

Stage Ziekenhuis Geel Realisatie

Mathijs Ooms
Student Bachelor in de Toegepaste Informatica – Applicatieontwikkeling

Inhoudsopgave

1. INLEIDING	8
2. WACHTZAAL APPLICATIE	9
2.1. Analyse	9
2.1.1. Klant	9
2.1.2. Huidige situatie	9
2.1.3. Probleem	9
2.1.4. Functionele eisen	9
2.1.4.1. Use-cases	10
2.1.5. Technische eisen	11
2.1.5.1. Data-laag	12
2.1.5.2. Business-laag	13
2.1.5.3. Presentatie-laag	13
2.1.6. Technologieën	14
2.2. Realisatie	15
2.2.1. Doel	15
2.2.2. Functionaliteiten	15
2.2.3. 3-tier architectuur	15
2.2.3.1. Data-laag	15
2.2.3.2. Business-laag	18
2.2.3.3. Presentatie-laag	19
2.2.4. Uitdagingen	25
2.2.5. Eindresultaat	25
3. MAGAZIJN ETIKETTEN	26
3.1. Analyse	26
3.1.1. Klant	26
3.1.2. Huidige situatie	26
3.1.3. Probleem	26
3.1.4. Functionele eisen	26
3.1.4.1. Use cases	26
3.1.5. Technische eisen	27
3.1.5.1. Data-laag	27
3.1.5.2. Business-laag	27
3.1.5.3. Presentatie-laag	27
3.1.6. Technologieën	28
3.2. Realisatie	28
3.2.1. Doel	28
3.2.2. Functionaliteiten	28
3.2.3. 3-tier architectuur	28
3.2.3.1. Data-laag	29
3.2.3.2. Business-laag	29
3.2.3.3. Presentatie-laag	30
3.2.4. Uitdagingen	30
3.2.5. Eindresultaat	31

4. KEUKEN ETIKETTEN	32
4.1. Analyse	32
4.1.1. Klant	32
4.1.2. Huidige situatie	32
4.1.3. Probleem	32
4.1.4. Functionele eisen	32
4.1.4.1. Use cases	32
4.1.5. Technische eisen	33
4.1.5.1. Data-laag	33
4.1.5.2. Business-laag	33
4.1.5.3. Presentatie-laag	34
4.1.6. Technologieën	34
4.2. Realisatie	35
4.2.1. Doel	35
4.2.2. Functionaliteiten	35
4.2.3. 3-tier architectuur	35
4.2.3.1. Business-laag	35
4.2.3.2. Presentatie-laag	36
4.2.4. Uitdagingen	38
4.2.5. Eindresultaat	39
5. KLEDIJ UITLENINGEN	40
5.1. Analyse	40
5.1.1. Klant	40
5.1.2. Huidige situatie	40
5.1.3. Probleem	40
5.1.4. Functionele eisen	40
5.1.4.1. Use cases	40
5.1.5. Technische eisen	41
5.1.5.1. Data-laag	41
5.1.5.2. Business-laag	42
5.1.5.3. Presentatie-laag	42
5.1.6. Technologieën	43
5.2. Realisatie	44
5.2.1. Doel	44
5.2.2. Functionaliteiten	44
5.2.3. 3-tier architectuur	44
5.2.3.1. Data-laag	44
5.2.3.2. Business-laag	46
5.2.3.3. Presentatie-laag	51
5.2.4. Uitdagingen	53
5.2.5. Eindresultaat	53
6. BADGE UITLENINGEN	54
6.1. Analyse	54
6.1.1. Klant	54
6.1.2. Huidige situatie	54
6.1.3. Probleem	54

6.1.4. Functionele eisen	54
6.1.4.1. <i>Use cases</i>	54
6.1.5. Technische eisen	55
6.1.5.1. <i>Data-laag</i>	55
6.1.5.2. <i>Business-laag</i>	56
6.1.5.3. <i>Presentatie-laag</i>	56
6.1.6. Technologieën	57
6.2. Realisatie	58
6.2.1. Doel	58
6.2.2. Functionaliteiten	58
6.2.3. 3-tier architectuur	58
6.2.3.1. <i>Data-laag</i>	58
6.2.3.2. <i>Business-laag</i>	61
6.2.3.3. <i>Presentatie-laag</i>	63
6.2.4. Uitdagingen	67
6.2.5. Eindresultaat	67
7. DEVICE MANAGER	68
7.1. Analyse	68
7.1.1. Klant	68
7.1.2. Huidige situatie	68
7.1.3. Probleem	68
7.1.4. Functionele eisen	68
7.1.4.1. <i>Use cases</i>	68
7.1.5. Technische eisen	69
7.1.5.1. <i>Data-laag</i>	69
7.1.5.2. <i>Business-laag</i>	69
7.1.5.3. <i>Presentatie-laag</i>	69
7.1.6. Technologieën	70
7.2. Realisatie	70
7.2.1. Doel	70
7.2.2. Functionaliteiten	70
7.2.3. 3-tier architectuur	70
7.2.3.1. <i>Data-laag</i>	71
7.2.3.2. <i>Business-laag</i>	72
7.2.3.3. <i>Presentatie-laag</i>	73
7.2.4. Uitdagingen	77
7.2.5. Eindresultaat	77
8. BESLUIT	78
8.1. Samenvatting	78
8.2. Conclusies	78
8.3. Evaluatie	78
9. VERWIJZINGEN	79

