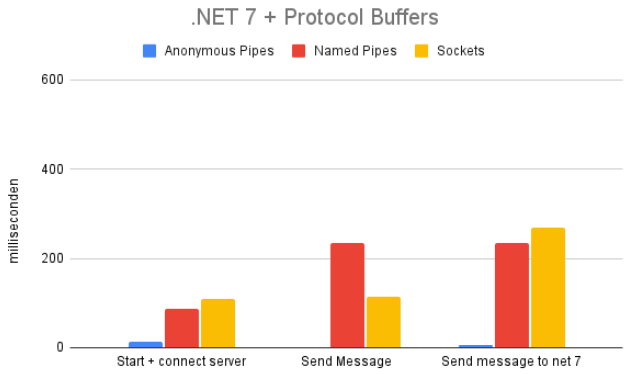
Figuur 10 Prestatie .NET Framework 4.7.2. + JSON



Figuur 11 Prestatie .NET Framework 4.7.2. + Protocol Buffers



Figuur 9 Prestatie .NET 7 + JSON



Figuur 8 prestatie .NET 7 + Protocol Buffers

Uit de bovenstaande grafieken is duidelijk te zien dat Anonymous pipes het snelst is. Dit geldt zowel voor het starten van de server, als voor het omzetten en verzenden van het bericht naar de andere applicatie.

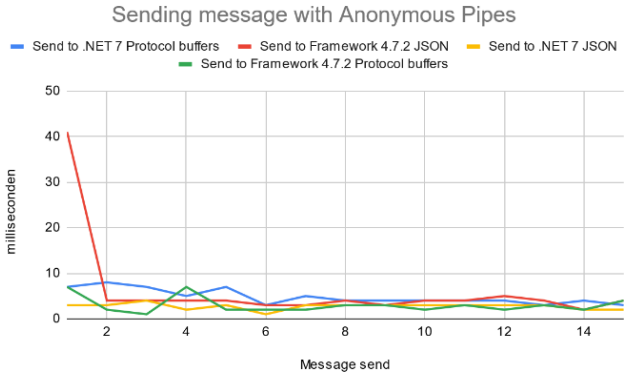
Tabel 3 Gemiddelde tijden per communicatie techniek

Milliseconden	Start + connect	Send Message	Send to other application
Anonymous Pipes	9,50	18,00	50,75
Named Pipes	103,00	88,25	135,50
Sockets	318,00	59,00	142,25

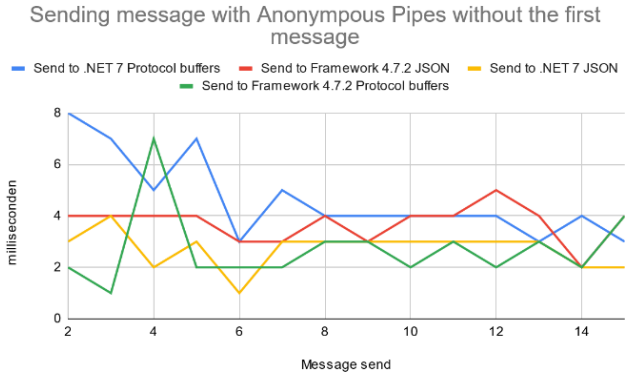
In de bovenstaande tabel zijn de gemiddelde tijden per communicatietechniek te zien. Deze tijden vertegenwoordigen het gemiddelde van beide applicaties en beide gegevensformaten. Uit de tabel blijkt dat anonymous pipes meer dan 10 keer sneller is dan named pipes en meer dan 33 keer sneller dan sockets bij het verzenden van een bericht. Bovendien is anonymous pipes bijna 5 keer sneller dan named pipes en meer dan 3

keer sneller dan sockets bij het verzenden van een bericht. Tot slot blijkt uit de tabel dat anonymous pipes ook meer dan 2,5 keer sneller is dan named pipes en sockets bij het verzenden van een bericht naar de andere applicatie.

Er is ook getest hoelang het duurt om vijftien berichten snel achter elkaar te versturen, waarbij de applicaties slechts één keer worden opgestart. Hieruit blijkt dat de eerste keren mogelijk langer kan duren bij zowel Protocol Buffers als JSON. Bij JSON is het verschil opvallend groot bij het eerste en tweede bericht vanaf .NET Framework 4.7.2., dit verspringt van 41 naar 4 milliseconden. Vanaf het tweede bericht is er echter weinig verschil te zien tussen JSON en Protocol Buffers. Volgens literatuur dat staat op het National Library of Medicine is er bij een onderzoek van Thompson *et al* bepaald dat de gemiddelde reactie tijd van mensen op visuele stimuli 180 tot 200 milliseconden is [25]. Dit betekend dat een mens dit op zichzelf waarschijnlijk niet zal merken maar het kan wel invloed hebben wanneer aspecten die met het bericht te maken hebben ook tijd kosten.



Figuur 12 Versturen van berichten met Anonymous Pipes



Figuur 13 Versturen van berichten met Anonymous Pipes zonder het eerste bericht

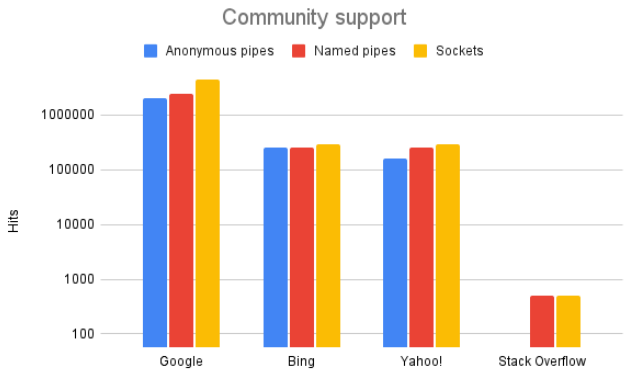
10.4.5.3 Codecomplexiteit

Codecomplexiteit geeft aan hoe ingewikkeld de code is [26]. Hierbij is gekeken naar het aantal lineaire onafhankelijke paden die zich in de code bevinden. Volgens NIST SP 235 is een limiet van 10 een goede richtlijn om aan te houden bij de analyse van codecomplexiteit [27]. Alleen delen van anonymous pipes gaan hierboven uit. De client-methoden van anonymous pipes hebben een complexiteit van 11 en 12.

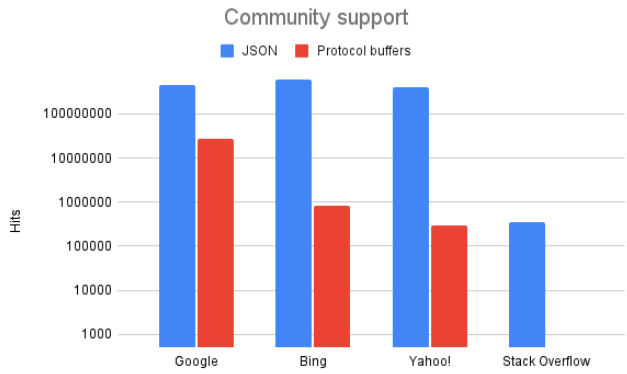
Bij class coupling is het magische getal 9 [28]. Veel code komt hier dichtbij of gaat hier net overheen. Bij sockets is er echter te zien dat dit regelmatig wordt overschreden, vooral bij de server startmethode, die een koppeling aantal heeft van 17.

De onderhoudbaarheid van de code kan met een formule van Microsoft worden berekend. Uit deze formule komt een score van 0 tot en met 100 [26]. Code wordt als goed beschouwd wanneer deze een score van 20 of hoger heeft. In de code is nergens een waarde lager dan 20 te vinden. Er is een klein verschil met anonymous pipes, die gemiddeld een score van 86 hebben, en named pipes en sockets, die gemiddeld 91 en 90 scoren hebben. Voor de gehele resultaten van de codecomplexiteit, class coupling en onderhoudbaarheid zie “Bijlage G. Test resultaten codecomplexiteit”.

De ondersteuning van de online community is ook van belang omdat dit invloed heeft op hoe moeilijk het is om problemen in de code op te lossen. Dit is gedaan door te zoeken naar het aantal resultaten op de zoekmachines Google, Bing en Yahoo! Bovendien is er gekeken naar Stack Overflow, een vraag- en antwoordwebsite voor programmeurs. Over het algemeen zijn er de minste zoekresultaten voor anonymous pipes. Dit verschil is het meest merkbaar op Stack Overflow, waar er slechts 56 resultaten zijn voor anonymous pipes. Tussen JSON en Protocol buffers is een groot verschil te zien. Hierbij is er te zien dat er veel meer community support is voor JSON. Zie figuur 14 en 15 op de volgende pagina voor de grafieken van deze data. Voor de gehele data zie “Bijlage F. Test resultaten community”.



Figuur 14 Community support communicatie technieken



Figuur 15 Community support gegevensformaten

10.4.5.4 Beveiliging

Om eventuele kwetsbaarheden in de code te identificeren, is de code geanalyseerd met behulp van Snyk. Het resultaat van deze analyse toonde geen kwetsbaarheden aan.

Een groot verschil tussen anonymous pipes, named pipes en sockets is dat named pipes en sockets allebei gebruikt kunnen worden over een netwerk. Met anonymous pipes kan dit niet. Dit betekent dat derden niet met de pipe zouden kunnen verbinden over het netwerk. Daarnaast kan er maar één client verbonden zijn met de anonymous pipe.

JSON is door een mens te lezen, Protocol Buffers niet. Dit maakt Protocol buffers via beveiliging door obfuscation iets veiliger.

10.4.5.5 Conclusie tests

In de onderstaande tabel worden de resultaten weergegeven van de eerder besproken aspecten met betrekking tot communicatie. De scores in deze tabel worden uitgedrukt in sterren. Hoe meer sterren er gevuld zijn, des te beter is het aspect beoordeeld. Er zijn in totaal 5 sterren die gevuld kunnen worden. Bij codecomplexiteit geldt dat er minder complexiteit is naarmate er meer sterren gevuld zijn.

Tabel 4 Multicriteria-analyse communicatie technieken

	Anonymous pipes	Named pipes	Sockets
Prestatie	★★★★★	★★☆☆☆	★☆☆☆☆
Codecomplexiteit	★★★☆☆	★★★★★	★★★★★
Beveiliging	★★★★★	★★★★☆	★★★☆☆

Uit de resultaten blijkt dat de prestaties van anonymous pipes significant beter zijn dan named pipes en sockets. Bovendien heeft anonymous pipes het laagste beveiligingsrisico. Hoewel anonymous pipes wat complexer is en er minder informatie over te vinden is op internet, is het nog steeds de beste keuze vanwege de betere scores op de aspecten die Logic4 belangrijk vindt, namelijk prestatie en beveiliging.

In de onderstaande tabel worden de resultaten weergegeven van de eerder besproken aspecten met betrekking tot gegevensformaten. De sterren functioneren op dezelfde manier als in de vorige tabel:

Tabel 5 Multicriteria-analyse gegevensformaten

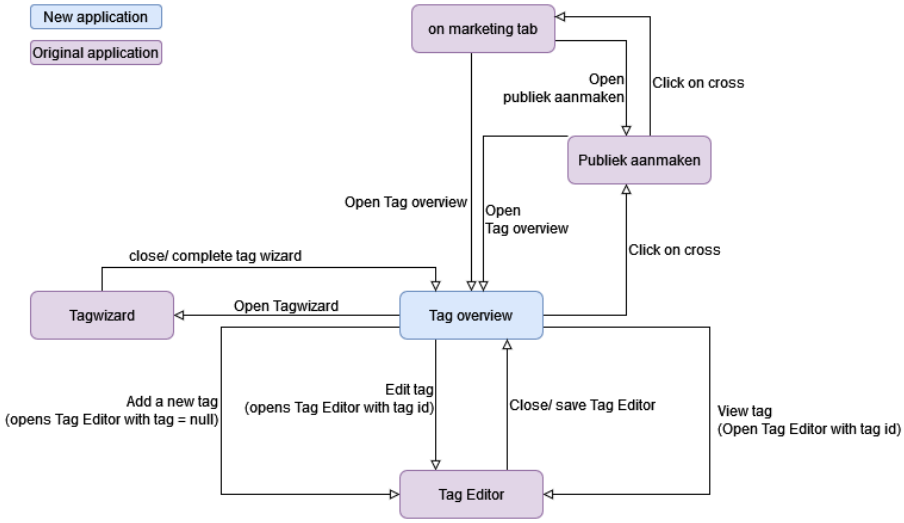
	JSON	Protocol buffers
Prestatie	★★★★☆	★★★★★
Codecomplexiteit	★★★★★	★★★☆☆
Beveiliging	★★★★☆	★★★★★

Protocol buffers is een stuk complexer dan JSON. Echter vind Logic4 prestatie en beveiliging het belangrijker dan codecomplexiteit. Hierop scoort Protocol Buffers net iets beter dan JSON. Hierdoor zal Protocol Buffers worden

gekozen als gegevensformaat voor het Proof of Concept. Het handige aan Protocol Buffers is dat deze zelf kan bepalen welk type bericht er ontvangen is. Mocht Logic4 toch nog voor JSON willen kiezen dan kan dit later zonder al te veel moeite worden aangepast.

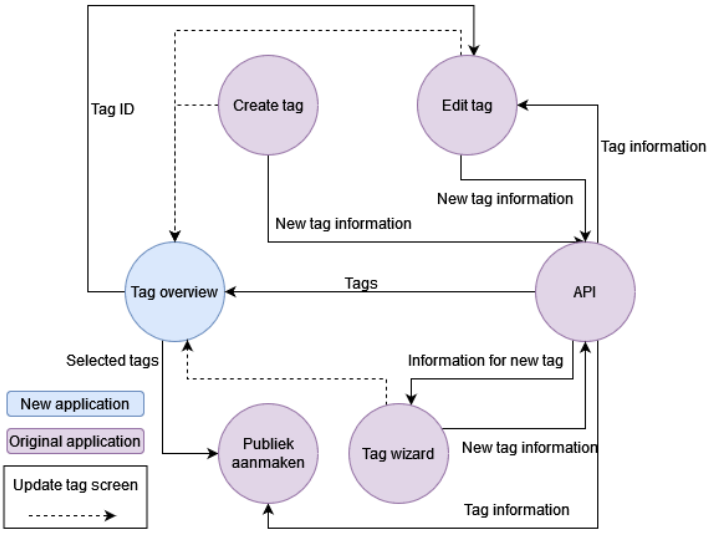
11 Ontwerp

Voor het bewijzen dat een scherm in een andere .NET (7) applicatie kan functioneren alsof het in de bestaande .NET Framework 4.7.2. applicatie zit, wordt er een Proof of Concept gemaakt. Voor dit PoC moet het tag-overzichtsscherm een eigen applicatie worden. Dit is nodig om de applicatie te kunnen moderniseren met nieuwere technieken. Tegelijkertijd moet het nog steeds kunnen communiceren met de oorspronkelijke applicatie. In de afbeelding hieronder is te zien met welke schermen de nieuwe applicatie moet kunnen communiceren. Op de onderstaande afbeelding is te zien met welke schermen in de bestaande applicatie de nieuwe applicatie zal moeten kunnen communiceren.



Figuur 16 Tag overzicht diagram andere schermen opgesplitst nieuwe en bestaande applicatie

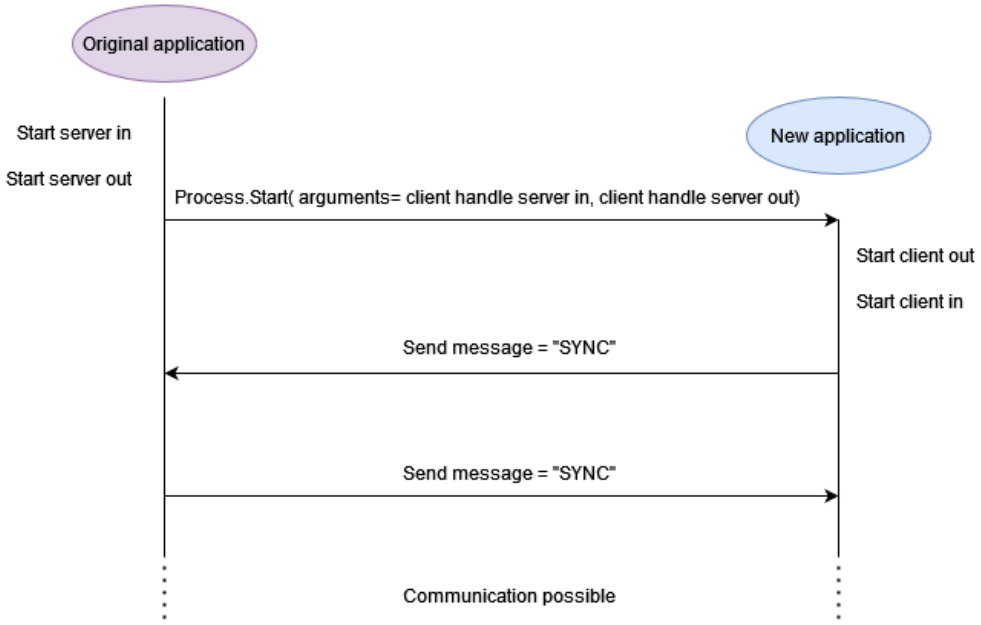
Het tag overzicht scherm wisselt gegevens uit met andere schermen. Bijvoorbeeld: het scherm om een tag te bewerken heeft mogelijk een tag-ID nodig, terwijl het scherm van “publiek aanmaken” de ID's van geselecteerde tags nodig heeft. Bovendien wordt er veel informatie uit de API gehaald, zoals de informatie van een tag via de tag-ID. Deze API is uitsluitend gericht op de marketingaspecten en is een microservice. Verder moet het tag overview scherm weten wanneer het moet worden bijgewerkt na aanpassingen aan de tags op één van de andere schermen. Dit betekent dat de .NET 7 applicatie deze gegevens zou moeten uitwisselen met schermen van de .NET Framework 4.7.2. applicatie. In de onderstaande afbeelding is te zien welke data waarheen moet kunnen worden verstuurd.



Figuur 17 Data flow diagram nieuwe en bestaande applicatie

11.1 Connectie

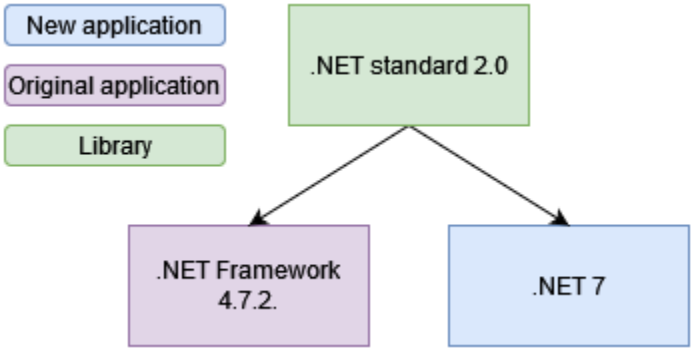
Voor het opzetten van een verbinding worden anonymous pipes gebruikt. Om dit full-duplex te maken, heeft de bestaande applicatie twee anonymous pipe servers nodig. Eén van deze servers heeft een in-richting en de ander een uit-richting, omdat anonymous pipes slechts één richting hebben. Vervolgens wordt de nieuwe applicatie gestart met behulp van `Process.Start`, zodat argumenten kunnen worden meegegeven. Deze argumenten bevatten de pipe-handle's van de server-in en de server-out. Deze worden doorgegeven aan de client. De verbinding wordt voltooid door "SYNC"-berichten heen en weer te sturen. Dit is nodig omdat de server-in en de client-in pas kunnen luisteren naar de pipe wanneer deze is opgezet.



Figuur 18 Diagram connectie Anonymous Pipes

11.2 Library voor anonymous pipes

Om de communicatie tussen beide applicaties mogelijk te maken, wordt een library gemaakt die de anonymous pipe-verbinding implementeert. Omdat de twee applicaties gebruikmaken van verschillende versies van .NET, zal de library worden geschreven in .NET Standard 2.0. Dit is een formele specificatie van .NET API's die het mogelijk maakt om door verschillende .NET-implementaties te worden gebruikt.



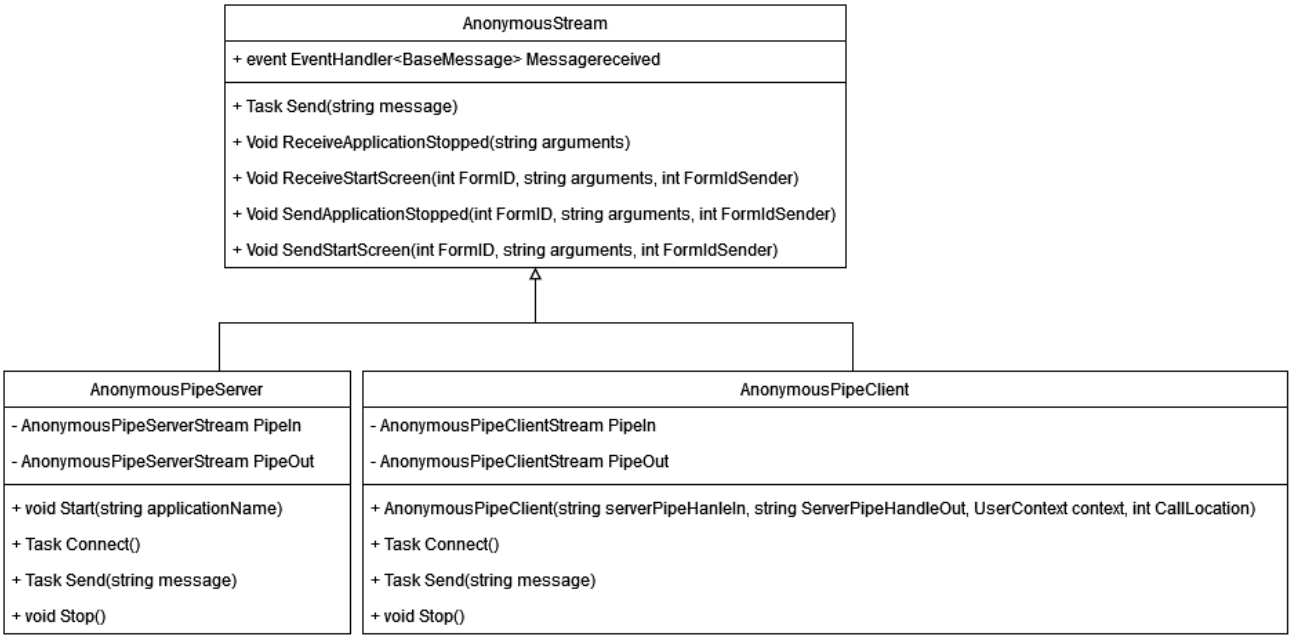
Figuur 19 Diagram .NET versies applicaties + library

Deze library zal de naam "AnonymousPipeCommunication" krijgen. Hierin bevindt zich een klasse genaamd "AnonymousPipeStream", waarin berichten worden gedefinieerd. Omdat zowel de client als de server gebruikmaken van dezelfde berichten, zullen deze klassen de "AnonymousPipeStream" erven. De bestaande

applicatie zal gebruik maken van de server via de "AnonymousPipeServer"-klasse. Deze klasse zal de nieuwe applicatie starten door de bijbehorende bestandsnaam door te geven. De nieuwe applicatie zal de "AnonymousPipeClient"-klasse gebruiken om te communiceren met de oorspronkelijke applicatie. Zie de onderstaande afbeelding.

11.2.1 Starten nieuwe applicatie

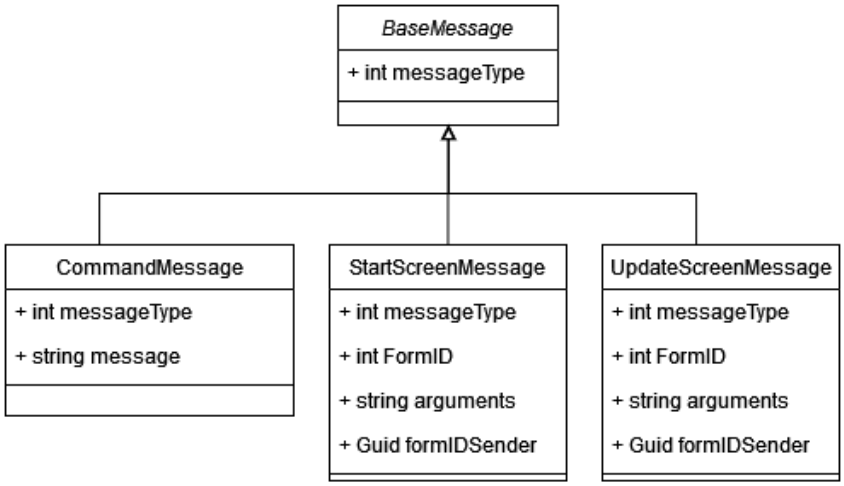
De .NET 7-applicatie zal worden gestart vanuit de .NET Framework 4.7.2. applicatie via de library. Omdat de pipe-handel's moeten worden meegeven aan de nieuwe applicatie zal de functie `Process.Start` gebruikt worden. Via de arguments-variabele van ProcessStartInfo zullen pipe-handle's meegeven worden aan `Process.Start`. Vervolgens worden in de .NET 7-applicatie de handles uit de arguments van de Main functie gehaald en doorgegeven aan de "AnonymousPipeClient"-klasse.



Figuur 20 Class diagram AnonymousStream + server + client

11.2.2 Verschillende berichten

Voor het versturen van berichten wordt gebruikgemaakt van Protocol Buffers. Dit wordt gedaan vanwege de snelheid en de mogelijkheid om het berichttype automatisch te herkennen. Er zijn drie verschillende typen berichten die afstammen van "BaseMessage", een basisbericht dat wordt gebruikt voor verzending. Op de onderstaande afbeelding is de classdiagram van het basis berichten en de berichten die hiervan af stammen te zien.



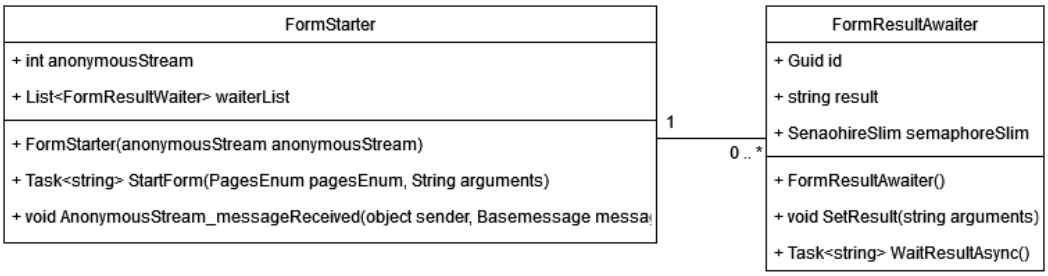
Figuur 21 Class diagram Protocol Buffers

Ten eerste is er een "CommandMessage", een bericht dat een commando zoals "SYNC" kan bevatten, zie figuur 18 Diagram connectie Anonymous Pipes. De ontvanger kan op basis van het meegegeven commando bepalen welke actie er moet worden uitgevoerd. Daarnaast is er een "StartScreenMessage" dat het type scherm bevat dat geopend moet worden, de meegegeven argumenten en de ID van de afzender, zodat de resultaten van het geopende scherm op de juiste manier kunnen worden gebruikt. Tot slot is er een "StopScreenMessage" die wordt gebruikt om aan de andere applicatie door te geven dat een geopend scherm is gesloten. Hierbij worden de resultaten van het scherm teruggestuurd naar de andere applicatie. Omdat de locatie waar het scherm werd geopend relevant kan zijn wanneer het wordt gesloten, wordt dit ook teruggegeven.

Wanneer een applicatie een bericht ontvangt, kan er door de ontvanger worden gekeken naar het type bericht dat is ontvangen en op basis hiervan de juiste actie worden uitgevoerd met de overige informatie.

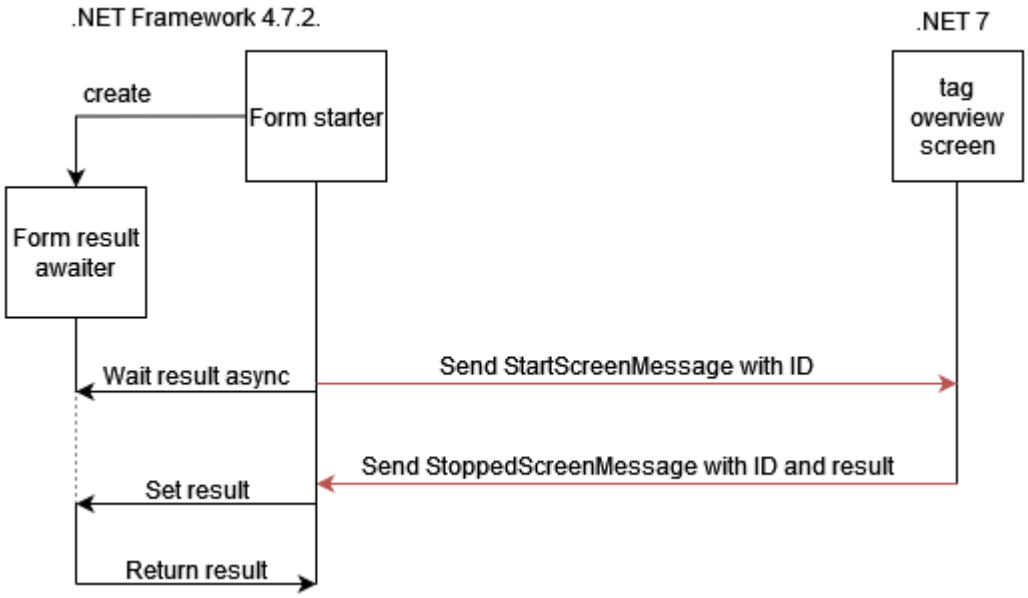
11.2.3 FormStarter

Wanneer Scherm A een ander scherm, Scherm B, opent om gegevens uit Scherm B te verkrijgen, moeten de resultaten van Scherm B correct worden teruggestuurd naar het juiste scherm. Om dit te bereiken, is er de FormStarter. Dit is een klasse die FormResultAwaiters bevat als wachters. Elke wachter heeft een uniek ID voor identificatie, ontvangt het resultaat en maakt gebruik van een SemaphoreSlim om de thread te laten wachten.



Figuur 22 Class diagram FormStarter

De werking van de FormStarter wordt geïllustreerd in het onderstaande voorbeeld, waarbij er verwacht wordt dat het Tag overzicht scherm resultaten retourneert:



Figuur 23 Flow FormStarter

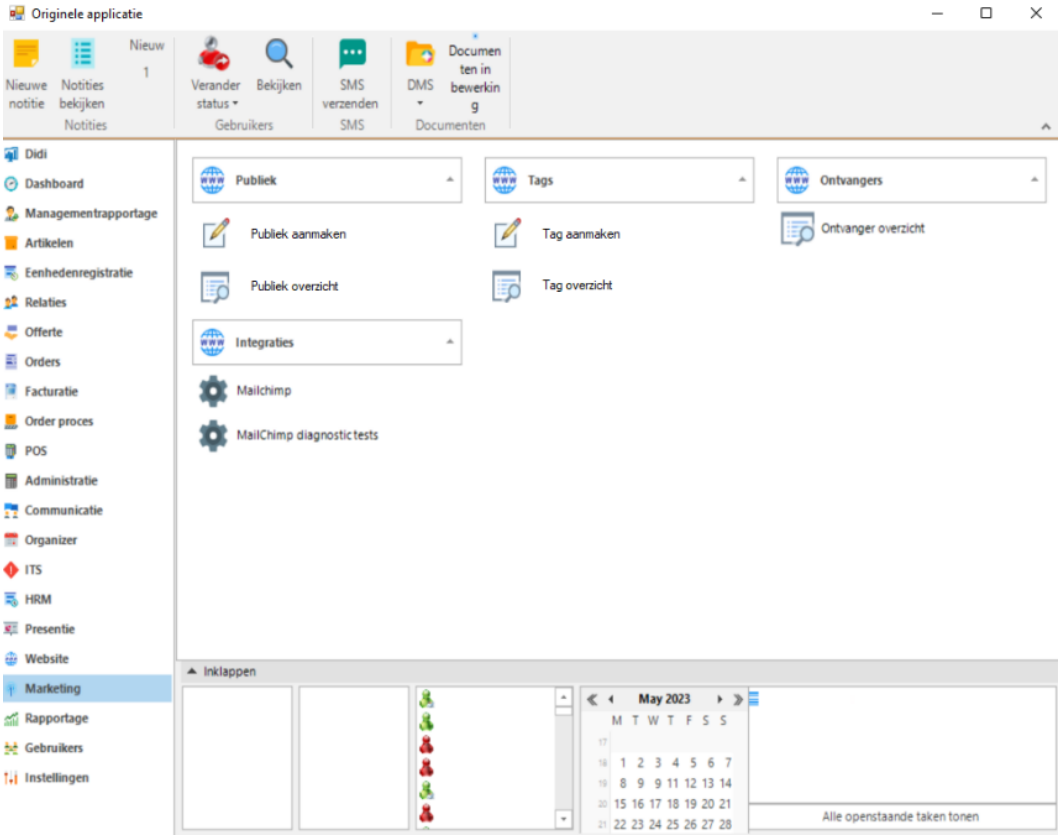
Omdat de FormStarter ook moet werken vanuit het Tag Overzicht-scherm naar een .NET Framework-applicatie, wordt het ID meegegeven via het StartScreenMessage. Nadat dit bericht is verzonden, wordt aan de bijbehorende wachter, FormResultAwaiter, de instructie gegeven te beginnen met wachten. Zodra het resultaat wordt ontvangen via een StoppedScreenMessage, samen met het ID van de wachter, wordt het resultaat doorgegeven aan de wachter, waarna de wachter stopt met wachten. De wachter geeft vervolgens het resultaat terug aan de FormStarter, die op zijn beurt wachtte, zodat de FormStarter het resultaat kan doorgeven aan het scherm dat het oorspronkelijk heeft geopend.

12 Proof of Concept

Het PoC bestaat uit de gedeelde library “AnonymousPipeCommunication”, een .NET framework applicatie, een .NET 7 applicatie en mock data. Aangezien het niet haalbaar is om tijdens de afstudeerstage het ontwerp toe te passen zonder veel kennis te hebben van de oorspronkelijke applicatie, zal er een mock-up worden gemaakt van de relevante delen van de oorspronkelijke applicatie. Met behulp van deze mock-up kan nog steeds worden aangetoond wat enkele mogelijkheden zijn.