Lijst van afbeeldingen

Afbeelding 1: Data UML Wachtzaal	12
Afbeelding 2: SQL database structuur	15
Afbeelding 3: Database tabel Parameters	16
Afbeelding 4: Database tabel Visits	16
Afbeelding 5: Database tabel VisitFases	16
Afbeelding 6: Database tabel VisitsLogging	17
Afbeelding 7: Data klasse Wachtzaal	17
Afbeelding 8: Object Patient	18
Afbeelding 9: Object Visit	18
Afbeelding 10: Startscherm Wachtzaal Nederlands	19
Afbeelding 11: Startscherm Wachtzaal Engels	19
Afbeelding 12: Startscherm Wachtzaal met statusbalk	20
Afbeelding 13: Meld aan bij inkom scherm	23
Afbeelding 14: Elektronisch aanvraagformulier gevonden scherm	20
Afbeelding 15: Wachtzaal scherm	21
Afbeelding 16: Geen elektronisch aanvraagformulier scherm	21
Afbeelding 17: Meld aan bij balie scherm	22
Afbeelding 18: Wachtzaal overzicht scherm	24
Afbeelding 19: Patiënt toevoegen scherm	24
Afbeelding 20: Voorbeeld magazijn etiket	27
Afbeelding 21: Object Etiket	29
Afbeelding 22: Startscherm Magazijn Etiketten	30
Afbeelding 23: Resultaat magazijn etiket	31
Afbeelding 24: Voorbeeld keuken etiket	34
Afbeelding 25: Object Bestelling	35
Afbeelding 26: Object Etiket	36
Afbeelding 27: Startscherm Keuken Etiketten (lege tabel)	36
Afbeelding 28: Excel importeren scherm	37
Afbeelding 29: Startscherm Keuken Etiketten (gevulde tabel)	37
Afbeelding 30: Etiket afdrukken scherm	38
Afbeelding 31: Foutief keuken etiket	38
Afbeelding 32: Resultaat keuken etiket	39
Afbeelding 33: Data klasse kledij uitleningen	41
Afbeelding 34: Mock-up overzicht scherm kledij uitleningen	42
Afbeelding 35: Mock-up nieuwe uitlening scherm	43
Afbeelding 36: Mock-up kledij inleveren scherm	43
Afbeelding 37: SQL database structuur	44
Afbeelding 38: Database tabel Uitleningen	45
Afbeelding 39: Database tabel KledingType	45

Afbeelding 40: Database tabel PersoneelType	45
Afbeelding 41: Database tabel PersoneelKleding	46
Afbeelding 42: Database tabel Logging	46
Afbeelding 43: Object Uitlening	47
Afbeelding 44: Object KledingType	48
Afbeelding 45Object PersoneelType	49
Afbeelding 46: Object PersoneelKleding	50
Afbeelding 47: Startscherf Kledij Uitleningen	51
Afbeelding 48: Nieuwe uitlening scherm	51
Afbeelding 49: Waarschuwing meerdere kledingstukken	51
Afbeelding 50: Open uitlening scherm	52
Afbeelding 51: Verwijder uitlening via scan scherm	52
Afbeelding 52: Bevestiging uitlening verwijderen	52
Afbeelding 53: Data UML Badge Uitleningen	55
Afbeelding 54: Overzicht scherm Badge Uitleningen configuratie	56
Afbeelding 55: Startscherf Badge Uitleningen	57
Afbeelding 56: Badge uitlenen scherm	57
Afbeelding 57: SQL database structuur	58
Afbeelding 58: Database tabel Badges	59
Afbeelding 59: Database tabel BadgeUitleningen	59
Afbeelding 60: Database tabel BadgeUitleningenLogging	59
Afbeelding 61: Data klasse Badge Uitleningen	60
Afbeelding 62: Object Badge	61
Afbeelding 63: Object BadgeUitlening	62
Afbeelding 64: Overzicht scherm (huidige situatie) Badge Uitleningen	63
Afbeelding 65: Overzicht scherm (geschiedenis) Badge Uitleningen	63
Afbeelding 66: Nieuwe badge pop-up	64
Afbeelding 67: Scan badge pop-up	64
Afbeelding 68: Open badge pop-up	64
Afbeelding 69: Startscherf web app	65
Afbeelding 70: Badge uitlenen scherm	65
Afbeelding 71: Startscherf met confirmatie van uitlening badge	66
Afbeelding 72: Startscherf met confirmatie van inlevering badge	66
Afbeelding 73: Data klasse MAC adres updater	71
Afbeelding 74: Data klasse Switchpoort updater	71
Afbeelding 75: Object Telefoon	72
Afbeelding 76: Object Switch	72
Afbeelding 77: Startscherf (lege tabel) MAC adres updater	73
Afbeelding 78: Excel importeren scherm	73
Afbeelding 79: Startscherf (gevulde tabel) MAC adres updater	74
Afbeelding 80: Bevestiging pop-up MAC adres updater	74
Afbeelding 81: Startscherf console applicatie MAC adres updater	75

Afbeelding 82: Update voltooid scherm	75
Afbeelding 83: Startscherm Switchpoort upda	76
Afbeelding 84: Update voltooid scherm	76
Afbeelding 85: Mail voorbeeld	77

1. Inleiding

In dit realisatielidmaatschap worden alle projecten die ik tijdens mijn stageperiode binnen het ziekenhuis heb uitgevoerd uitgebreid en stap voor stap toegelicht. Mijn stageperiode duurde van 22 september tot 19 december en binnen deze periode heb ik aan verschillende projecten gewerkt die elk een eigen analyse-, ontwikkelings- en testproces vereisten. Dit document richt zich op de volledige uitwerking van elk project en beschrijft hoe ik van de analyse tot het uiteindelijke eindresultaat ben gekomen.

Voor ieder project licht ik het volledige traject toe, beginnend bij het analyseren van de behoeften van de interne klant en het opstellen van use cases en technische specificaties. Vervolgens beschrijf ik hoe de applicatiestructuur werd bepaald, hoe de database en 3-tier architectuur werden opgebouwd en hoe de ontwikkeling stap voor stap werd uitgevoerd. Ook de grootste uitdagingen en de eindoplevering komen uitgebreid aan bod.

De projecten verschillen sterk in complexiteit en vereisten, waardoor ik met een brede waaier aan technologieën heb gewerkt zoals C#, Windows Forms, SQL Server, SAP-databases, Excel-verwerking, barcodescanners, eID-lezers en labelprinters. Elk project bracht specifieke uitdagingen met zich mee, wat mij de kans gaf om mijn technische, analytische en probleemoplossende vaardigheden verder te verdiepen.

Dit document biedt daarmee een helder en volledig overzicht van mijn aanpak, de gemaakte keuzes en de gerealiseerde oplossingen. Het vormt een concreet bewijs van mijn bijdrage aan de optimalisatie en digitalisering van interne processen binnen het ziekenhuis.

2. Wachtzaal applicatie

2.1. Analyse

2.1.1. Klant

De klant voor dit project is de dienst Laboratorium van het ziekenhuis. Deze dienst is verantwoordelijk voor het uitvoeren van bloedafnames en het onderzoeken en analyseren van bloed- en andere stalen. De medewerkers van het laboratorium werken dagelijks met een grote stroom patiënten en spelen een essentiële rol in het diagnostische proces binnen het ziekenhuis.

2.1.2. Huidige situatie

Op dit moment worden alle patiënten die het labo bezoeken ontvangen door de baliemedewerkers. Patiënten kunnen om drie redenen langskomen: het ophalen van materiaal zoals potjes voor stalen, het afgeven van stalen of het ondergaan van een bloedafname. Voor elke patiënt moeten de baliemedewerkers in het elektronisch patiëntendossier (HiX) alle relevante gegevens opzoeken en verwerken. Dit omvat onder meer het controleren van persoonlijke gegevens, het raadplegen van de medische voorgeschiedenis en het nakijken van specifieke instructies voor de afname of verwerking van stalen. Dit is een nauwkeurig en tijdsintensief proces dat veel aandacht vereist.

2.1.3. Probleem

De huidige werkwijze leidt regelmatig tot een hoge werkdruck aan de balie, vooral tijdens piekmomenten waarop meerdere patiënten tegelijk aankomen. Hierdoor ontstaan wachtrijen, wat zowel vertraging voor de patiënten als verhoogde stress voor het personeel veroorzaakt. Daarnaast bestaat het risico dat door de drukte fouten ontstaan of dat patiënten minder vlot geholpen worden dan gewenst.

2.1.4. Functionele eisen

Functionele eisen beschrijven wat een applicatie minimaal moet kunnen doen om aan de verwachtingen van de klant te voldoen. Het gaat hierbij om concrete, observeerbare functies en gedragingen van het systeem. Ze vormen de basis voor de verdere analyse, ontwikkeling en testen van de applicatie.

Bij functionele eisen wordt niet gekeken naar hoe de toepassing technisch wordt opgebouwd, maar wel naar welke acties gebruikers moeten kunnen uitvoeren, welke processen ondersteund moeten worden en welke resultaten het systeem moet opleveren.

2.1.4.1. Use-cases

Use case 1

Naam: Aanmelden afspraak bloedanalyse

Actors: Patiënt

Functionaliteit: Als patiënt kan ik mezelf aanmelden voor een bloedafname

Voorwaarden: De patiënt heeft zijn/haar eID bij.

Normale flow:

Systeem toont een startscherm dat de patiënt vraagt om zijn/haar identiteitskaart in de ID-reader te steken. De patiënt steekt de eID in de lezer. Het systeem leest de info van de ID, zoekt naar een elektronisch aanvraagformulier, toont een lijst van dokters en vraagt of de juiste dokter in de lijst staat. De patiënt drukt op "Ja". Het systeem stuurt de patiënt naar de wachtaal.

Alternatieve flow:

De patiënt is nog niet ingeschreven bij de inkom: Het systeem stuurt de patiënt naar de inkom.

De eID kan niet uitgelezen worden: Het systeem stuurt de patiënt naar de balie.

De patiënt ziet zijn/haar dokter niet: Het systeem vraagt of de patiënt een papieren aanvraagformulier heeft.

- De patiënt drukt op "Ja". Het systeem stuurt de patiënt naar de wachtaal.
- De patiënt drukt op "Nee". Het systeem stuurt de patiënt naar de balie.

Het systeem vindt geen elektronisch aanvraagformulier: Het systeem vraagt of de patiënt een papieren aanvraagformulier heeft.

- De patiënt drukt op "Ja". Het systeem stuurt de patiënt naar de wachtaal.
- De patiënt drukt op "Nee". Het systeem stuurt de patiënt naar de balie.

Use case 2

Naam: Balie inschrijving

Actors: Bاليemedewerker en patiënt

Functionaliteit: Als baliemedewerker kan ik patiënten in de wachtlijst zetten als ze aan de balie komen

Voorwaarden: /

Normale flow:

Het systeem toont een scherm met een lijst van patiënten die naar de balie zijn gestuurd. De baliemedewerker zoekt extern de patiëntinformatie en checkt dat de patiënt een aanvraagformulier heeft. Als alles in orde is klikt de baliemedewerker op de juiste patiënt en daarna op "Naar wachtaal". Het systeem zet de patiënt in fase "Wachtaal".

Uitzondering:

De patiënt staat niet in de lijst: De baliemedewerker drukt op de knop "Patiënt toevoegen". Het systeem toont een scherm met invulvelden voor patiënt informatie. De patiënt steekt zijn eID in de reader of de baliemedewerker vult de informatie zelf in en drukt op "Ok". Het systeem slaat de patiënt op en zet hem/haar in de juiste fase.

De patiënt komt niet opdagen: De baliemedewerker klikt op de patiënt en daarna op "Patiënt annuleren". Het systeem zet de patiënt in de fase "Geannuleerd".

Use case 3

Naam: Lijst patiënten zien en beheren

Actors: Verpleger

Functionaliteit: Als verpleger kan ik een lijst van aanwezige patiënten zien en hun fase veranderen

Voorwaarden: /

Normal flow:

Het systeem toont een scherm met een lijst van patiënten die in de wachtaal zitten. De verpleger klikt op de patiënt en daarna op de knop "In behandeling" zodat de fase van de patiënt op in behandeling komt. Na de behandeling klikt de verpleger op de knop "Behandeld". Het systeem verandert de fase van patiënt naar behandeld en wordt verwijderd uit de lijst.