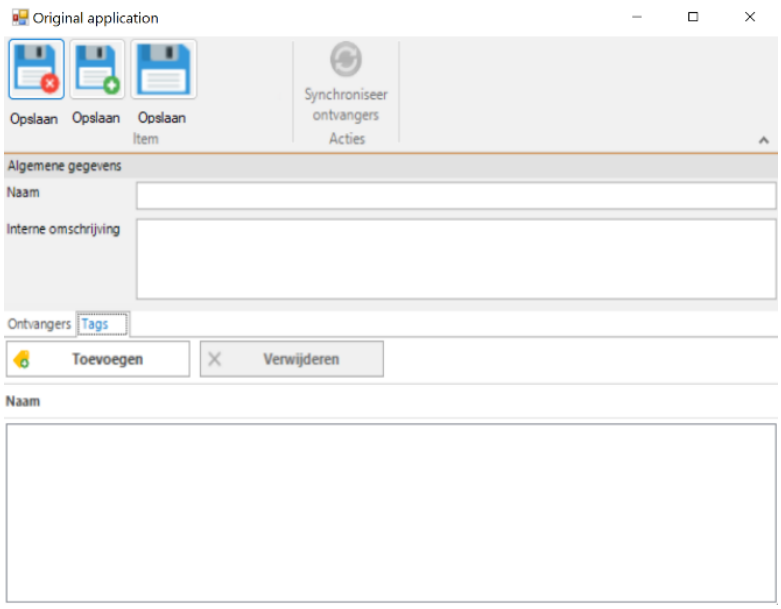
12.1 .NET Framework 4.7.2. applicatie

In de .NET Framework 4.7.2. applicatie zijn het marketingscherm, het publiek aanmaken scherm en het edit tag scherm nagemaakt. Dit is gedaan door middel van een screenshot van het marketingtabblad waaruit enkele knoppen zijn geëxtraheerd en vervolgens zijn nagemaakt met Windows Forms.



Figuur 24 Schermafbeelding marketing scherm

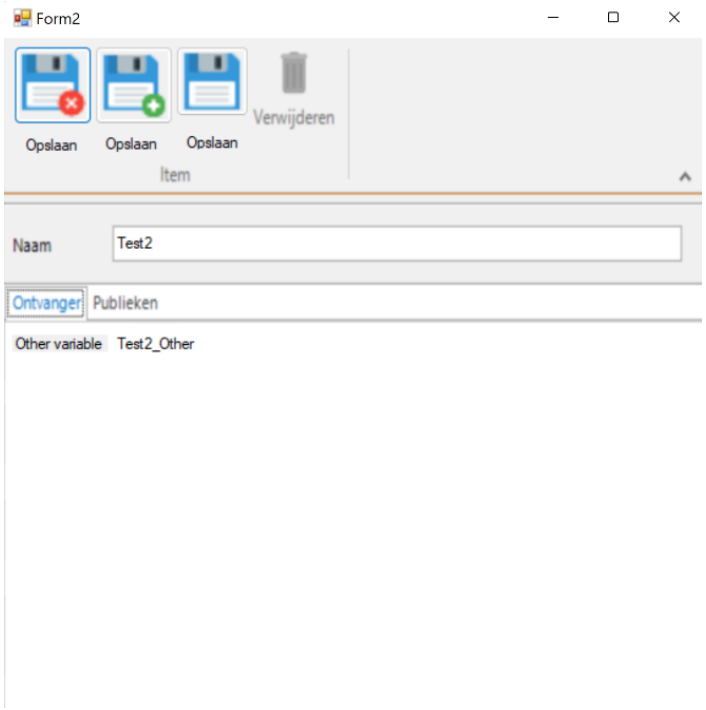
In de mock-up zal meteen het marketingtabblad worden weergegeven wanneer de applicatie wordt gestart. Hierop bevinden zich vier zelfgemaakte knoppen, namelijk "Publiek aanmaken," "Publiek overzicht," "Tag aanmaken" en "Tag overzicht".



Figuur 25 Schermafbeelding publiek aanmaken

De knop “Publiek aanmaken” start het publiek aanmaken scherm, en de knop “Tag overzicht” start het tag overzicht scherm in de nieuwe applicatie.

Via het publiek aanmaken scherm is het mogelijk om tags aan een publiek toe te voegen. Dit kan worden gedaan door op "Toevoegen" te klikken, waarna het scherm voor het overzicht van tags wordt geopend waarin tags kunnen worden geselecteerd.



Figuur 26 Schermafbeelding edit scherm

Via het edittag scherm is het mogelijk om de naam van de tag en een andere string variabele aan te passen. Dit scherm kan alleen worden geopend vanaf het tag overzicht scherm.

12.1.1 Communicatie naar en van de .NET 7 applicatie

De communicatie van deze applicatie wordt geregeld vanuit program.cs, dit is ook het bestand waarin de main-methode zich bevindt. In deze klasse wordt een lijst met servers opgeslagen. Elke server kan met maximaal één client een connectie leggen, daarom moet er voor elke nieuwe client een server beschikbaar zijn. Van het tag overzicht scherm moeten meerdere instanties tegelijkertijd kunnen bestaan.

De methode voor het starten van de tag overview-applicatie (de .NET 7 applicatie) ziet er als volgt uit.

```
public static async Task<string> StartTagOverview()
        {
            var server = new AnonymousPipeServer();
            FormStarter starter = new FormStarter(server);
            server.MessageReceived += ReceiveMessage;
            servers.Add(server);
            //TODO: Change to your location

server.Start(@"C:\Users\LisaMolhoekLogic4\Documents\afstuderen\Proof_of_conecpt_anonymo
us_pipes\proof_of_concept_anonymous_pipes\TagOverview\bin\Debug\net7.0-
windows\TagOverview.exe");
            var result = await starter.StartForm(PagesEnum.t, "");
            return result;
        }
```

Code voorbeeld 2 Start tag overzicht scherm

Zoveel mogelijk verantwoordelijkheden zijn opgenomen in de library. Omdat het scherm voor het aanmaken van het publiek het resultaat terug wil krijgen van het tag overzicht-scherm dat het heeft geopend, wordt er gewacht op het resultaat. Voor het ontvangen van berichten wordt de ReceiveMessage-methode aan de server toegevoegd.

Omdat alle openstaande tag overview-schermen vernieuwd moeten worden wanneer de naam van een tag is aangepast, is de RefreshTagOverviews-methode gemaakt. Hiervoor wordt gebruik gemaakt van het bericht SendApplicationStopped.

```
public static void RefreshTagOverviews(PagesEnum pageEnum)
        {
            foreach (AnonymousPipeServer server in servers)
            {
                server.SendApplicationStopped(pageEnum, "", Guid.NewGuid());
}
        }
```

Code voorbeeld 3 Vernieuw de tag overzicht schermen na aanpassing tag

Wanneer er een bericht wordt ontvangen, wordt de receiveMessage-methode uitgevoerd. Hierin wordt eerst gekeken naar welk type bericht er is ontvangen. Bij dit PoC kan er momenteel alleen iets worden gedaan met een StartScreenMessage. Vervolgens wordt bepaald welk soort scherm er moet worden geopend.

12.2 .NET 7 applicatie

In de .NET 7 applicatie zit het tag overview-scherm. Ook deze achtergrond is gemaakt door middel van een screenshot van de oorspronkelijke applicatie. De wijzigen en de select-knop werken voeren iets uit wanneer er op wordt geklikt. Ook de lijst onder naam is een functionerende lijst. De nieuw knop, de verwijderen knop en de tagwizard knop zijn echte knoppen. De rest op het scherm is een onderdeel van het screenshot.


```
public static void ReceiveMessage(object sender, BaseMessage message)
        {
            if (message is StartScreenMessage startScreen)
            {
                if(startScreen.formID == PagesEnum.editScreen)
                {
                    var tag = MockData.GetTag(Int32.Parse(startScreen.arguments));
                    var formToInvokeOn = Application.OpenForms[0];
                    if (formToInvokeOn.InvokeRequired)
                    {
                        Application.OpenForms[0].Invoke(new Action(() =>
                        {
                            var editScreen = new EditTagScreen(tag.ID, tag.Name,
tag.OtherVariable);
                            editScreen.Show();
                        }));
                    }
                    else
                    {
                        var editScreen = new EditTagScreen(tag.ID, tag.Name,
tag.OtherVariable);
                        editScreen.Show();
                    }
                }

            }
        }
```

Code voorbeeld 4 Het ontvangen en afhandel van een bericht aan de server kant

Bij het starten van de .NET 7 applicatie wordt de Main-methode uitgevoerd. Om de argumenten op te halen, die de pipe-handvatten bevatten die de clients moeten gebruiken om een verbinding op te zetten, moeten de parameter strings[] args worden opgehaald.

12.2.1 Communicatie naar en van .NET framework 4.7.2 applicatie

Omdat de client van deze applicatie alleen kan verbinden met één server, namelijk de bestaande applicatie, is er ook maar één instantie van “AnonymousPipeClient” nodig. De “AnonymousPipeClient” wordt meteen aangemaakt bij het aanmaken van de klasse waarin deze staat. Dit is public gemaakt zodat de client vanuit de hele applicatie kan worden bereikt. Wanneer er dan op de edit knop wordt geklikt kan er meteen een startScreenMessage worden verzonden.

```
Private void BtnIconChange_Click(object sender, EventArgs e)
        {
            Tag tag = (Tag)ListBoxTags.SelectedItem;
            Program.client.SendStartScreen(PagesEnum.editScreen, tag.ID.ToString(), new
Guid());
        }
```

Code voorbeeld 6 Verzend SendStartScreen wanneer er op de knop wordt gedrukt

Het ontvangen van berichten werkt hetzelfde als de implementatie bij de bestaande applicatie. Het StartScreenMessage wordt gebruikt wanneer er resultaat van het tag overview scherm wordt verwacht. Het ID van

```
static void Main(string[] args)
        {
            //If you need to debug this method.
            //Thread.Sleep(15000);

            // To customize application configuration such as set high DPI settings or
default font,
            // see https://aka.ms/applicationconfiguration.

            //Set anonymous pipes communication
            var arguments = args[0];
            var anonymousHandleIn = arguments.Split(',')[0];
            var anonymousHandleOut = arguments.Split(',')[1];

            client.Start(anonymousHandleIn, anonymousHandleOut, "", 0);
            client.MessageReceived +=ReceiveMessage;

            //StartScreen
            ApplicationConfiguration.Initialize();
            tagScreen = new TagOverviewScreen();
            Application.Run(tagScreen);
        }
```

Code voorbeeld 5 Main methode van .NET 7 applicatie

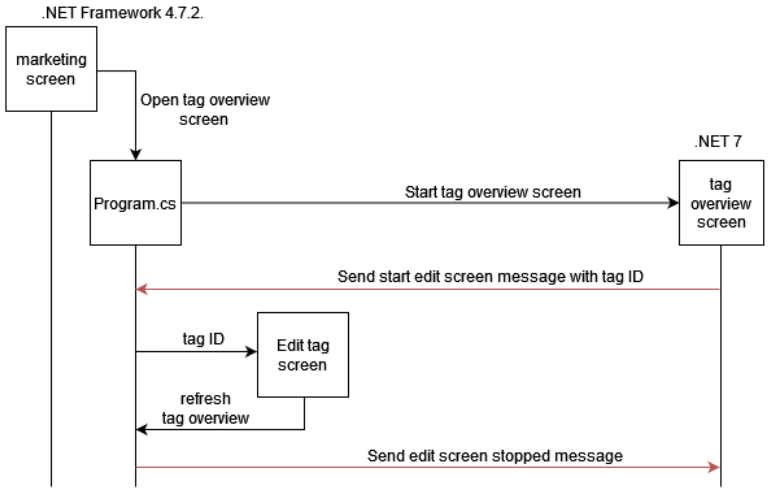
het scherm dat het tag overview scherm heeft gestart, wordt dan meegegeven aan het scherm.

```
public static void ReceiveMessage(object sender, BaseMessage message)
        {
            Debug.Print("TagOverview received a message");
            if (message is StopScreenMessage closeTagScreen)
            {
                Debug.Print("Test stop screen");
                tagScreen.UpdateTags();

            }
            else if (message is CommandMessage commandMessage)
            {
                Debug.Print("Tag command: " + commandMessage.message);
            }
            else if (message is StartScreenMessage startScreenMessage)
            {
                (Application.OpenForms[0] as TagOverviewScreen).Id =
startScreenMessage.formIDSender;
            }
        }
```

Code voorbeeld 7 Het ontvangen en afhandelen van een bericht

12.3 Mogelijke communicatie

Het is mogelijk om vanuit de .NET Framework 4.7.2 applicatie een tagoverzichtscherm te openen in de .NET 7-applicatie. Vervolgens kan een tag worden aangepast via een scherm in de .NET Framework 4.7.2-applicatie. Om

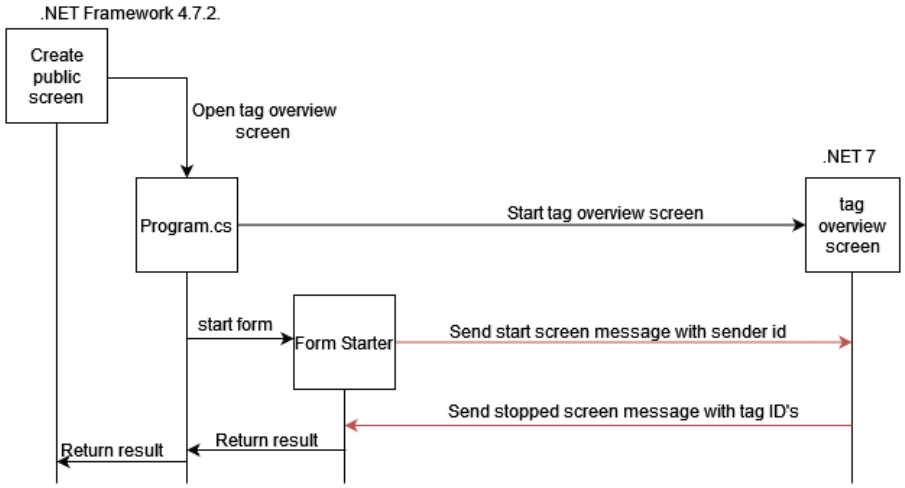


Figuur 28 Diagram edit tag

dit te communiceren, wordt de "AnonymousPipeCommunication" library gebruikt. Dit bericht wordt verwerkt in program.cs (het bestand waar zich ook de main-methode bevind), dat ervoor zorgt dat het edit scherm met de juiste tag wordt geopend. Zodra het edit scherm voltooid is, wordt dit doorgegeven aan program.cs, die vervolgens een StoppedScreenMessage via de "AnonymousPipeCommunication" library naar de .NET 7 applicatie stuurt. Hierdoor wordt het tag overzicht scherm vernieuwd met de nieuwe informatie. Zie de onderstaande afbeelding, de rode lijn toont anonymous pipe communicatie.

Het publiek aanmaken scherm moet van het tag overzicht scherm één of meerdere tags terug krijgen. Hierbij kan er een probleem ontstaan als er meerdere publiek aanmaak schermen open staan. In dat geval moeten de geselecteerde tags wel bij het juiste publiek aanmaken scherm uitkomen.

Om dit probleem op te lossen is de form starter gemaakt. Deze class stuurt een ID naar het tag overzicht scherm en gaat wachten op een reactie. Vervolgens stuurt het tag overzicht scherm het resultaten samen met het eerder ontvangen ID, dit komt binnen bij de form starter. De form starter zoekt dan de juiste wachter op en geeft hem de resultaten waardoor de wachter weet dat hij mag stoppen met wachten. Dit resultaat geeft de wachter aan program.cs en dit geeft dit weer terug aan het publiek aanmaken scherm. Zie de onderstaande afbeelding voor de flow.



Figuur 29 Flow publiek aanmaken

12.4 Gebruikerservaring

De gebruiker zal mogelijk niet meteen opmerken dat er een andere applicatie wordt gestart. Dit komt mede doordat de applicatie nog steeds in .NET is ontwikkeld. Er zijn echter wel een aantal aspecten waaraan dit opgemerkt zou kunnen worden. Schermen die door de andere applicatie worden geopend, verschijnen achter de applicatie die hem opent. Dit gebeurt omdat ze als twee afzonderlijke applicaties worden gezien. Hierdoor kan het voorkomen dat een scherm wordt geopend, maar dat de gebruiker dit niet opmerkt omdat het scherm zich achter het scherm bevindt dat het heeft geopend. Daarnaast worden momenteel twee applicaties weergegeven op de taakbalk. Ook het inlogproces vormt een probleem. De gebruikerscontext wordt niet overgedragen naar de andere applicatie, waardoor de gebruiker opnieuw moet inloggen. Ook kan er bij schermen die oorspronkelijk modal waren gebruikersinteractie plaatsvinden, terwijl dit binnen de oorspronkelijke applicatie niet het geval is. Tot slot sluit de nieuwe applicatie niet wanneer de bestaande applicatie wordt gesloten, dit is wel een verwachting wanneer het uit één applicatie bestaat.

12.5 Technische Implementatie en Ontwikkelingsprocessen

De eerste keer dat de project wordt geopend in Microsoft Visual Studio moet de child applicatie handmatig gebuild worden. Dit moet omdat de .exe file pas wordt gemaakt na een build. Deze .exe file wordt gebruikt om de child applicatie op te starten. Daarnaast moet elke keer wanneer er een aanpassing is gedaan in de child applicatie die getest moet worden de applicatie herbuild worden. Dit wordt niet automatisch gedaan bij het starten van de parent applicatie.

Ook bij het debuggen kunnen problemen voorkomen. De debugger moet handmatig toegevoegd worden. Wanneer het starten getest moet worden kan dit een groter probleem zijn. Een manier om dit op te lossen is door de applicatie het aantal seconden te laten slapen als nodig is om de debugger toe te voegen.

13 Conclusie

De hoofdvraag luidt: “Wat is voor de front end van het ERP pakket van Logic4 een manier om een aparte applicatie te maken van een scherm dat kan worden gestart vanuit de bestaande ERP applicatie en met de bestaande applicatie kan communiceren?”

Hoe zien de onderdelen en principes van de ERP applicatie van Logic4 die relevant zijn voor deze opdracht eruit?

Voor het Proof of Concept hoeft slechts een klein deel van de applicatie te worden uitgewerkt. Dit betreft de schermen die interactie hebben met het tagoverzichtscherm. Dit scherm geeft een overzicht van tags weer. Via andere schermen is het mogelijk om tags toe te voegen, aan te passen en te verwijderen. Ook is het mogelijk om een tag toe te voegen via de tagwizard. Daarnaast kan een publiek worden aangemaakt en kunnen tags daaraan worden toegevoegd. Het scherm voor het aanmaken van een publiek en het tagoverzichtscherm zijn te bereiken via het marketingscherm. Het tagoverzichtscherm moet als een aparte applicatie worden gemaakt, omdat dit scherm de belangrijkste communicatie met andere schermen kan aantonen.

Welke technieken zijn er voor het maken van een aparte applicatie voor een front end?

Veel technieken kunnen direct weggestreept worden doordat deze .NET Framework 4.7.2. en .NET 7 moeten ondersteunen. Voorbeelden hiervan zijn WCF, .NET Remoting, Message Queuing en gRPC.

Voor de communicatie tussen de applicaties kan gekozen worden tussen anonymous pipes, named pipes en socketprogrammering. Anonymous pipes is een vorm van IPC die alleen kan worden gebruikt tussen een parent-childproces en threads binnen dezelfde applicatie. Named pipes is ook een vorm van IPC, maar kan ook worden gebruikt tussen applicaties die geen parent-child relatie hebben en named pipes kan over een netwerk worden gebruikt. Socketprogrammering is een methode om twee nodes met elkaar te verbinden voor communicatie over een netwerk, maar kan ook worden gebruikt tussen programma's op dezelfde computer. Uit tests is gebleken dat anonymous pipes het snelst is, maar ook het meest complex. Socketprogrammering is het minst veilig.

Er zal ook een gegevensuitwisselingsformaat worden gebruikt voor het verzenden en ontvangen van berichten. De twee meest gangbare formaten die beide .NET-versies ondersteunen, zijn JSON en Protocol Buffers. Het voordeel van JSON is dat het veel wordt gebruikt en leesbaar is voor mensen. Het nadeel is echter dat het minder veilig is vanwege de leesbaarheid. Daarnaast is Protocol Buffers bij het eerste bericht aanzienlijk sneller. Protocol Buffers zijn complexer, vooral omdat er online niet veel informatie over te vinden is, maar het is wel gemakkelijk om het type bericht te identificeren.

Hoe moet de aparte applicatie van het gekozen scherm met de best toepasselijke techniek eruitzien?
Aangezien snelheid van de code belangrijker is dan de complexiteit van de code binnen het bedrijf is er voor anonymous pipes gekozen voor het PoC. Door de snelheid en veiligheid van Protocol Buffers zal ook deze worden gekozen voor het PoC. Bovendien kan het automatisch herkennen van het type bericht goed worden gebruikt.

Voor het delen van de code voor anonymous pipes is er voor gekozen een library te maken. Deze library is geschreven in .NET Standard 2.0 om beide .NET-versies te ondersteunen. De library bevat een serverklasse met twee servers en een clientklasse met twee clients. Dit maakt full-duplex communicatie mogelijk binnen de applicatie. De serverklasse wordt gebruikt door de .NET Framework-applicatie en de clientklasse wordt gebruikt door de .NET 7-applicatie. Dit komt doordat de applicatie die de andere applicatie start, de server moet bevatten. Voor Protocol Buffers zijn vier soorten berichten gemaakt: een basisbericht waar alle andere berichten van afgeleid zijn, een commandobericht voor het verzenden van kleine opdrachten zoals "SYNC", een StartScreenMessage waarmee een scherm in de andere applicatie kan worden gestart, en een StoppedScreenMessage waarmee kan worden aangegeven dat een scherm is gestopt. Met deze berichttypen weet de ontvanger wat hij met de informatie in het bericht moet doen.

Wat zijn de gevolgen van het maken van een aparte applicatie van het gekozen scherm met de best toepasselijke techniek uit de bestaande applicatie?

Het is mogelijk om een .NET 7-applicatie naast een .NET Framework-applicatie te laten draaien en te laten functioneren als één geheel. Het is mogelijk om een scherm te openen dat zich in een andere applicatie bevindt.

Dat scherm in de andere applicatie kan op zijn beurt een scherm openen in de bestaande applicatie. Aanpassingen in gegevens kunnen worden gecommuniceerd naar de andere applicatie, zodat de gegevens altijd up-to-date zijn. Bovendien is het mogelijk om een scherm in de andere applicatie te openen, gegevens daaruit te halen en deze gegevens terug te communiceren naar het specifieke scherm dat het andere scherm heeft geopend. Dit toont aan dat er veel mogelijkheden zijn wat betreft gegevensuitwisseling. Het is echter nog steeds mogelijk dat de gebruiker ervaart dat het twee verschillende applicaties zijn. Ook kunnen er uitdagingen zijn voor ontwikkelaars die aan de andere applicatie moeten werken.

Het bedrijf

Logic4 is optimistisch over de vooruitgang van dit project. Hoewel er nog verder onderzoek nodig is, zijn ze ervan overtuigd dat ze kunnen voortbouwen op dit project vanwege het grondige en gedegen onderzoek, inclusief uitgebreide tests. Ze zijn van plan om de adviezen, zoals te lezen in het volgende hoofdstuk verder te onderzoeken. Het is van cruciaal belang voor hen om met dit project verder te gaan, aangezien dit een van de weinige mogelijkheden is om hun huidige front-end applicatie te moderniseren.

14 Advies

Er zijn nog een aantal aspecten die mogelijk nog meer onderzoek nodig hebben daarnaast zijn er nog andere aspecten waar Logic4 naar zou kunnen kijken. Dit wordt hieronder besproken.

14.1 Het toepassen van dit project

Het is raadzaam om na verder onderzoek dit project toe te passen binnen de bestaande applicatie. Hierbij zou het gebruik van anonymous pipes samen met Protocol Buffers een goede keuze zijn, omdat deze technologieën hoog scoren op zowel prestaties als beveiliging. Door deze technieken toe te passen, kan de schaalbaarheid van het project worden verbeterd. Bovendien zal het project beter bestand zijn tegen toekomstige ontwikkelingen, aangezien nieuwe en mogelijk andere technologieën kunnen worden geïmplementeerd. Voor de alternatieven zie het adviesrapport.

14.2 Inloggen

In het PoC is de functionaliteit voor het inloggen niet gerepliceerd. Dit betekent dat verzoeken naar de API niet kunnen worden voltooid als dit PoC zou worden geïmplementeerd in de huidige applicatie. Dit probleem kan worden opgelost door de gebruiker opnieuw te laten inloggen of door de gebruikerscontext door te geven. Aangezien het bedrijf wil voorkomen dat de gebruiker het gevoel heeft met verschillende applicaties te werken, lijkt het doorgeven van de gebruikerscontext de beste oplossing. Dit kan worden geïmplementeerd bij het opstarten van de applicatie of via anonymous pipes.

Een andere mogelijkheid is om alle verzoeken naar de database te laten verlopen via de bestaande applicatie, door deze verzoeken via de anonymous pipe te verzenden. Het is belangrijk om te onderzoeken welke aanpak de beste oplossing biedt voor dit project.

14.3 Gebruikerservaring

Er is geconstateerd dat schermen soms onder het actieve scherm terechtkomen, wat te wijten is aan het feit dat het twee afzonderlijke applicaties zijn. Dit probleem kan mogelijk worden opgelost door gebruik te maken van de TopMost'-eigenschap van het geopende formulier [29]. Deze eigenschap moet dan geactiveerd en gedeactiveerd worden, omdat het anders permanent bovenop blijft.

Bovendien worden beide applicaties momenteel weergegeven op de taakbalk, wat wellicht kan worden verholpen door beide applicaties hetzelfde 'Application Model ID' toe te wijzen [30]. Dit ID wordt door de shell gebruikt om applicaties te identificeren. Als beide applicaties hetzelfde ID hebben, worden ze gezien als één applicatie.

Daarnaast moet het mogelijk zijn om bij sommige schermen gebruikersinteractie niet meer toe te staan wanneer ze een ander scherm openen. In het Proof of Concept (PoC) kan dit probleem niet worden opgelost wanneer het plaatsvindt tussen de twee applicaties. Mogelijk kan dit worden opgelost door gebruik te maken van 'Control.enabled', maar hier is meer onderzoek voor nodig [31].

Als laatst wanneer de gebruiker de parent applicatie sluit zal ook de child applicatie moeten sluiten. De gebruiker zou immers moeten denken dat er één applicatie gesloten wordt. Dit zou opgelost kunnen worden door bij het `Form.Closing` event van de parent applicatie een StopScreenMessage te sturen naar de child applicatie. Vervolgens zou de child applicatie dat bericht juist moeten afhandelen.

14.4 Ontwikkelaarservaring

Op dit moment ervaart de ontwikkelaar moeilijkheden doordat het .exe-bestand niet automatisch wordt aangemaakt of bijgewerkt wanneer er wijzigingen worden aangebracht. Dit kan frustraties veroorzaken bij de ontwikkelaars. Een mogelijke oplossing hiervoor is het toevoegen van de child applicatie aan de afhankelijkheden (dependencies) van de parent applicatie [32]. Bij het starten van de applicatie worden alle afhankelijkheden gebouwd, zodat de applicatie er gebruik van kan maken.

Daarnaast kan overwogen worden om met nepgebruikersgegevens te werken. Deze aanpak is echter alleen een goede oplossing wanneer er enkel aan de child applicatie gewerkt hoeft te worden. In dat geval kan de child applicatie gestart worden met de nepgegevens.

14.5 Een child applicatie met meerdere schermen

Voor het PoC is een child applicatie ontwikkeld met één scherm. Als alternatief zou het ook mogelijk zijn om een child applicatie naast de bestaande applicatie te creëren, die bestaat uit meerdere schermen. Dit kan worden gerealiseerd door één scherm te maken dat gebruikmaakt van andere schermen, waarbij weinig aanpassingen aan de PoC-code nodig zijn. Daarnaast is het ook mogelijk om een applicatie met meerdere schermen te ontwikkelen, die allemaal vanuit de bestaande applicatie kunnen worden gestart. Voor deze laatste optie zijn echter meer aanpassingen in de code vereist. Dit is omdat dan altijd moet worden doorgegeven voor welk scherm de communicatie bedoeld is.

14.6 Uit de .NET wereld stappen

Het is mijn advies om verder te kijken dan alleen .NET. Er zijn ook andere frameworks die anonymous pipes ondersteunen waarmee een windows applicatie gemaakt kan worden. Een voorbeeld hiervan is Flutter [33]. Dit is een open-source framework voor cross-platform applicaties dat gebruik maakt van de programmeertaal Dart. Dit ondersteund het maken van Android, IOS, web, Windows, macOS, Linux en embedded applicaties. Een reden om voor flutter te kiezen is dat het een meerwaarde kan hebben om van delen van de applicatie ook zo te maken dat het als mobiele applicatie beschikbaar is. Er zou dan met een codebase een Windows en mobiele applicatie kunnen worden gemaakt. Vervolgens zou dan de bestaande .NET Framework 4.7.2. applicatie communiceren met de Flutter windows applicatie. De Mobiele applicatie zou dan dezelfde API calls kunnen maken.

Naast Flutter zouden er ook naar andere frameworks kunnen worden gekeken.

15 Reflectie

In deze reflectie kijken we terug op de inhoud van het project, geven we zelfreflectie en bespreken we de verworven competenties.

15.1 Inhoudelijke reflectie

Het project is succesvol verlopen. Er is meer tijd besteed aan het testen van verschillende communicatiemiddelen dan oorspronkelijk gepland in het Plan van Aanpak. Desondanks heeft dit geresulteerd in een uitkomst waar ik volledig achter sta. Het nadeel hiervan is dat er minder tijd overbleef voor het Proof of Concept, waardoor er minder functionaliteiten konden worden getest. Desalniettemin ben ik tevreden met de extra tijd die ik aan het testen heb besteed, omdat hierdoor werd voorkomen dat het PoC waardeloos zou zijn door het gebruik van de verkeerde techniek.

In overleg met het bedrijf is besloten om een mock-up te maken in plaats van direct aan de werkelijke applicatie te werken, vanwege de tijdslimiet. Het voordeel hiervan is dat de communicatie kon worden aangetoond. Het nadeel is dat het inloggen niet kon worden meegenomen. Ook kan het in de uiteindelijke applicatie een probleem zijn dat het gewenste scherm veel klassen deelt met de bestaande applicatie. Daarvoor zal nog een oplossing moeten worden gevonden. Als het bedrijf dit project wil implementeren in de bestaande applicatie, zal waarschijnlijk aanvullend onderzoek nodig zijn.

Voor dit project is gebruikgemaakt van ChatGPT. Er is echter geen inhoudelijke informatie direct overgenomen van ChatGPT. Het AI-programma is gebruikt als zoekmachine. ChatGPT bleek vaak meer manieren te vinden om iets te doen dan wanneer er op Google werd gezocht of een collega werd geraadpleegd. Hierdoor kon er vervolgens onderzoek worden gedaan naar deze alternatieve benaderingen. Dit heeft geleid tot een betere vergelijking van verschillende opties. Wel moet worden opgemerkt dat ChatGPT ook met veel opties kwam die direct konden worden uitgesloten.

15.2 Zelfreflectie

Hier wil ik terugblikken op mijn sterke en zwakke punten. Bovendien wil ik algemene ervaringen delen.

15.2.1 Planning

In het Plan van Aanpak had ik een wekelijkse planning opgesteld, waarin ik zeven weken had gereserveerd voor het maken van het Proof of Concept en het advies. Uiteindelijk ben ik hier twee weken later mee begonnen dan gepland, vanwege de tijd die ik besteedde aan het testen van de verschillende communicatiemogelijkheden. Aan het begin van het project had ik nog niet bedacht dat ik testen zou gaan maken. Dit heeft echter veel tijd in beslag genomen. Toch heb ik hierdoor wel de juiste keuze kunnen maken, en het heeft daarna tijd bespaard bij het ontwerp en het PoC, omdat ik de communicatiemogelijkheden al had verkend. Ook kon ik hierdoor betere keuzes maken voor het ontwerp en het PoC.

15.2.2 projectmanagementkader

B ij dit project heb ik gebruikgemaakt van een projectmanagementkader dat geïnspireerd was door Scrum. Het was prettig om wekelijkse doelen op te stellen, dagelijks voortgangsgesprekken te hebben, wekelijkse voortgangsgesprekken te hebben en te reflecteren. Het bijhouden van individuele issues op de online omgeving is iets wat ik snel vergat en ook weinig nut van had. Omdat ik alleen werkte, kon niemand per ongeluk dezelfde issue oppakken en de personen die op de hoogte moesten zijn van mijn voortgang waren op de hoogte tijdens de

dagelijkse of wekelijkse gesprekken. Het was echter prettig om dagelijkse taken op papier te schrijven, zodat ik ze kon afstrepen. Als ik dit project opnieuw zou starten, zou ik de wekelijkse planning op een andere manier noteren en de dagelijkse planning op papier schrijven.

15.2.3 Nieuwe technologie

Voor mij was het een uitdaging om met de technologie van de bestaande applicatie aan de slag te gaan. Ik had nog nooit een Windows-applicatie gemaakt, nog nooit met .NET gewerkt en nog nooit geprogrammeerd in C#. Daarnaast was het gebrek aan kennis van Windows Forms een uitdaging. In het begin van het project wilde ik zoveel mogelijk zelf uitzoeken, maar dit zorgde er soms voor dat het langer duurde vanwege mijn gebrek aan kennis. Later in het project leerde ik sneller hulp te vragen. Mijn collega's hebben zoveel kennis, het is zonde om daar geen gebruik van te maken.

Bovendien heeft naar mijn weten nog niemand iets soortgelijks geprobeerd voor een Windows-applicatie. Dit beperkte de hoeveelheid beschikbare kennis online. Er waren wel vergelijkbare dingen gemaakt voor het web, maar dit kon niet worden toegepast op Windows. Hierdoor raakte ik tijdens het Plan van Aanpak even op het verkeerde spoor, maar gelukkig kwam ik hier snel achter.

15.2.4 Communicatie

Ik vond dat de communicatie met het bedrijf goed verliep. Hoewel het bedrijf en ik het niet altijd eens waren, konden we heldere gesprekken voeren. Daarnaast waren de eisen en wensen naar mijn mening heel duidelijk, omdat ik hier meerdere keren over had gecommuniceerd. Ook toen we gezamenlijk besloten om een mock-up van de applicatie te maken, was het duidelijk wat de gevolgen hiervan waren.

15.2.5 Grenzen

Ik leg veel druk op mezelf. Ik wil graag alles goed doen en niemand teleurstellen. Dit komt echter niet altijd goed over. Ik denk dat hierdoor het bedrijf extra druk op mij wilde leggen. Doordat ik al veel druk op mezelf legde, het bedrijf druk op mij legde en ik naast dit project ook nog veel verantwoordelijkheden had, werd het me te veel. Dit gebeurde tijdens het weekend van het conceptverslag. Ik heb dit na het weekend aangegeven. Dit heb ik nog niet vaak gedaan en daarom was ik trots op mezelf. Volgens mij kwam dit ook goed over bij het bedrijf. Hierna lukte het me om de draad weer op te pakken.

Daarnaast had ik nog een andere uitdaging. Door een aanrijding in november heb ik sindsdien constant rugpijn. Hierdoor kan stress sneller te veel worden. Het is ook pijnlijk om de hele dag in dezelfde houding te zitten. Soms was het ook een uitdaging om uit bed te komen, omdat ik wist dat ik die dag veel pijn zou hebben. Dit betekende dat ik tijdens dit project veel beter op mijn grenzen moest letten. Ook moest ik een keer eerder naar huis omdat de pijn ondraaglijk werd. Elke maandag werkte ik op afstand, omdat ik aan het eind van de dag naar de fysiotherapeut moest. Dit had als voordeel dat ik één dag per week in een andere omgeving kon werken, wat ik erg prettig vond. Het nadeel was dat ik online minder gemakkelijk om hulp kon vragen. Ik geef de voorkeur aan persoonlijk contact. Het verplichte me wel om beter op mijn grenzen te letten op werk en in privé.