Uitzondering: /

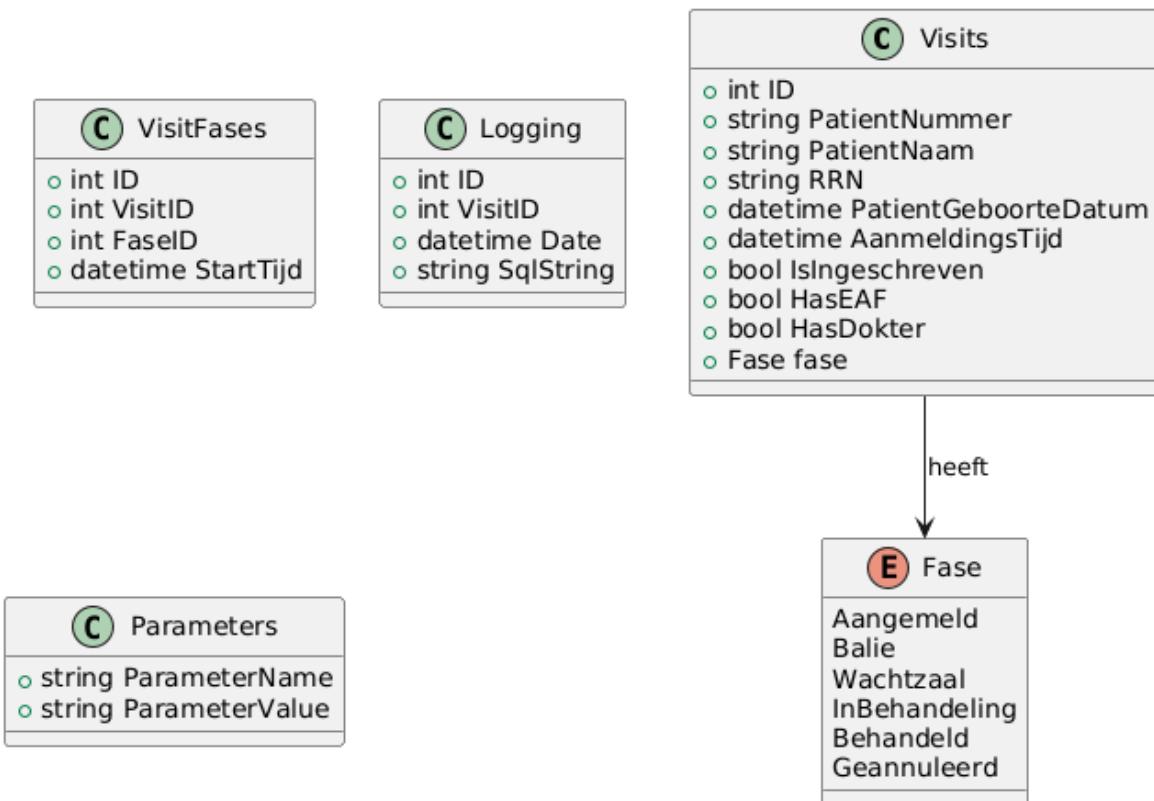
2.1.5. Technische eisen

Technische eisen beschrijven hoe de applicatie technisch gerealiseerd moet worden. Terwijl functionele eisen bepalen wat de applicatie moet doen, leggen technische eisen vast op welke manier dit moet gebeuren, binnen welke technische randvoorwaarden en met welke technologieën.

Technische eisen zijn bedoeld om duidelijkheid te scheppen voor ontwikkelaars, systeembeheerders en andere technische betrokkenen. Ze zorgen ervoor dat de applicatie voldoet aan de technische standaarden van de organisatie en dat het systeem veilig, stabiel, performant en onderhoudbaar is.

2.1.5.1. Data-laag

Onderstaand UML-klassendiagram geeft een overzicht van de structuur en de onderlinge relaties tussen de belangrijkste componenten binnen het registratiesysteem. Het diagram bestaat uit drie kernonderdelen: Visits, Fase, Logging VisitFases en Parameters.



Afbeelding 1: Data UML Wachtzaal

De centrale klasse Visits bevat alle gegevens van een patiëntbezoek, zoals persoonsgegevens, aanmeldingstijd en verschillende statusindicatoren. Elk bezoek heeft een gekoppelde Fase, een enumeratie die de actuele status van het proces weergeeft (zoals Aangemeld, Balie, Wachtzaal, InBehandeling, Behandeld en Geannuleerd).

Om de voortgang van een bezoek nauwkeurig te kunnen volgen, registreert de klasse VisitFases elke fasewijziging met het bijbehorende tijdstip. Daarnaast legt de klasse Logging belangrijke systeemacties vast, zoals uitgevoerde SQL-instructies en het moment waarop deze plaatsvonden. Dit ondersteunt foutopsporing en auditeerbaarheid.

De klasse Parameters bevat configuratie-instellingen waardoor bepaalde aspecten van het systeem flexibel aangepast kunnen worden zonder codewijzigingen.

2.1.5.2. Business-laag

De business-laag vormt het hart van de applicatie en bevat de kernlogica die het aanmeld- en opvolgproces van patiënten ondersteunt. Deze laag definieert de belangrijkste objecten en zorgt ervoor dat de applicatie op een consistente en gestructureerde manier met patiëntgegevens en bezoeken omgaat.

Het object Patiënt vertegenwoordigt alle relevante persoonsgegevens van een patiënt, waaronder naam, patiëntnummer, rijksregisternummer en geboortedatum. Dit model biedt een uniforme structuur voor het opslaan en hergebruiken van patiëntinformatie binnen het hele systeem.

Het object Visit beschrijft een individueel bezoek aan het labo. Elk bezoek bevat een uniek ID en een gekoppelde Patiënt, aangevuld met informatie zoals het tijdstip van aanmelding, of er een elektronische aanvraag (EAF) aanwezig is, of de patiënt een behandelende arts heeft, en in welke Fase het bezoek zich bevindt. Deze fase volgt de voortgang van de patiënt doorheen het aanmeldproces.

De Fase-enumeratie definieert de mogelijke stappen in de workflow van een bezoek, gaande van Aangemeld tot Behandeld of Geannuleerd. Door gebruik te maken van een duidelijke set statussen kan het systeem nauwkeurig bepalen waar een patiënt zich in het proces bevindt en hierop gepaste acties ondernemen.

2.1.5.3. Presentatie-laag

De presentatie-laag vormt de visuele interface van de applicatie en bestaat uit twee delen: een gebruiksvriendelijke aanmeldinterface voor patiënten aan de inkom van de wachtzaal en een beheeroverzicht voor de medewerkers van het laboratorium. Beide onderdelen zijn ontworpen met een duidelijke en intuïtieve structuur, zodat gebruikers snel en efficiënt kunnen handelen.

PATIËNT DEEL - AANMELDING

Het eerste onderdeel van de presentatie-laag is een Windows Form die draait op een all-in-one pc bij de ingang van de wachtzaal. Dit scherm begeleidt patiënten doorheen het aanmeldproces voor bloedafnames.

Form 1 bevat meerdere GroupBoxes (schermen) die verschijnen op basis van de situatie van de patiënt:

- **GroupBox 1 – Startscherm**
Dit scherm geeft duidelijke instructies voor alle types bezoekers.
 - *Voor materiaal ophalen*: ga naar de balie
 - *Voor staal afgeven*: ga naar de balie
 - *Voor bloedafname*: steek uw eID in de kaartlezer
 - Indien *geen aanmelding gevonden*: GroupBox 5
 - Indien *geen elektronisch aanvraagformulier gevonden*: GroupBox 2
 - Indien *wel een aanvraagformulier gevonden*: GroupBox 3
 - *Voor bloedafname zonder eID*: ga naar de balie
- **GroupBox 2 – Geen elektronische aanvraag gevonden**
De patiënt kan hier kiezen:
 - *Ik heb een papieren aanvraagformulier*: verder naar GroupBox 3
 - *Geen aanvraagformulier*: verder naar GroupBox 4
- **GroupBox 3 – Wachtzaalinstructie**
Meldt duidelijk dat de patiënt mag plaatsnemen in de wachtzaal.
- **GroupBox 4 – Bezoek zonder aanvraagformulier**
Instructie om zich aan de balie aan te melden voor verdere ondersteuning.
- **GroupBox 5 – Geen aanmelding gevonden**
Geeft aan dat de patiënt zich eerst moet registreren aan de inkom van het ziekenhuis.