15.2.6 Zelfstandigheid

Ik werkte zelfstandig aan het project. Hoewel ik iemand had waar ik dagelijks bij terecht kon voor begeleiding, was het een individuele opdracht. Daarom deed ik elke week een reflectie. Hiervoor gebruikte ik de STARR-methode met hierbij een eigen toevoeging het punt ‘doel’ [34]. Bij de reflectie keek ik eerst naar de situatie van die week, wat ik had gedaan, waarom ik het zo had gedaan, wat het resultaat was en wat ik ervan vond. Vervolgens keek ik naar

mijn doel voor de komende week. Dit omvatte altijd het werk dat ik wilde doen, zoals het voltooien van het ontwerp voor het Proof of Concept. Soms omvatte het ook persoonlijke doelen, zoals eerder hulp vragen. Dit stelde me in staat om snel mijn gedrag aan te passen en goed op de hoogte te blijven van de voortgang in het project. Dit vond ik erg prettig.

15.3 Competenties

Voor het voltooien van de afstudeerstage zijn minimaal drie competenties op beheersingsniveau drie vereist op het gebied van softwareontwikkeling. Met deze opdracht verwacht ik dat ik vaardigheden op het gebied van ontwerpen, implementeren en adviseren heb beheert.

Ontwerpen
Er is een ontwerp gemaakt voor een softwarearchitectuur die betrekking heeft op zowel bestaande als nieuwe systemen. Bij het ontwerpen is gebruikgemaakt van bestaande technieken om iets nieuws te creëren. Er moest niet alleen rekening worden gehouden met de specifieke situatie van het PoC (Proof of Concept), maar ook met de functionaliteit in de gehele applicatie. Daarnaast zijn de belangrijkste aspecten voor het bedrijf, namelijk prestatie en beveiliging, in overweging genomen. De complexiteit hoefde minder zwaar te wegen.

Realiseren
Op basis van het ontwerp is een PoC ontwikkeld. Het product verbetert de schaalbaarheid van het bestaande project. Aangezien anonymous pipes en Protocol Buffers zo efficiënt werken, is het ook het meest toekomstbestendig bij toenemend gebruik. De PoC sluit aan bij het bestaande softwarepakket, maar er is wel verder onderzoek aanbevolen. Bij dit project zijn bestaande softwaretechnieken op een unieke manier gecombineerd.

Adviseren
Er is advies gegeven over de keuze voor toekomstige softwarearchitectuur. Hierbij is rekening gehouden met prestatie, beveiliging, schaalbaarheid en toekomstbestendigheid. Ook is er advies gegeven over een mogelijk alternatief framework. Ten slotte zijn er aanbevelingen gedaan voor vervolgonderzoek naar specifieke onderdelen.

16 Bronnen

[1] Microsoft, „Interaction.Shell(String, AppWinStyle, Boolean, Int32),” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.interaction.shell?view=net-7.0 . [Geopend 8 maart 2023].

[2] Microsoft, „AppWinStyle Enum,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/microsoft.visualbasic.appwinstyle?view=net-7.0. [Geopend 12 juni 2023].

[3] Microsoft, „Process.Start Method,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.process.start?view=net-5.0. [Geopend 6 maart 2023].

[4] Microsoft, „ProcessStartInfo Class,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.diagnostics.processstartinfo?view=net-7.0. [Geopend 6 maart 2023].

[5] H. e. al, „What Is Windows Communication Foundation,” Microsoft, 16 december 2021. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf . [Geopend 10 maart 2023].

[6] Microsoft, „System.Messaging Namespace,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.messaging?view=netframework-4.8.1&viewFallbackFrom=net-7.0 . [Geopend 13 maart 2023].

[7] C. e. al, „.NET Framework technologies unavailable on .NET,” Microsoft, 3 januari 2023. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/porting/net-framework-tech-unavailable. [Geopend 13 maart 2023].

[8] gRPC , „gRPC,” [Online]. Available: https://grpc.io/. [Geopend 14 maart 2023].

[9] Wikipedia, „Remote procedure call,” 22 januari 2023. [Online]. Available: https://en.wikipedia.org/wiki/Remote_procedure_call . [Geopend 14 maart 2023].

[10] J. e. al, „Inter-process communication with gRPC,” Microsoft, 2 maart 2023. [Online]. Available: https://learn.microsoft.com/en-us/aspnet/core/grpc/interprocess?view=aspnetcore-7.0 . [Geopend 14 maart 2023].

[11] Microsoft, „How to: Use Anonymous Pipes for Local Interprocess Communication,” 14 december 2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/io/how-to-use-anonymous-pipes-for-local-interprocess-communication. [Geopend 13 juni 2023].

[12] Wikipedia, „Anonymous Pipes,” 4 april 2020. [Online]. Available: https://en.wikipedia.org/wiki/Anonymous_pipe. [Geopend 13 juni 2023].

[13] A. Onsman, „What is Interprocess Communication?,” Tutorialspoint, 23 juni 2020. [Online]. Available: https://www.tutorialspoint.com/what-is-interprocess-communication . [Geopend 9 maart 2023].

[14] Wikipedia, „Inter-process communication,” 8 maart 2023. [Online]. Available: https://en.wikipedia.org/wiki/Inter-process_communication. [Geopend 9 maart 2023].

[15] A. e. al, „Pipe Operations in .NET,” Microsoft, 15 september 2021. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/io/pipe-operations. [Geopend 10 maart 2023].

[16] A. e. al, „How to: Use Named Pipes for Network Interprocess Communication,” Microsoft, 15 september 2021. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/standard/io/how-to-use-named-pipes-for-network-interprocess-communication . [Geopend 10 maart 2023].

[17] Microsoft, „System.Net.Sockets Namespace,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.net.sockets?view=net-7.0. [Geopend 14 maart 2023].

[18] GiovanniDezio, „Socket Programming in C#,” GeeksforGeeks, 10 september 2021. [Online]. Available: https://www.geeksforgeeks.org/socket-programming-in-c-sharp/ . [Geopend 14 maart 2023].

[19] IEvangelist en liveans, „Use Sockets to send and receive data over TCP,” Microsoft, 12 januari 2022. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/fundamentals/networking/sockets/socket-services. [Geopend 14 maart 2023].

[20] JSON, „Introducing JSON,” [Online]. Available: https://www.json.org/json-en.html. [Geopend 13 juni 2023].

[21] Protobuf , „Protocol Buffers,” [Online]. Available: https://protobuf.dev/. [Geopend 29 maart 2023].

[22] ParTech Media, „What is protobuf?,” Partech, 11 oktober 2021. [Online]. Available: https://www.partech.nl/nl/publicaties/2021/10/what-is-protobuf#. [Geopend 29 maart 2023].

[23] mgravell, „protobuf-net,” GitHub, [Online]. Available: https://github.com/protobuf-net/protobuf-net. [Geopend 29 maart 2023].

[24] Wikipedia, „Deplex,” 12 mei 2023. [Online]. Available: https://en.wikipedia.org/wiki/Duplex_(telecommunications). [Geopend 13 juni 2023].

[25] A. Jain, R. Bansal, A. Kumar en K. Singh, „A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students,” National Insitute of Health, mei-augustus 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4456887/. [Geopend 13 juni 2023].

[26] M. e. al, „Code metrics values,” Microsoft, 10 Oktober 2022. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/code-quality/code-metrics-values?view=vs-2022. [Geopend 13 juni 2023].

[27] A. H. Watson en T. J. McCabe, „Structured Testing: A Testing Methodology,” NIST, Gaithersburg, 1996.

[28] Mikejo5000, ghogen en pritamso, „Code metrics - Class coupling,” Microsoft, 30 april 2022. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/code-quality/code-metrics-class-coupling?view=vs-2022. [Geopend 13 juni 2023].

[29] Microsoft , „Form.TopMost Property,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.form.topmost?view=windowsdesktop-7.0. [Geopend 2023 juni 2].

[30] Chen, „What if I have two programs that are logically a single application, and I want them to be treated as a single group on the taskbar?,” Microsoft, 10 augustus 2015. [Online]. Available: https://devblogs.microsoft.com/oldnewthing/201508. [Geopend 2 juni 2023].

[31] Mircrosoft, „Control.Enabled property,” [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/system.windows.forms.control.enabled?view=windowsdesktop-7.0. [Geopend 14 juni 2023].

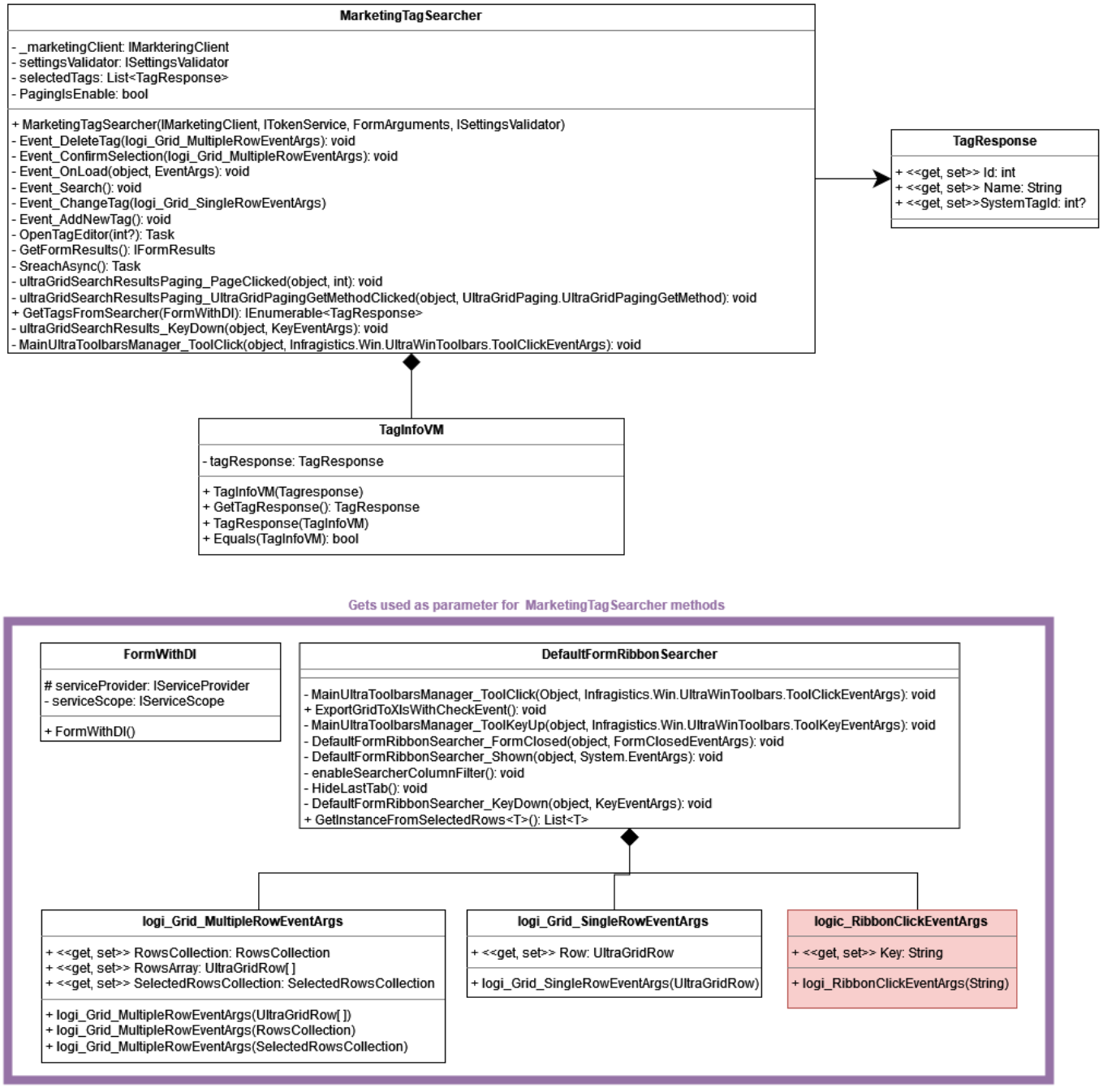
[32] g. e. al, „How to: Create and remove project dependencies,” Microsoft, 10 maart 2023. [Online]. Available: https://learn.microsoft.com/en-us/visualstudio/ide/how-to-create-and-remove-project-dependencies?view=vs-2022. [Geopend 6 juni 2023].

[33] Flutter, „Pipe class,” [Online]. Available: https://api.flutter.dev/flutter/dart-io/Pipe-class.html. [Geopend 6 juni 2023].

[34] L. Benders, „Reflecteren met de STARR-methode | Stappenplan + voorbeelden,” Scribbr, 4 april 2023. [Online]. Available: https://www.scribbr.nl/stage/starr-methode/. [Geopend 15 06 2023].

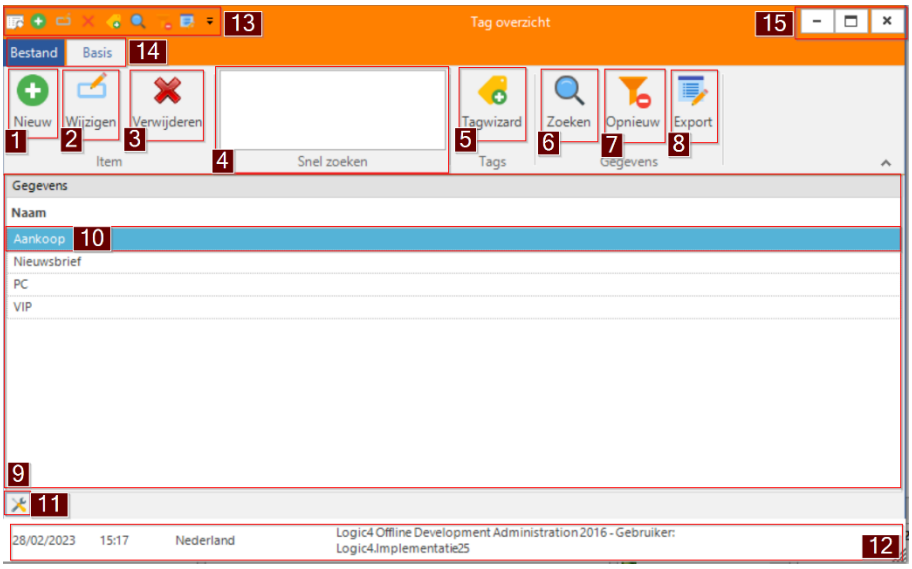
17 Bijlagen

Bijlage A. Class diagram MarketingTagSearcher



Bijlage B. Overzicht scherm knoppen

De mogelijke acties die kunnen worden uitgevoerd op het scherm. Op de screenshot hieronder (figuur 3) zijn nummers gezet waarmee de onderdelen worden omschreven.



1: Een knop om een nieuwe tag te maken, dit opent een nieuwe pagina.

2: Een knop om een geselecteerde tag te wijzigen. Dit lijkt een Tag aanmaken pagina te openen waarbij data al is ingevuld. (Boven in de pagina staat “Tag aanmaken”.

3: Een knop om een tag te verwijderen. Dit opent een pop-up die om bevestiging vraagt.

4: Hier kan een tag gezocht worden door er een woord in te zetten en om enter te drukken of door op de zoek knop te drukken (nummer 6)

5: Opent de tagwizard, hiermee kan een tag worden aangemaakt met behulp van de wizard.

6: Wanneer er iets is ingevuld bij vak 4 dan kan er op worden gezocht door op deze knop te klikken.

7: Haalt alle zoekresultaten weg, je ziet dan geen tags.

8: Dit exporteert de tags naar een Excel bestand die direct wordt geopend.

9: Hier staan alle tags gesorteerd op naam.

10: Dit is een geselecteerde tag.

11: Een settingsknop, hiermee kan het aantal resultaten geselecteerd worden.

12: Een vak met alle algemene informatie.

13: Een vak met in het klein dezelfde knoppen als 1, 2, 3, 4, 5, 6, 7, 8

14: Een manier om naar de bestand instellingen te gaan zie “bijlage B Afbeelding Tag overzicht bestand”.

15: Hiermee kan het scherm worden geminimaliseerd, het volledige beeld laten vullen en worden gesloten.

Bijlage C. Classdiagram schermen

Original program

Going to be the new program

Main

+ OpenMainMenuFromsInThread()

MarketingTagSearcher

- _marketingClient: IMarkteringClient
- settingsValidator: ISettingsValidator
- selectedTags: List<TagResponse>
- PagingIsEnable: bool

+ MarketingTagSearcher(IMarketingClient, ITokenService, FormArguments, ISettingsValidator)
- Event_DeleteTag(logi_Grid_MultipleRowEventArgs): void
- Event_ConfirmSelection(logi_Grid_MultipleRowEventArgs): void
- Event_OnLoad(object, EventArgs): void
- Event_Search(): void
- Event_ChangeTag(logi_Grid_SingleRowEventArgs)
- Event_AddNewTag(): void
- OpenTagEditor(int?): Task
- GetFormResults(): IFormResults
- SreachAsync(): Task
- ultraGridSearchResultsPaging_PageClicked(object, int): void
- ultraGridSearchResultsPaging_UltraGridPagingGetMethodClicked(object, UltraGridPaging.UltraGridPagingGetMethod): void
+ GetTagsFromSearcher(FormWithDI): IEnumerable<TagResponse>
- ultraGridSearchResults_KeyDown(object, KeyEventArgs): void
- MainUltraToolbarsManager_ToolClick(object, Infragistics.Win.UltraWinToolbars.ToolClickEventArgs): void

<<Designer>>
MarketingTagSearcher

- components: System.ComponentModel.IContainer

Dispose(bool): void
- InitializeComponent(): void

Original program

MarketingTagEditor

+ MarketingTagEditor(IMarketingClient, FormArguments, ISettingsValidator)
# GEtFormResults(): IFormResults
- Event_DeleteTag(): void
- MarketingTagEditor_logi_SaveRequest(): void
- Save(): Task<bool>
- MarketingTagEditor_logi_SaveAndNewRequest(): void
- MarketingTagEditor_logi_SaveAndCloseRequest(): void
- MarketingTagEditor_Load(object, EventArgs): void
- SetTagControls(bool): void
- txtTagName_Validating(object, System.ComponentModel.CancelEventArgs): void

<<Designer>>
MarketingTagEditor

- components: System.ComponentModel.IContainer
- txtTagName: Infragistics.Win.UltraWinEditors.UltraTextEditor
- lblTagName: Infragistics.Win.Misc.UltraLabel
- ultraTabControl1: Infragistics.Win.UltraWinTabControl.UltraTabControl
- ultraTabSharedControlsPage1: Infragistics.Win.UltraWinTabControl.UltraTabSharedControlsPage
- ultraTabPageControl1: Infragistics.Win.UltraWinTabControl.UltraTabPageControl
- ugMarketingCustomers: Infragistics.Win.UltraWinGrid.UltraGrid
- ultraTabPageControl2: Infragistics.Win.UltraWinTabControl.UltraTabPageControl
- ugMarketingAudiences: Infragistics.Win.UltraWinGrid.UltraGrid
- groupBox1: System.Windows.Forms.GroupBox

Dispose(bool): void- InitalizaComponent(): void

MarketingTagSearcher

- _marketingClient: IMarkteringClient
- settingsValidator: ISettingsValidator
- selectedTags: List<TagResponse>
- PagingIsEnable: bool

+ MarketingTagSearcher(IMarketingClient, ITokenService, FormA
- Event_DeleteTag(logi_Grid_MultipleRowEventArgs): void

<<Designer>>
MarketingTagSearcher

- components: System.ComponentModel.IContainer

Dispose(bool): void
- InitializeComponent(): void

MarketingTagWizard

- marketingClient: IMarketingClient

+ MarketingTagWizard(IMarketingClient, Logic.TagWizardViewModel)
# OnLoad(EventArgs): void
# OnStarted(EventArgs): void
# GetWizardsStept(): IEnumerable<IWizardStepControl>

<<Designer>>
MarketingTagWizard

- components: System.ComponentModel.IContainer

Dispose(bool): void
- InitializeComponent(): void

Bijlage D. Test resultaten communicatie

run	Time Server net7 (ms)	Time send net 7 (ms)	Time Server frame (ms)	Time Send frame (ms)	Start time send message net 7	Time message received frame	Difference (ms)	Start time send message frame	Time message received net 7	Difference (ms)	Total (ms)
Sockets + Protocol Buffers											
1	129	98	543	25	15:25:41.194	15:25:41.432	238	15:25:43.285	15:25:43.327	42	1075
2	112	102	560	19	15:30:22.858	15:30:23.149	336	15:30:25.049	15:30:25.081	32	1161
3	108	116	539	14	15:32:24.687	15:32:24.958	271	15:32:26.829	15:32:26.854	25	1073
4	106	99	539	16	15:34:43.511	15:34:43.761	250	15:34:45.636	15:34:45.663	27	1037
5	90	116	542	20	15:38:22.676	15:38:22.937	261	15:38:23.688	15:38:23.731	43	1072
6	104	153	558	17	15:47:43.394	15:47:43.696	302	15:47:45.521	15:47:45.552	31	1165
7	109	116	544	12	15:49:09.065	15:49:09.333	283	15:49:11.028	15:49:11.050	22	1086
8	93	119	538	13	15:50:36.918	15:50:37.176	258	15:50:39.523	15:50:39.550	27	1048
9	119	101	533	17	15:51:48.205	15:51:48.454	249	15:51:50.427	15:51:50.456	29	1048
10	123	112	548	21	15:53:09.179	15:53:09.430	251	15:53:11.321	15:53:11.357	36	1091
Average	109,3	113,2	544,4	17,4			269,9			31,4	1085,6
Sockets + JSON											
1	75	68	530	40	16:06:49.319	16:06:49.545	226	16:06:51.308	16:06:51.365	57	996
2	98	74	533	32	16:09:52.807	16:09:53.022	215	16:09:55.090	16:09:55.133	43	995
3	84	74	540	28	16:11:35.820	16:11:36.50	230	16:11:38.175	16:11:38.213	38	994
4	93	49	531	27	16:14:23.355	16:14:23.519	164	16:14:25.420	16:14:25.456	36	900
5	83	62	532	34	16:18:06.802	16:18:07.013	211	16:18:09.88	16:18:09.133	45	967
6	79	67	543	40	16:19:35.477	16:19:35.700	223	16:19:38.484	16:19:38.537	53	1005
7	80	87	536	27	16:21:39.390	16:21:39.629	239	16:21:41.177	16:21:41.217	40	1009
8	82	81	534	35	16:23:22.503	16:23:22.732	229	16:23:25.007	16:23:25.056	49	1010
9	84	88	544	45	16:25:09.994	16:25:10.244	250	16:25:11.989	16:25:12.050	61	1072
10	81	66	537	36	16:27:09.719	16:27:09.947	228	16:27:11.991	16:27:12.041	50	998
Average	83,9	71,6	536	34,4			221,5			47,2	994,6

Named pipes + JSON											
1	58	72	261	33	16:32:17.181	16:32:17.414	233	16:32:19.657	16:32:19.703	46	703
2	60	60	98	31	16:35:39.701	16:35:39.913	212	16:35:41.658	16:35:41.702	44	505
3	59	58	100	28	16:37:14.067	16:37:14.311	244	16:37:15.937	16:37:15.974	37	526
4	58	62	114	28	16:39:00.687	16:39:00.916	229	16:39:03.383	16:39:03.422	39	530
5	61	71	108	36	16:43:26.207	16:43:26.429	122	16:43:28.398	16:43:28.446	48	446
6	62	61	148	27	16:44:57.939	16:44:58.196	257	16:45:00.095	16:45:00.133	38	593
7	51	55	113	37	19:14:15.973	19:14:16.184	211	19:14:18.347	19:14:18.394	47	514
8	58	50	55	30	19:16:06.414	19:16:06.744	330	19:16:08.442	19:16:08.481	39	562
9	51	92	121	38	19:17:35.011	19:17:36.308	297	19:17:37.204	19:17:37.251	47	646
10	56	81	85	30	19:19:38.423	19:19:38.635	212	19:19:43.187	19:19:43.227	40	504
Average	57,4	66,2	120,3	31,8			234,7			42,5	552,9
Named pipes + Protocol Buffers											
1	81	223	94	15	19:33:24.871	19:33:25.108	237	19:33:27.015	19:33:27.037	22	672
2	84	240	110	24	19:34:57.791	19:34:58.041	250	19:34:59.897	19:34:59.932	34	742
3	80	212	94	21	19:36:31.727	19:36:31.949	222	19:36:33.640	19:36:33.669	29	658
4	83	230	126	25	19:37:56.596	19:37:56.838	242	19:37:58.728	19:37:58.763	35	741
5	86	248	154	27	19:39:14.331	19:39:14.590	259	19:39:16.885	19:39:16.922	37	811
6	85	221	82	24	19:40:25.745	19:40:25.974	229	19:40:27.893	19:40:27.924	31	672
7	88	260	103	24	19:41:31.224	19:41:31.492	268	19:41:33.291	19:41:33.324	33	767
8	119	252	177	22	19:43:18.396	19:43:18.657	261	19:43:20.397	19:43:20.428	31	862
9	89	228	108	15	19:45:47.771	19:45:48.011	240	19:45:50.628	19:45:50.649	21	701
10	93	231	158	24	19:47:52.314	19:47:52.555	241	19:47:54.538	19:47:54.571	33	780
Average	88,8	234,5	120,6	22,1			244,9			30,6	740,6
Anonymous pipes + JSON											
1	14	97	5	8	13:18:35.444	13:18:35.589	145	13:18:36.618	13:18:36.651	33	302
2	19	41	4	22	13:23:43.269	13:23:43.413	144	13:23:44.897	13:23:44.929	32	262
3	13	46	4	24	13:26:10.189	13:26:10.368	179	13:26:12.416	13:26:12.449	33	299
4	12	44	4	21	13:41:08.644	13:41:08.784	140	13:41:09.750	13:41:09.784	34	255

5	12	38	8	21	13:44:05.222	13:44:05.352	130	13:44:06.590	13:44:06.620	30	239
6	12	57	3	24	13:48:08.726	13:48:08.922	196	13:48:10.202	13:48:10.237	35	327
7	17	41	3	21	13:50:23.496	13:50:23.648	152	13:50:24.554	13:50:24.585	31	255
8	13	42	6	23	13:53:35.076	13:53:35.213	137	13:53:36.875	13:53:36.911	36	257
9	14	56	6	22	13:56:19.620	13:56:19.776	156	13:56:21.167	13:56:21.199	32	286
10	11	55	4	21	14:02:58.064	14:02:58.256	192	14:02:59.174	14:02:59.207	33	316
Average	13,7	51,7	4,7	20,7			157,1			32,9	279,8
Anonoymous pipes + Protocol Buffers											
1	13	0	6	0	15:05:38.542	15:05:38.548	6	15:05:39.984	15:05:39.989	5	30
2	12	0	6	0	15:08:45.364	15:08:45.371	7	15:08:47.134	15:08:47.139	5	30
3	14	0	5	0	15:10:25.092	15:10:25.097	5	15:10:26.316	15:10:26.324	8	32
4	12	0	6	0	15:12:01.475	15:12:01.479	5	15:12:02.660	15:12:02.666	6	29
5	12	0	6	0	15:13:28.061	15:13:28.064	3	15:13:30.184	15:13:30.192	8	29
6	17	0	6	0	15:15:14.003	15:15:14.009	6	15:15:16.371	15:15:16.377	6	35
7	14	0	6	0	15:18:20.588	15:18:20.598	10	15:18:21.758	15:18:21.767	9	30
8	12	0	6	0	15:19:55.671	15:19:55.675	4	15:19:56.620	15:19:56.625	5	27
9	12	0	6	0	15:21:14.558	15:21:14.564	6	15:21:15.568	15:21:15.573	5	29
10	12	0	6	0	15:22:43.607	15:22:43.610	3	15:22:44.599	15:22:44.607	8	29
Average	13	0	5,9	0			5,5			6,5	30

Bijlage E. Test resultaten data format

Anonymous pipes + Protocol Buffers	Start time send message net 7	Time message received frame	Difference (ms)	Start time send message frame	Time message received net 7	Difference (ms)
1	16:01:14.353	16:01:14.360	7	16:01:15.196	16:01:15.203	7
2	16:01:16.014	16:01:16.016	2	16:01:16.785	16:01:16.793	8
3	16:01:17.557	16:01:17.558	1	16:01:18.332	16:01:18.339	7
4	16:01:19.101	16:01:19.108	7	16:01:19.801	16:01:19.806	5
5	16:01:20.627	16:01:20.629	2	16:01:21.342	16:01:21.349	7
6	16:01:22.004	16:01:22.006	2	16:01:22.645	16:01:22.648	3
7	16:01:23.197	16:01:23.199	2	16:01:23.757	16:01:23.762	5
8	16:01:24.315	16:01:24.318	3	16:01:24.886	16:01:24.890	4
9	16:01:25.520	16:01:25.523	3	16:01:26.058	16:01:26.062	4
10	16:01:26.720	16:01:26.722	2	16:01:27.260	16:01:27.264	4
11	16:01:27.897	16:01:27.900	3	16:01:28.504	16:01:28.508	4
12	16:01:29.116	16:01:29.118	2	16:01:29.681	16:01:29.685	4
13	16:01:30.317	16:01:30.320	3	16:01:30.924	16:01:30.927	3
14	16:01:31.742	16:01:31.744	2	16:01:32.222	16:01:32.226	4
15	16:01:33.009	16:01:33.013	4	16:01:33.503	16:01:33.506	3
Average			3			4,8
Anonymous pipes + JSON	Start time send message net 7	Time message received frame	Difference (ms)	Start time send message frame	Time message received net 7	Difference (ms)
1	16:22:44.605	16:22:44.646	41	16:22:45.522	16:22:45.525	3
2	16:22:46.147	16:22:46.151	4	16:22:46.865	16:22:46.868	3
3	16:22:47.365	16:22:47.369	4	16:22:48.041	16:22:48.045	4
4	16:22:48.575	16:22:48.579	4	16:22:49.152	16:22:49.154	2
5	16:22:49.643	16:22:49.647	4	16:22:50.261	16:22:50.264	3
6	16:22:50.752	16:22:50.755	3	16:22:51.347	16:22:51.348	1
7	16:22:51.871	16:22:51.874	3	16:22:52.423	16:22:52.426	3
8	16:22:52.872	16:22:52.876	4	16:22:53.422	16:22:53.425	3
9	16:22:53.875	16:22:53.878	3	16:22:54.421	16:22:54.424	3
10	16:22:54.852	16:22:54.856	4	16:22:55.393	16:22:55.396	3
11	16:22:55.853	16:22:55.857	4	16:22:56.420	16:22:56.423	3
12	16:22:56.911	16:22:56.916	5	16:22:57.546	16:22:57.549	3
13	16:22:58.020	16:22:58.024	4	16:22:58.580	16:22:58.583	3
14	16:22:59.089	16:22:59.091	2	16:22:59.673	16:22:59.675	2
15	16:23:00.191	16:23:00.195	4	16:23:00.759	16:23:00.761	2
Average			6,2			2,733333

Bijlage F.Test resultaten community

resultaten	Google	Bing	Yahoo!	Stack Overflow
Named pipes	2480000	252000	252000	500
Sockets	4400000	289000	287000	500
Anonymous Pipes	2050000	250000	160000	56

milliseconden	Google	Bing	Yahoo!	Stack Overflow
JSON	454000000	595000000	400000000	355600
Protocol buffers	27300000	821000	294000	500

(27-04-2023)

Bijlage G. Test resultaten codecomplexiteit via visual studio

Scope	Type	Member	Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code
Assembly			84	105	2	44	702	238
Namespace			84	105	2	44	702	238
Type	BaseMessage		87	1	1	3	5	4
Member	BaseMessage	MessageType : int	100	1		0	1	0
Member	BaseMessage	MessageType.get() : int	100	1		0	1	0
Type	JSONThings		94	2	1	2	13	2
Member	JSONThings	toJSON(string) : string	96	1		2	4	1
Member	JSONThings	fromJSONString(string) : string	96	1		2	4	1
Type	MessageButton		93	3	2	3	11	5
Member	MessageButton	message_content : string	100	2		1	2	2
Member	MessageButton	message_content.get() : string	100	1		0	1	0
Member	MessageButton	message_content.set(string) : void	100	1		0	1	0
Member	MessageButton	MessageType : int	100	1		0	4	1
Member	MessageButton	MessageType.get() : int	100	1		0	1	1
Type	PipeServer		100	1	1	0	507	0
Type	PipeServer.Anonymous-PipeCLientJson		77	11	1	7	67	23
Member	PipeServer.Anonymous-PipeCLientJson	MessageReceived : EventHandler<string>	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientJson	pipeClientIn : AnonymousPipeClientStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientJson	pipeClientOut : AnonymousPipeClientStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientJson	ss : PipeServer.StreamString	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientJson	IsConnected : bool	100	1		0	1	0
Member	PipeServer.Anonymous-	IsConnected.get() : bool	100	1		0	1	0

	PipeCLientJson							
Member	PipeServer.Anonymous-PipeCLientJson	AnonymousPipeCLientJson(string, string)	82	1		2	11	2
Member	PipeServer.Anonymous-PipeCLientJson	connect() : void	58	5		6	27	14
Member	PipeServer.Anonymous-PipeCLientJson	Send(string) : Task	79	1		4	7	3
Member	PipeServer.Anonymous-PipeCLientJson	Stop() : void	76	3		1	11	4
Type	PipeServer.Anonymous-PipeCLientProto<T>		76	12	1	11	67	23
Member	PipeServer.Anonymous-PipeCLientProto<T>	MessageReceived : EventHandler<T>	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientProto<T>	pipeClientIn : AnonymousPipeClientStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientProto<T>	pipeClientOut : AnonymousPipeClientStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientProto<T>	ss : PipeServer.StreamString	100	0		1	1	0
Member	PipeServer.Anonymous-PipeCLientProto<T>	IsConnected : bool	100	1		0	1	0
Member	PipeServer.Anonymous-PipeCLientProto<T>	IsConnected.get() : bool	100	1		0	1	0
Member	PipeServer.Anonymous-PipeCLientProto<T>	AnonymousPipeCLientProto(string, string)	82	1		2	11	2
Member	PipeServer.Anonymous-PipeCLientProto<T>	connect() : void	57	6		9	28	15
Member	PipeServer.Anonymous-PipeCLientProto<T>	Send(T) : Task	85	1		5	6	2
Member	PipeServer.Anonymous-PipeCLientProto<T>	Stop() : void	76	3		1	11	4
Type	PipeServer.Anonymous-PipeReceiverJson		68	9	1	10	68	31
Member	PipeServer.Anonymous-PipeReceiverJson	MessageReceived : EventHandler<string>	100	0		1	1	0
Member	PipeServer.Anonymous-PipeReceiverJson	ansIn : AnonymousPipeServerStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeReceiverJson	ansOut : AnonymousPipeServerStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeReceiverJson	stream : bool	93	0		0	1	1
Member	PipeServer.Anonymous-PipeReceiverJson	Start(string) : Task	50	5		10	39	23