Deze schermen zorgen voor een vlot en geautomatiseerd aanmeldproces, waarbij de handelingen van de baliemedewerkers worden beperkt en patiënten snel de juiste richting krijgen.

LABOMEDEWERKERS DEEL - OVERZICHTSSCHERM

Het tweede onderdeel van de presentatie-laag bestaat uit een beheerscherm voor de medewerkers van het labo. Dit overzicht geeft real-time inzicht in alle lopende bezoeken.

Het bevat onder andere:

- **Een lijst met alle patiënten die momenteel in behandeling zijn**, inclusief hun naam, patiëntnummer, rijksregisternummer en geboortedatum.
- **De huidige fase van elk bezoek**, zoals Aangemeld, Balie, Wachtzaal of In Behandeling. Zo weten medewerkers precies wie waar in het proces zit.
- **Mogelijkheden om een bezoek handmatig bij te werken**, zoals het wijzigen van de fase of het afronden van een behandeling.

Dit overzichtsscherm brengt alle relevante informatie samen in één centrale interface, wat zorgt voor efficiëntie, transparantie en een betere workflow voor het labopersoneel.

2.1.6. Technologieën

Voor de realisatie van de wachtaalapplicatie wordt gebruikgemaakt van een combinatie van betrouwbare technologieën die afgestemd zijn op de bestaande IT-omgeving van het ziekenhuis. De applicatie zal ontwikkeld worden in C# met behulp van Windows Forms.

Alle gegevens worden centraal opgeslagen in een SQL Server-database, waarin verschillende tabellen zoals Visits, VisitFases, Logging en Parameters worden beheerd.

Voor het inlezen van patiëntgegevens wordt een eID-lezer gebruikt in combinatie met Swelio, een bibliotheek voor het uitlezen van Belgische elektronische identiteitskaarten. Dit maakt een snelle en automatische registratie mogelijk zonder manuele invoer.

2.2. Realisatie

2.2.1. Doel

Het doel van dit project is het ontwikkelen van een gebruiksvriendelijke en efficiënte wachtzaalapplicatie voor de dienst Laboratorium. De applicatie moet het aanmeldingsproces voor patiënten die een bloedafname komen laten uitvoeren automatiseren, de werkbelasting aan de balie verlagen en de patiëntengroei duidelijker en overzichtelijker maken. Daarnaast moet het laboratoriumpersoneel een overzicht krijgen van alle patiënten per fase in het proces, zodat zij het volledige verloop kunnen beheren en opvolgen.

2.2.2. Functionaliteiten

PATIËNTGEDEELTE (ALL-IN-ONE PC AAN DE INGANG):

- Automatische aanmelding via eID.
- Uitlezen van patiëntgegevens via Swelio.
- Controleren of de patiënt correct is aangemeld in het ziekenhuis.
- Controleren of er een elektronisch aanvraagformulier aanwezig is.
- Automatisch toewijzen van een fase: Aangemeld → Balie → Wachtzaal.
- Duidelijke visuele instructies op basis van de gevonden informatie.
- Alternatieve flow voor patiënten zonder eID of zonder aanvraagformulier.
- Systeem om te kunnen switchen tussen Nederlands en Engels.

PERSONEELSGEDEELTE (OVERZICHTSSCHERM):

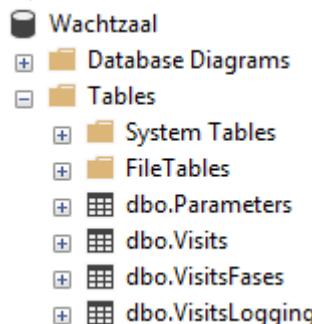
- Overzicht van alle patiënten per fase met filters.
- Mogelijkheid om patiënten handmatig door te schuiven naar de volgende fase.
- Manueel patiënten kunnen toevoegen aan de wachtlijst.
- Automatische registratie van aanmeldings- en doorstroomtijden.

2.2.3. 3-tier architectuur

De applicatie werd opgebouwd volgens de 3-tier architectuur om een duidelijke scheiding te creëren tussen data opslag, logica en presentatie. Dit verhoogt de onderhoudbaarheid en flexibiliteit van de software.

2.2.3.1. Data-laag

SQL Server structuur:



Afbeelding 2: SQL database structuur

dbo.Parameters:

Column Name	Data Type	Allow Nulls
ParameterName	nvarchar(50)	<input type="checkbox"/>
ParameterValue	nvarchar(50)	<input type="checkbox"/>

Afbeelding 3: Database tabel Parameters

dbo.Visits:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
PatientNummer	nvarchar(50)	<input checked="" type="checkbox"/>
PatientNaam	nvarchar(250)	<input checked="" type="checkbox"/>
RRN	nvarchar(50)	<input checked="" type="checkbox"/>
PatientGeboorteDatum	datetime	<input checked="" type="checkbox"/>
AanmeldingsTijd	datetime	<input checked="" type="checkbox"/>
FaseID	int	<input checked="" type="checkbox"/>
IsIngeschreven	bit	<input checked="" type="checkbox"/>
HasEAF	bit	<input checked="" type="checkbox"/>
HasDokter	bit	<input checked="" type="checkbox"/>
InsertedOn	datetime	<input checked="" type="checkbox"/>
InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
UpdatedOn	datetime	<input checked="" type="checkbox"/>
UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 4: Database tabel Visits

dbo.VisitFases:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
VisitID	int	<input checked="" type="checkbox"/>
FaseID	int	<input checked="" type="checkbox"/>
StartTijd	datetime	<input checked="" type="checkbox"/>

Afbeelding 5: Database tabel VisitFases

dbo.VisitsLogging:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
VisitID	int	<input checked="" type="checkbox"/>
ExecutedTijd	datetime	<input checked="" type="checkbox"/>
SQLCommandText	nvarchar(MAX)	<input checked="" type="checkbox"/>
InsertedOn	datetime	<input checked="" type="checkbox"/>
InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
UpdatedOn	datetime	<input checked="" type="checkbox"/>
UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 6: Database tabel VisitsLogging

Data.cs:

De data klasse bevat een reeks aan functies die data ophalen, updaten, invoegen en verwijderen aan de hand van een SQL connectie.

```
9 references
public static class Data
{
    // Connection string naar SQL Server
    private const string ConnectionString = @"...";

    2 references
    public static DataTable GetVisits(long id = 0, string patientNummer = null, string patientNaam = null, string rrn = null,
        DateTime? patientGeboorteDatum = null, DateTime? aanmeldingsTijd = null, bool? isIngeschreven = null,
        bool? hasEAF = null, bool? hasDokter = null, int? faseID = null)...

    1 reference
    private static string BuildWhereVisits(long? id, string patientNummer, string patientNaam, string rrn, DateTime? patientGeboorteDatum,
        DateTime? aanmeldingsTijd, bool? isIngeschreven, bool? hasEAF, bool? hasDokter, int? faseID)...

    // Haalt één visit op via Id
    1 reference
    public static DataRow GetVisit(long id)...;

    // Nieuwe visit toevoegen
    1 reference
    public static long InsertVisit(string patientNummer, string patientNaam, string rrn, DateTime? patientGeboorteDatum, DateTime? aanmeldingsTijd,
        bool? isIngeschreven, bool? hasEAF, bool? hasDokter, int faseID)...

    // Update een bestaande visit volledig
    1 reference
    public static int UpdateVisit(long id, string patientNummer, string patientNaam, string rrn, DateTime? patientGeboorteDatum,
        DateTime? aanmeldingsTijd, bool? isIngeschreven, bool? hasEAF, bool? hasDokter, int faseID)...

    1 reference
    public static int UpdateVisitFase(long id, int faseID)...;

    // Verwijderd een visit op basis van Id
    0 references
    public static int DeleteVisit(long id)...;

    1 reference
    public static long InsertVisitFase(long visitID, long faseID, DateTime startTijd)...;

    // Methode om nieuwe ID op te halen
    3 references
    public static long GetNewID(string tableName)...;
```

Afbeelding 7: Data klasse Wachtzaal

2.2.3.2. Business-laag

De business-laag bevat de logica die bepaalt hoe de applicatie moet functioneren. Hier worden onder andere de objecten Patient en Visit beheerd en wordt de validatie van data uitgevoerd.

C# ObjPatient.cs

C# ObjVisit.cs

ObjPatient:

```
public class ObjPatient
{
    3 references
    public ObjPatient() { }

    0 references
    public ObjPatient(string patientNummer, string patientNaam, string rrn, DateTime patientGeboorteDatum)
    {
        PatientNummer = patientNummer;
        PatientNaam = patientNaam;
        RRN = rrn;
        PatientGeboorteDatum = patientGeboorteDatum;
    }

    9 references
    public string PatientNummer { get; set; }
    7 references
    public string PatientNaam { get; set; }
    10 references
    public string RRN { get; set; }
    8 references
    public DateTime? PatientGeboorteDatum { get; set; }

    1 reference
    public bool IsValid(out string errorText)...
}
```

Afbeelding 8: Object Patient

ObjVisit:

```
public class ObjVisit
{
    2 references
    public ObjVisit()
    {
        ID = null;
        Patient = new ObjPatient();
    }

    0 references
    public ObjVisit(long? id)
    {
        ID = id;
        Patient = new ObjPatient();
        Load();
    }

    9 references
    public long? ID { get; set; }
    22 references
    public ObjPatient Patient { get; set; }
    6 references
    public DateTime? AanmeldingsTijd { get; set; }
    6 references
    public bool? IsIngeschreven { get; set; }
    7 references
    public bool? HasElecAfspraak { get; set; }
    11 references
    public bool? HasDokter { get; set; }
    11 references
    public Enums.Fase? Fase { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    1 reference
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    2 references
    public bool IsValid(out string errorText)...

    1 reference
    public bool Load()...

    3 references
    public bool Save(Enums.Fase? fasePrevious, out string errorText)...
}
```

Afbeelding 9: Object Visit

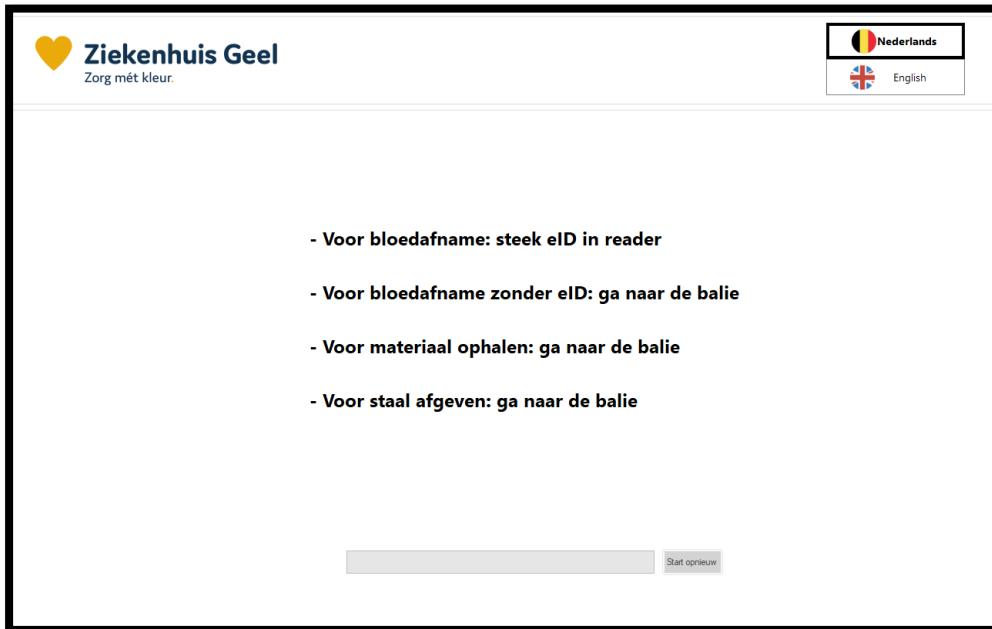
2.2.3.3. Presentatie-laag

De presentatie-laag bestaat uit Windows Forms en vormt de visuele interface voor zowel patiënten als laboratoriummedewerkers.