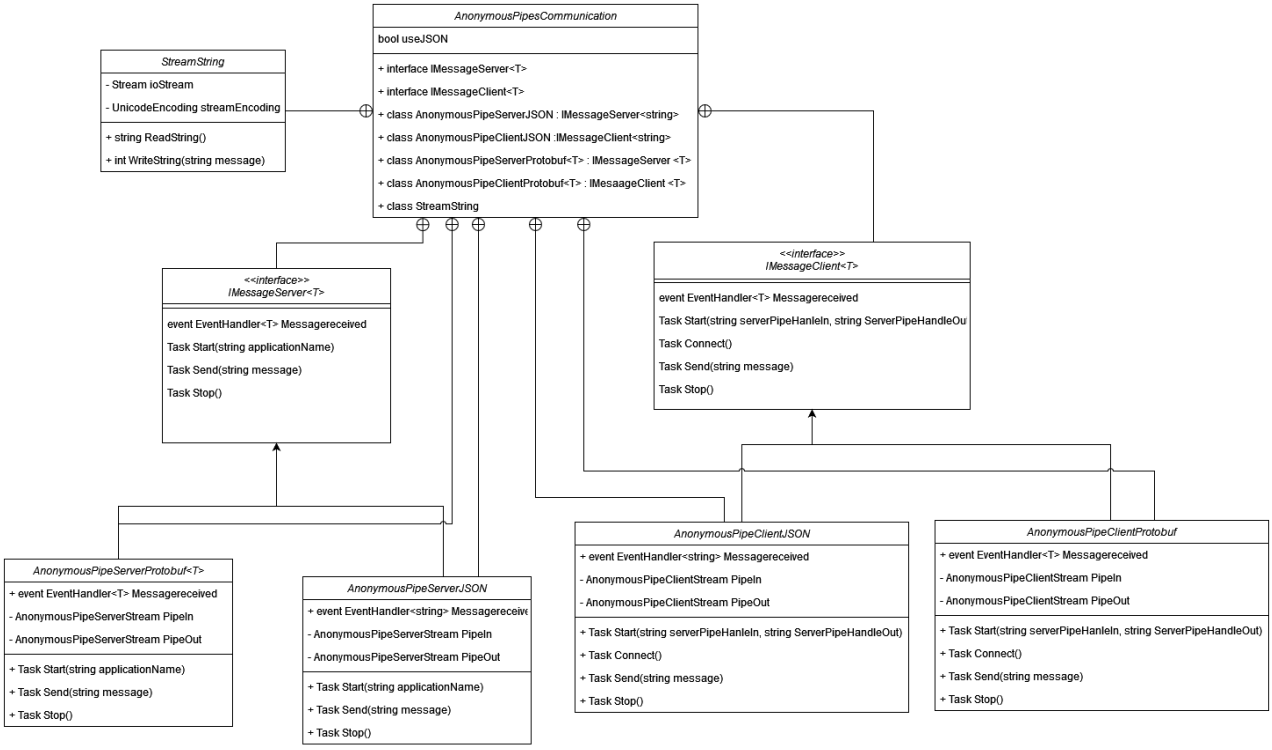
Member	PipeServer.Anonymous-PipeReceiverJson	Send(string) : Task	79	1		4	7	3
Member	PipeServer.Anonymous-PipeReceiverJson	Stop() : void	76	3		1	11	4
Type	PipeServer.Anonymous-PipeReceiverProto<T>		67	10	1	13	70	31
Member	PipeServer.Anonymous-PipeReceiverProto<T>	MessageReceived : EventHandler<T>	100	0		1	1	0
Member	PipeServer.Anonymous-PipeReceiverProto<T>	ansIn : AnonymousPipeServerStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeReceiverProto<T>	ansOut : AnonymousPipeServerStream	100	0		1	1	0
Member	PipeServer.Anonymous-PipeReceiverProto<T>	stream : bool	93	0		0	1	1
Member	PipeServer.Anonymous-PipeReceiverProto<T>	Start(string) : Task	50	6		13	42	24
Member	PipeServer.Anonymous-PipeReceiverProto<T>	Send(T) : Task	85	1		5	6	2
Member	PipeServer.Anonymous-PipeReceiverProto<T>	Stop() : void	76	3		1	11	4
Type	PipeServer.IMessageReceiver<T>		100	3	0	2	9	0
Member	PipeServer.IMessageReceiver<T>	hasStream() : bool	100	1		0	1	0
Member	PipeServer.IMessageReceiver<T>	Start(string) : Task	100	1		1	1	0
Member	PipeServer.IMessageReceiver<T>	MessageReceived : EventHandler<T>	100	0		1	1	0
Member	PipeServer.IMessageReceiver<T>	Stop() : void	100	1		0	1	0
Type	PipeServer.IMessageSender<T>		100	4	0	2	7	0
Member	PipeServer.IMessageSender<T>	IsConnected : bool	100	1		0	1	0
Member	PipeServer.IMessageSender<T>	IsConnected.get() : bool	100	1		0	1	0
Member	PipeServer.IMessageSender<T>	ConnectAsync(string) : Task	100	1		1	1	0
Member	PipeServer.IMessageSender<T>	Send(T) : Task	100	1		1	1	0
Member	PipeServer.IMessageSender<T>	GetStream() : Stream	100	1		1	1	0
Type	PipeServer.IStreamProvider		100	1	0	1	5	0
Member	PipeServer.IStreamProvider	GetStream() : Stream	100	1		1	1	0
Type	PipeServer.Named-PipeCLientJson		82	6	1	10	41	13
Member	PipeServer.Named-PipeCLientJson	pipeClient : NamedPipeClientStream	100	0		1	1	0

Member	PipeServer.Named-PipeCLientJson	IsConnected : bool	100	1		0	1	0
Member	PipeServer.Named-PipeCLientJson	IsConnected.get() : bool	100	1		0	1	0
Member	PipeServer.Named-PipeCLientJson	NamedPipeCLientJson(string)	90	1		3	6	1
Member	PipeServer.Named-PipeCLientJson	ConnectAsync(string) : Task	77	1		4	7	4
Member	PipeServer.Named-PipeCLientJson	Send(string) : Task	79	1		4	7	3
Member	PipeServer.Named-PipeCLientJson	GetStream() : Stream	100	1		2	4	1
Member	PipeServer.Named-PipeCLientJson	SendFile(string) : Task	74	1		4	8	4
Type	PipeServer.Named-PipeCLientProtobuf<T>		91	5	1	8	30	5
Member	PipeServer.Named-PipeCLientProtobuf<T>	pipeClient : NamedPipeClientStream	100	0		1	1	0
Member	PipeServer.Named-PipeCLientProtobuf<T>	IsConnected : bool	100	1		0	1	0
Member	PipeServer.Named-PipeCLientProtobuf<T>	IsConnected.get() : bool	100	1		0	1	0
Member	PipeServer.Named-PipeCLientProtobuf<T>	NamedPipeCLientProtobuf(string)	90	1		3	7	1
Member	PipeServer.Named-PipeCLientProtobuf<T>	ConnectAsync(string) : Task	95	1		2	5	1
Member	PipeServer.Named-PipeCLientProtobuf<T>	Send(T) : Task	85	1		5	6	2
Member	PipeServer.Named-PipeCLientProtobuf<T>	GetStream() : Stream	100	1		2	4	1
Type	PipeServer.Named-PipeReceiverJson		78	6	1	8	36	14
Member	PipeServer.Named-PipeReceiverJson	MessageReceived : EventHandler<string>	100	0		1	1	0
Member	PipeServer.Named-PipeReceiverJson	ns : NamedPipeServerStream	100	0		1	1	0
Member	PipeServer.Named-PipeReceiverJson	stream : bool	93	0		0	1	1
Member	PipeServer.Named-PipeReceiverJson	Start(string) : Task	63	3		7	15	10
Member	PipeServer.Named-PipeReceiverJson	hasStream() : bool	100	1		0	4	1
Member	PipeServer.Named-PipeReceiverJson	Stop() : void	86	2		1	7	2

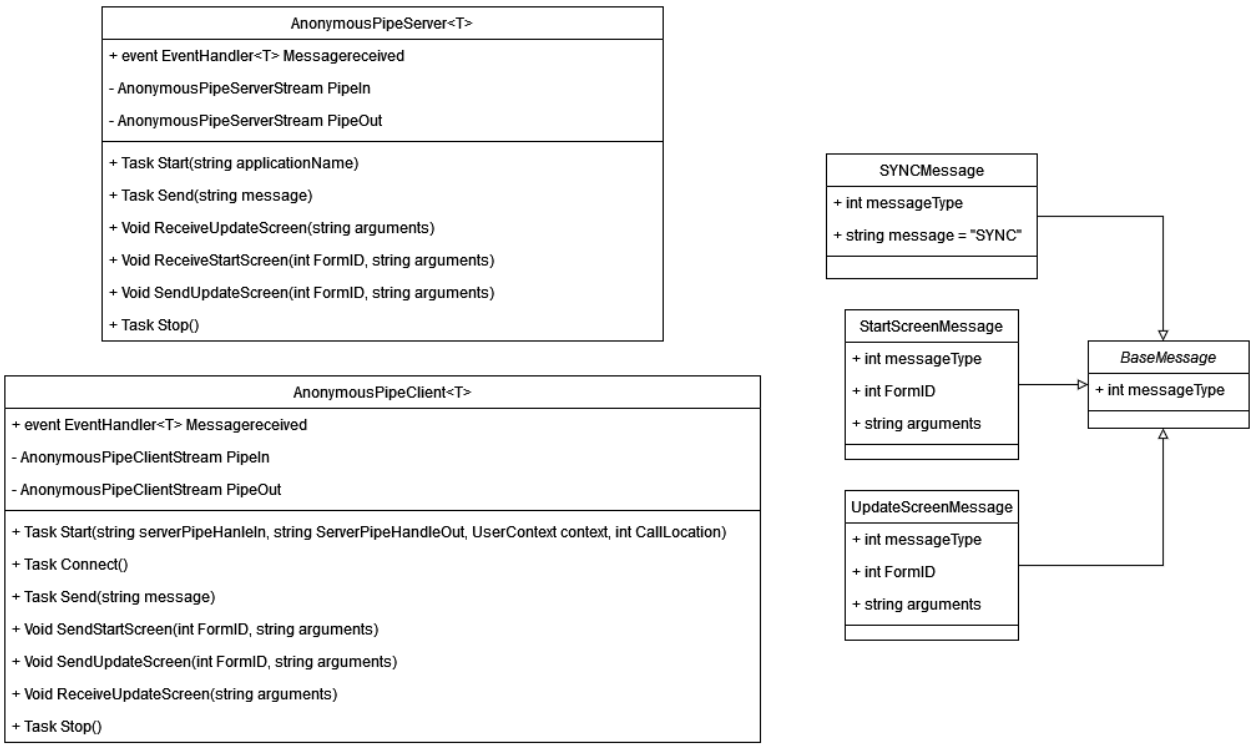
Type	PipeServer.Named-PipeReceiverProtobuf<T>		83	6	1	9	42	13
Member	PipeServer.Named-PipeReceiverProtobuf<T>	MessageReceived : EventHandler<T>	100	0		1	1	0
Member	PipeServer.Named-PipeReceiverProtobuf<T>	ns : NamedPipeServerStream	100	0		1	1	0
Member	PipeServer.Named-PipeReceiverProtobuf<T>	stream : bool	93	0		0	1	1
Member	PipeServer.Named-PipeReceiverProtobuf<T>	Start(string) : Task	64	3		8	22	9
Member	PipeServer.Named-PipeReceiverProtobuf<T>	hasStream() : bool	100	1		0	1	1
Member	PipeServer.Named-PipeReceiverProtobuf<T>	Stop() : void	86	2		1	7	2
Type	PipeServer.Named-PipeReceiverProtobuf<T>		70	4	1	4	51	21
Member	PipeServer.StreamString	ioStream : Stream	100	0		1	1	0
Member	PipeServer.StreamString	streamEncoding : UnicodeEncoding	100	0		1	1	0
Member	PipeServer.StreamString	StreamString(Stream)	85	1		2	5	2
Member	PipeServer.StreamString	ReadString() : string	69	1		3	13	6
Member	PipeServer.StreamString	ReadStringAsync() : Task<string>	68	1		4	13	6
Member	PipeServer.StreamString	WriteString(string) : int	66	1		3	11	7
Type	Sockets		100	1	1	0	153	0
Type	Sockets.SocketClientJson		81	5	1	14	35	12
Member	Sockets.SocketClientJson	stream : Stream	100	0		1	1	0
Member	Sockets.SocketClientJson	IsConnected : bool	100	1		0	1	0
Member	Sockets.SocketClientJson	IsConnected.get() : bool	100	1		0	1	0
Member	Sockets.SocketClientJson	ConnectAsync(string) : Task	66	1		11	11	7
Member	Sockets.SocketClientJson	Send(string) : Task	79	1		3	6	3
Member	Sockets.SocketClientJson	SendFile(string) : Task	100	1		1	4	1
Member	Sockets.SocketClientJson	GetStream() : Stream	100	1		1	4	1
Type	Sockets.SocketClientProtobuf<T>		82	5	1	15	34	11
Member	Sockets.SocketClientProtobuf<T>	stream : Stream	100	0		1	1	0
Member	Sockets.SocketClientProtobuf<T>	IsConnected : bool	100	1		0	1	0
Member	Sockets.SocketClientProtobuf<T>	IsConnected.get() : bool	100	1		0	1	0

Member	Sockets.SocketClientProtobuf<T>	ConnectAsync(string) : Task	66	1		11	11	7
Member	Sockets.SocketClientProtobuf<T>	Send(T) : Task	85	1		4	5	2
Member	Sockets.SocketClientProtobuf<T>	SendFile(string) : Task	100	1		1	4	1
Member	Sockets.SocketClientProtobuf<T>	GetStream() : Stream	100	1		1	4	1
Type	Sockets.SocketServerJson		73	5	1	19	39	16
Member	Sockets.SocketServerJson	tcpListener : TcpListener	100	0		1	1	0
Member	Sockets.SocketServerJson	MessageReceived : EventHandler<string>	100	0		1	1	0
Member	Sockets.SocketServerJson	Start(string) : Task	57	3		17	21	14
Member	Sockets.SocketServerJson	hasStream() : bool	100	1		0	4	1
Member	Sockets.SocketServerJson	Stop() : void	100	1		1	4	1
Type	Sockets.SocketServerProtobuf<T>		76	5	1	19	37	14
Member	Sockets.SocketServerProtobuf<T>	tcpListener : TcpListener	100	0		1	1	0
Member	Sockets.SocketServerProtobuf<T>	MessageReceived : EventHandler<T>	100	0		1	1	0
Member	Sockets.SocketServerProtobuf<T>	Start(string) : Task	59	3		17	19	12
Member	Sockets.SocketServerProtobuf<T>	hasStream() : bool	100	1		0	4	1
Member	Sockets.SocketServerProtobuf<T>	Stop() : void	100	1		1	4	1

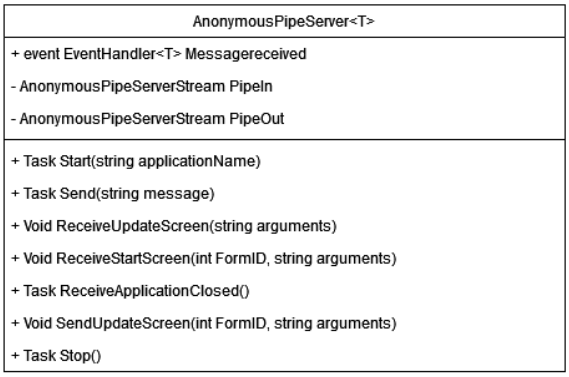
Bijlage H. Verschillende versies anonymousPipes