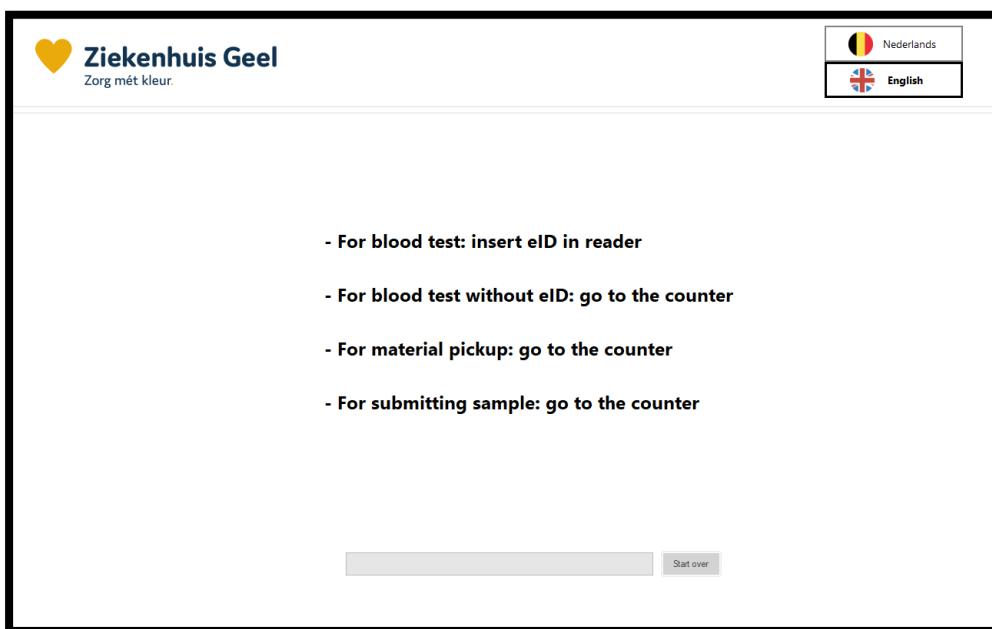
PATIËNT DEEL – AANMELDING

Alle schermen zijn zowel in het Nederlands als Engels te gebruiken. Voor de patiënt is het niet mogelijk om de applicatie te sluiten, dit is enkel mogelijk met een toetsencombinatie.

Startscherm (NL – EN):



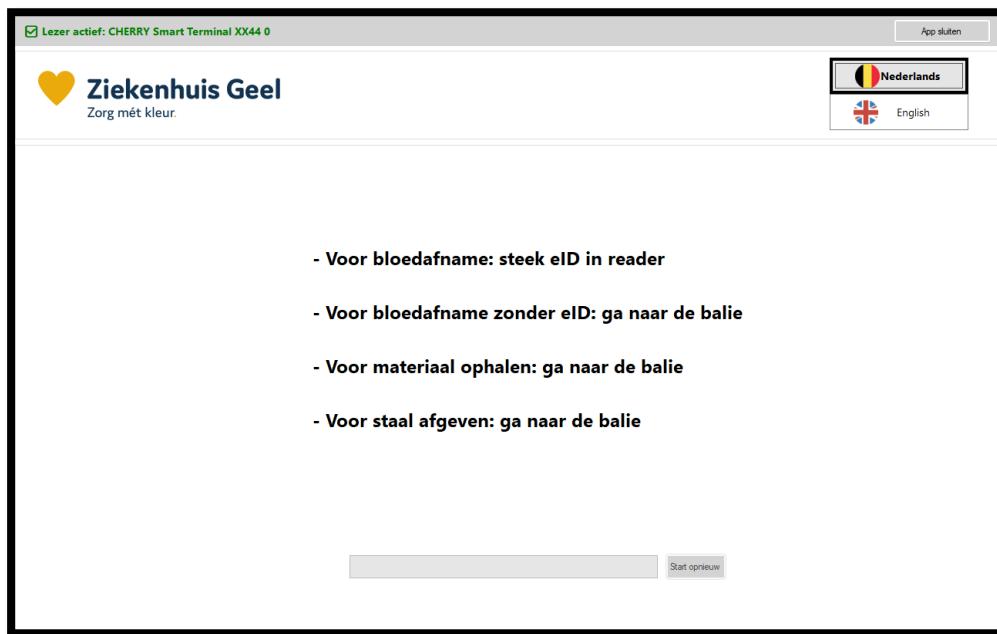
Afbeelding 10: Startscherm Wachtzaal Nederlands



Afbeelding 11: Startscherm Wachtzaal Engels

Startscherm met statusbalk (enkel voor personeel):

Dit is bedoeld zodat er gecheckt kan worden of de eID-reader wel herkent wordt.

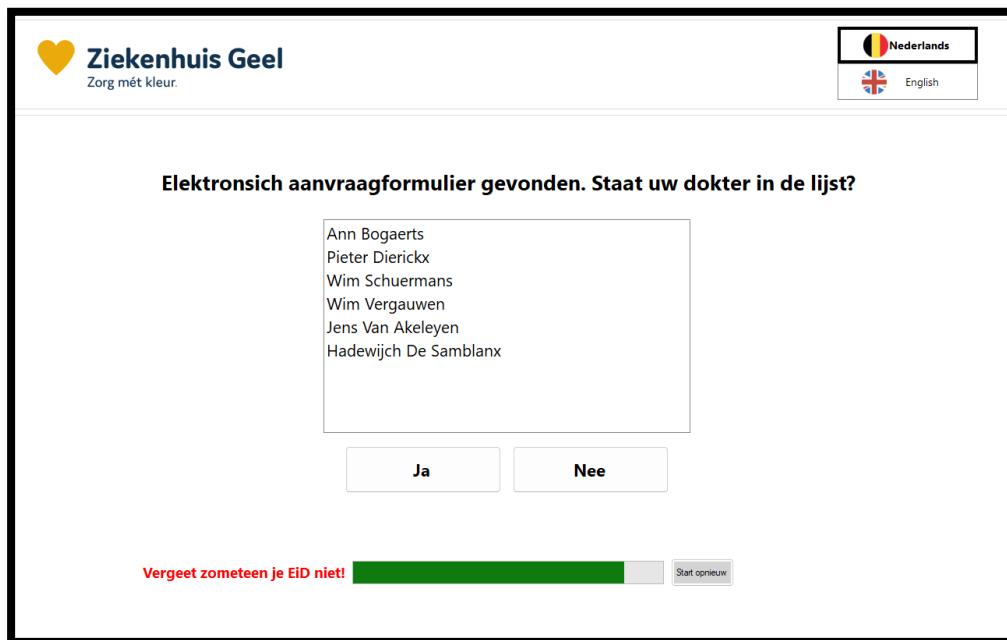


Afbeelding 12: Startscherm Wachtzaal met statusbalk

Bij alle schermen buiten het startscherm is er een balk die afloopt om te voorkomen dat een onafgewerkte aanmelding op het scherm blijft staan. Als de balk leeg is ga je terug naar het startscherm. Er is een mogelijkheid om de balk te herstarten als je meer tijd nodig hebt. Ook komt er een rood label op het scherm als je eID nog in de reader zit.