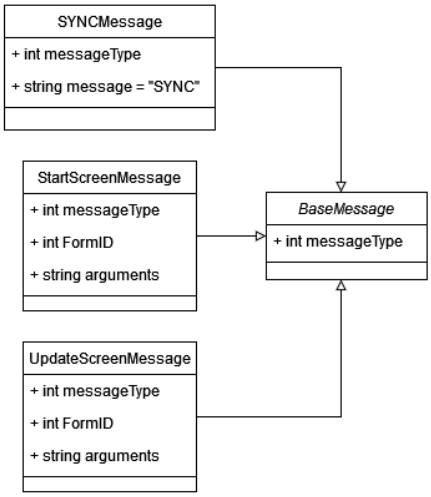


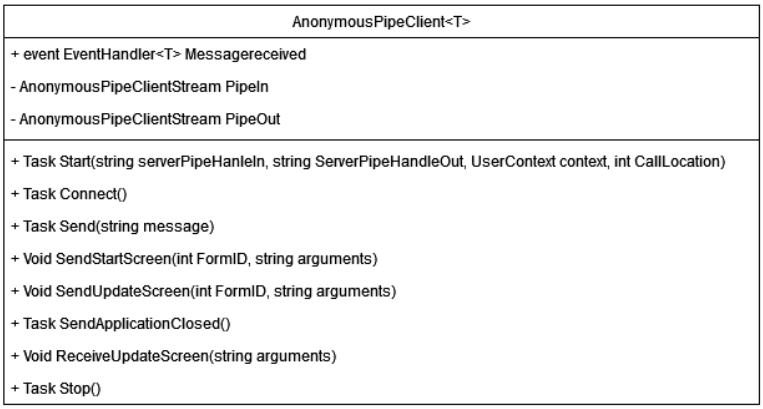
Versie 1.0



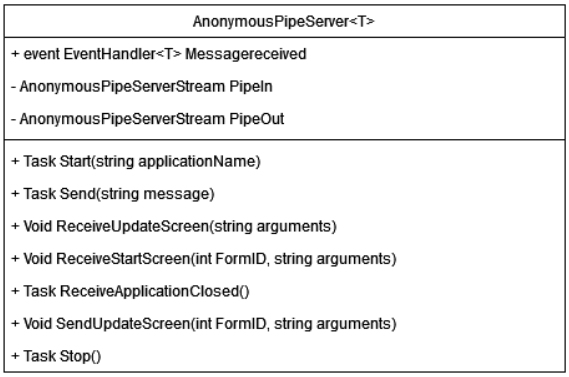
Versie 2.0

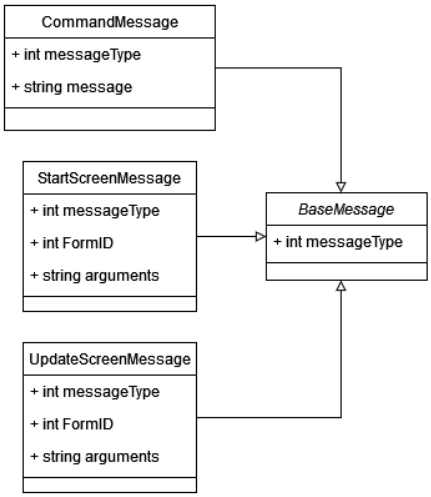

AnonymousPipeServer<T>
+ event EventHandler<T> Messagereceived
- AnonymousPipeServerStream PipeIn
- AnonymousPipeServerStream PipeOut
+ Task Start(string applicationName)
+ Task Send(string message)
+ Void ReceiveUpdateScreen(string arguments)
+ Void ReceiveStartScreen(int FormID, string arguments)
+ Task ReceiveApplicationClosed()
+ Void SendUpdateScreen(int FormID, string arguments)
+ Task Stop()

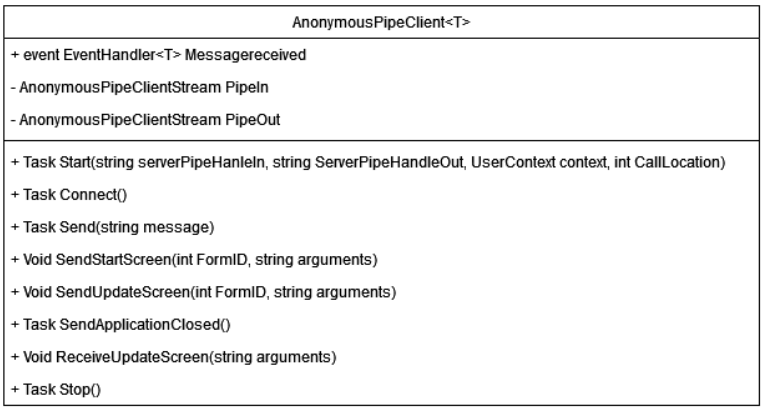

SYNCMessage
+ int messageType
+ string message = "SYNC"
StartScreenMessage
+ int messageType
+ int FormID
+ string arguments
BaseMessage
+ int messageType
UpdateScreenMessage
+ int messageType
+ int FormID
+ string arguments


AnonymousPipeClient<T>
+ event EventHandler<T> Messagereceived
- AnonymousPipeClientStream PipeIn
- AnonymousPipeClientStream PipeOut
+ Task Start(string serverPipeHanleIn, string ServerPipeHandleOut, UserContext context, int CallLocation)
+ Task Connect()
+ Task Send(string message)
+ Void SendStartScreen(int FormID, string arguments)
+ Void SendUpdateScreen(int FormID, string arguments)
+ Task SendApplicationClosed()
+ Void ReceiveUpdateScreen(string arguments)
+ Task Stop()

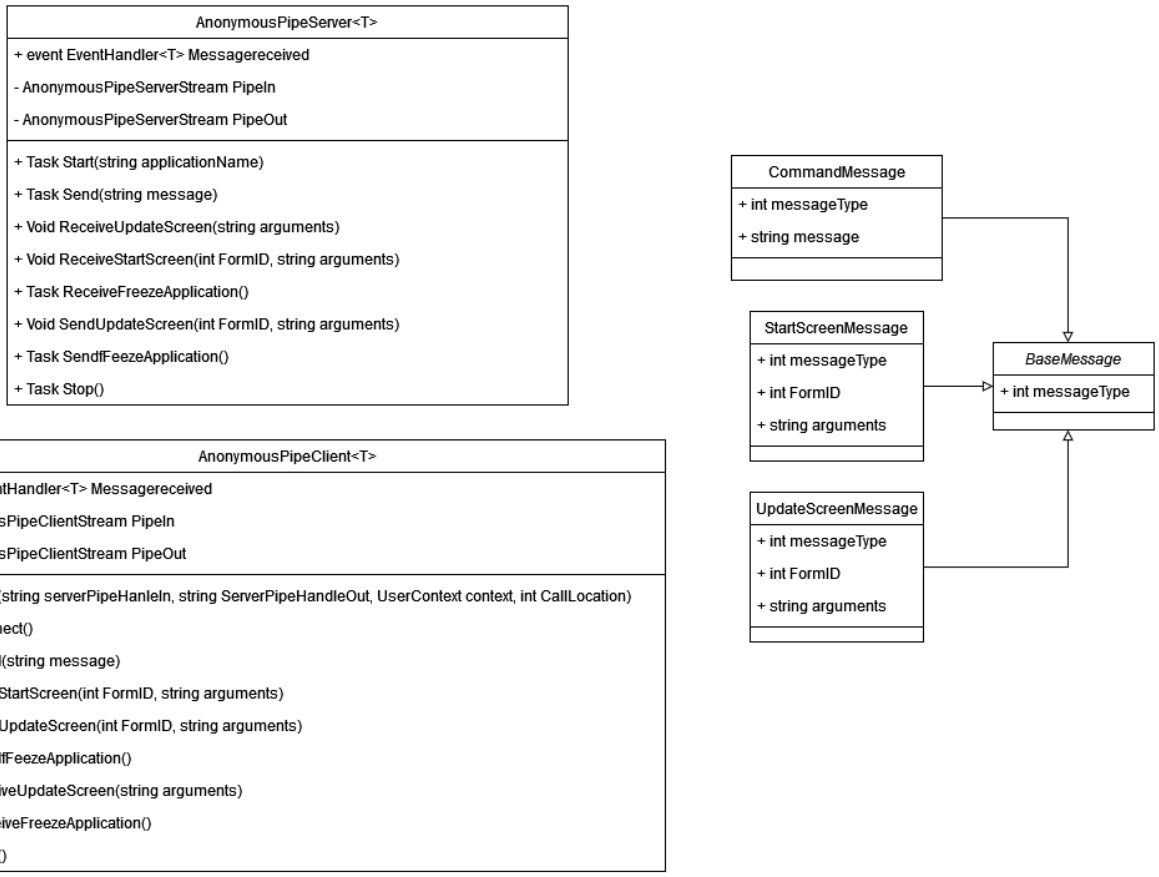
Versie 3


AnonymousPipeServer<T>
+ event EventHandler<T> Messagereceived
- AnonymousPipeServerStream PipeIn
- AnonymousPipeServerStream PipeOut
+ Task Start(string applicationName)
+ Task Send(string message)
+ Void ReceiveUpdateScreen(string arguments)
+ Void ReceiveStartScreen(int FormID, string arguments)
+ Task ReceiveApplicationClosed()
+ Void SendUpdateScreen(int FormID, string arguments)
+ Task Stop()

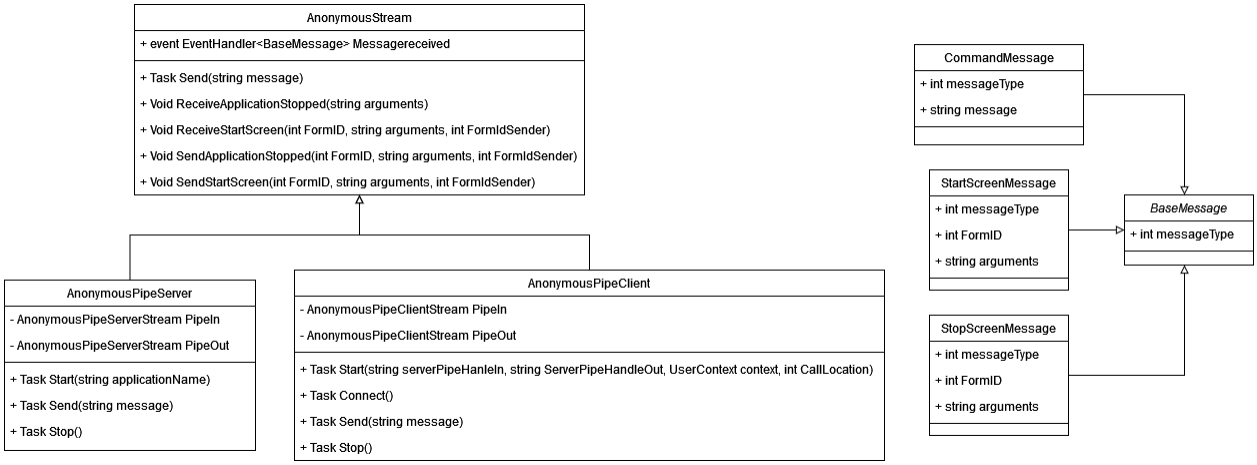

CommandMessage
+ int messageType
+ string message
StartScreenMessage
+ int messageType
+ int FormID
+ string arguments
BaseMessage
+ int messageType
UpdateScreenMessage
+ int messageType
+ int FormID
+ string arguments


AnonymousPipeClient<T>
+ event EventHandler<T> Messagereceived
- AnonymousPipeClientStream PipeIn
- AnonymousPipeClientStream PipeOut
+ Task Start(string serverPipeHanleIn, string ServerPipeHandleOut, UserContext context, int CallLocation)
+ Task Connect()
+ Task Send(string message)
+ Void SendStartScreen(int FormID, string arguments)
+ Void SendUpdateScreen(int FormID, string arguments)
+ Task SendApplicationClosed()
+ Void ReceiveUpdateScreen(string arguments)
+ Task Stop()

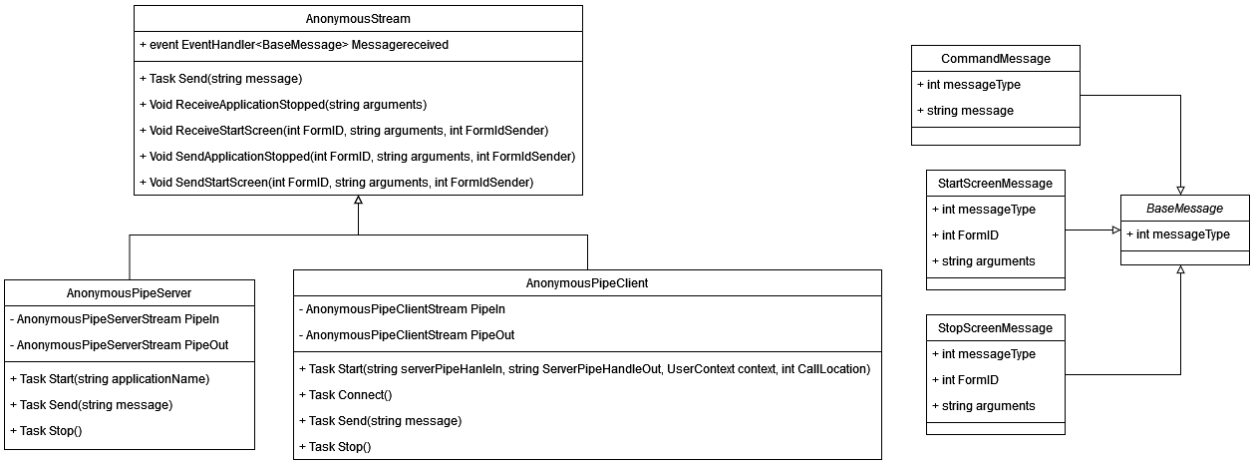
Versie 4



Versie 5

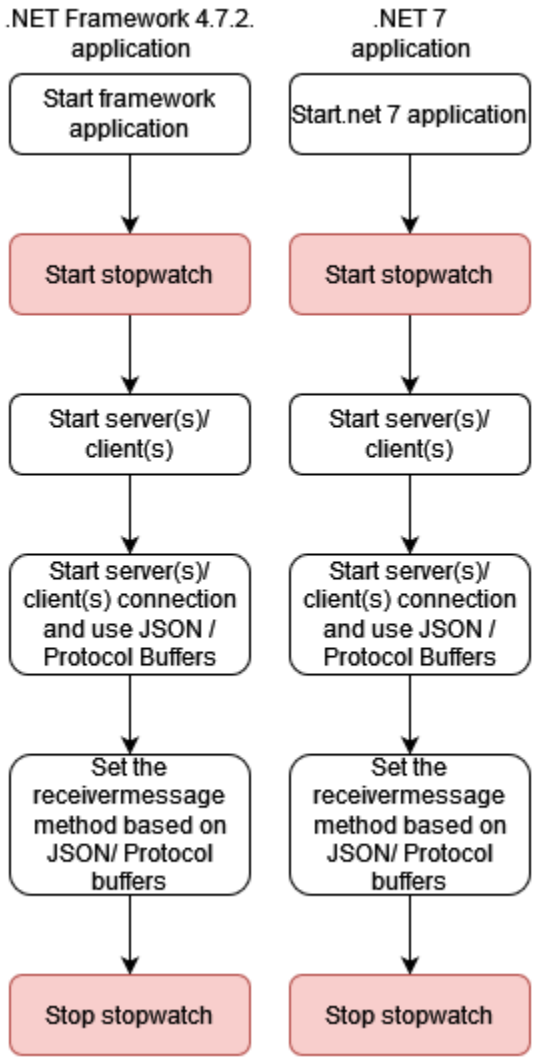


Versie 6

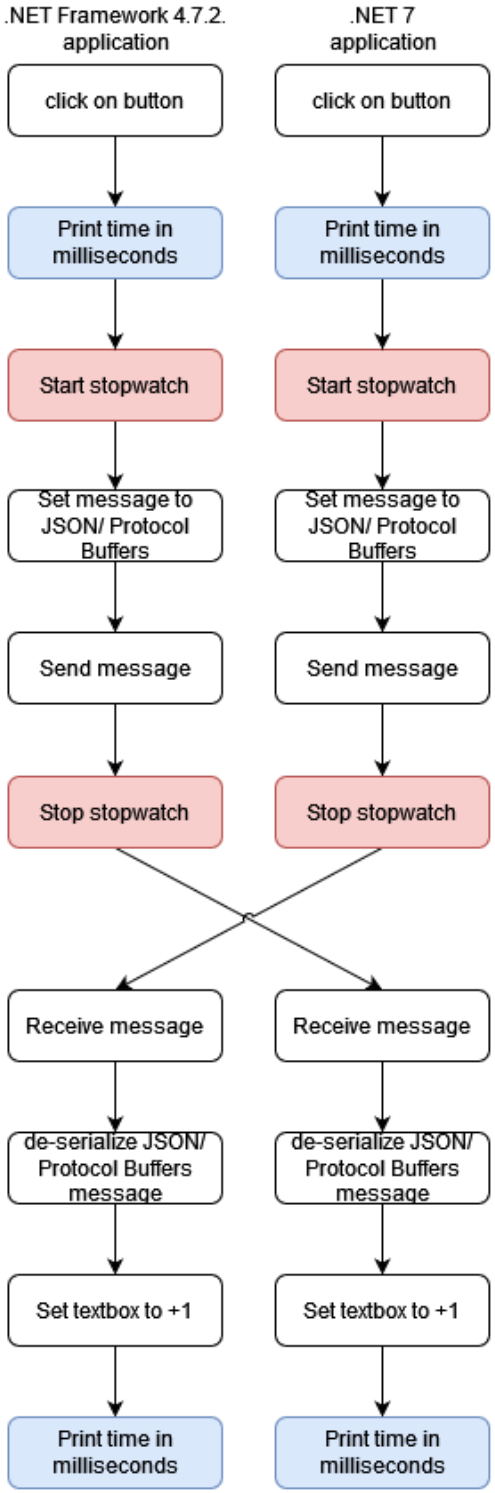

AnonymousStream
+ event EventHandler<BaseMessage> Messagereceived
+ Task Send(string message)
+ Void ReceiveApplicationStopped(string arguments)
+ Void ReceiveStartScreen(int FormID, string arguments, int FormIdSender)
+ Void SendApplicationStopped(int FormID, string arguments, int FormIdSender)
+ Void SendStartScreen(int FormID, string arguments, int FormIdSender)
AnonymousPipeServer
- AnonymousPipeServerStream PipeIn
- AnonymousPipeServerStream PipeOut
+ Task Start(string applicationName)
+ Task Send(string message)
+ Task Stop()
AnonymousPipeClient
- AnonymousPipeClientStream PipeIn
- AnonymousPipeClientStream PipeOut
+ Task Start(string serverPipeHanleIn, string ServerPipeHandleOut, UserContext context, int CallLocation)
+ Task Connect()
+ Task Send(string message)
+ Task Stop()
CommandMessage
+ int messageType
+ string message
StartScreenMessage
+ int messageType
+ int FormID
+ string arguments
BaseMessage
+ int messageType
StopScreenMessage
+ int messageType
+ int FormID
+ string arguments

Versie 7

Bijlage I. Flow test start + connect



Bijlage J. Flow test message tijd



Bijlage K. Wekelijkse planning