NORMAAL VERLOOP

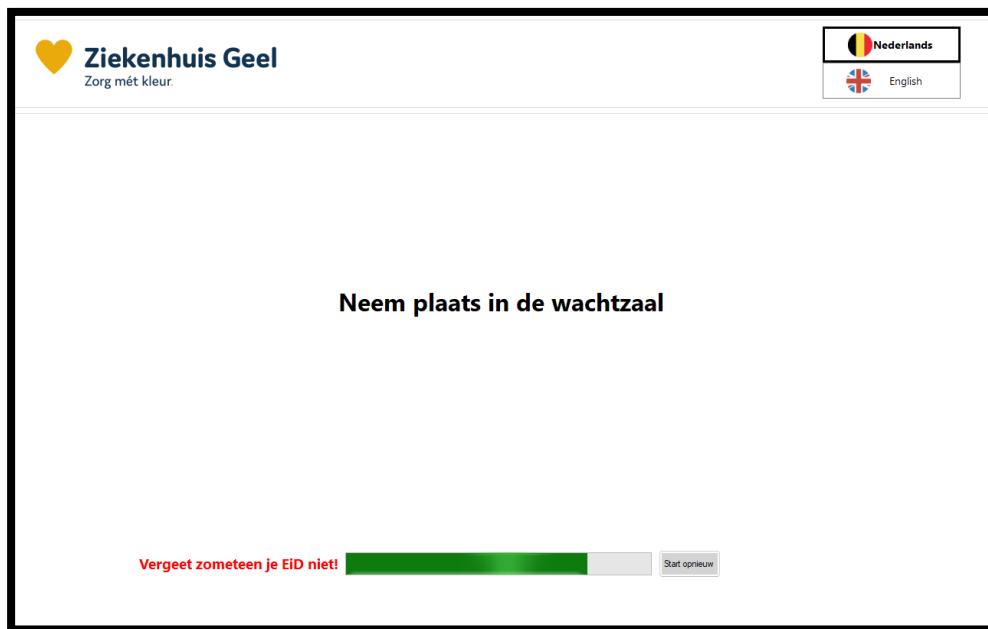
Elektronisch aanvraagformulier gevonden scherm:



Afbeelding 13: Elektronisch aanvraagformulier gevonden scherm

DOKTER STAAT IN DE LIJST → PATIËNT KLIKT OP “JA”

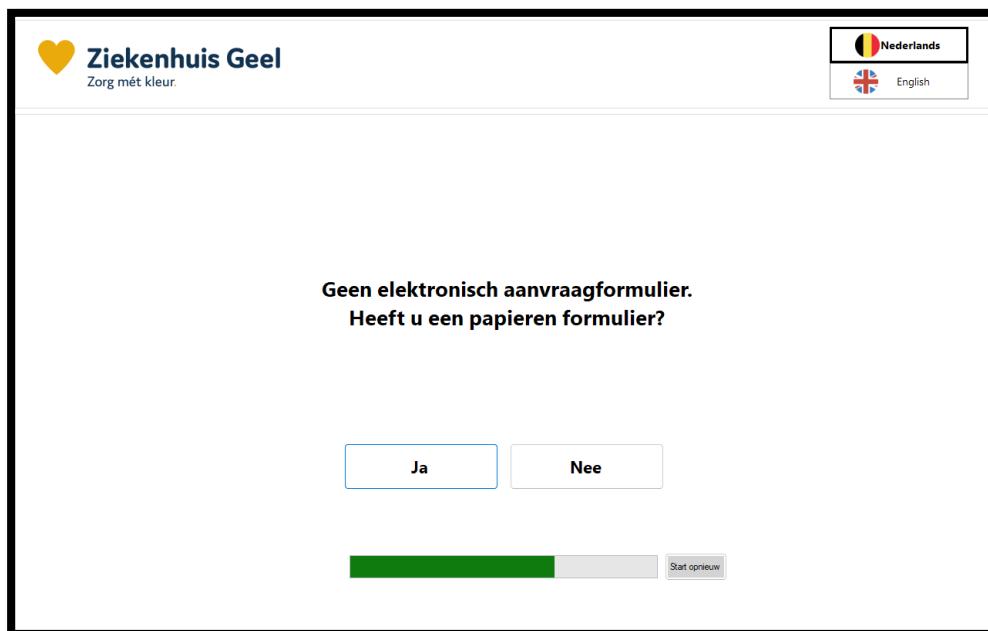
Wachtzaal scherm:



Afbeelding 14: Wachtzaal scherm

DOKTER STAAT NIET IN DE LIJST → PATIËNT KLIKT OP “NEE”

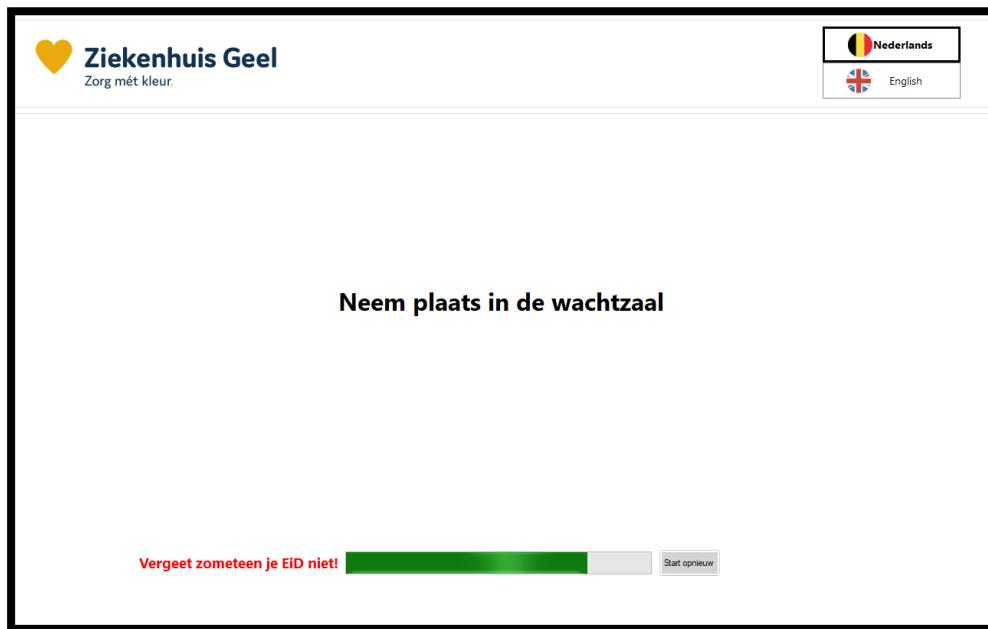
Geen elektronisch aanvraagformulier scherm:



Afbeelding 15: Geen elektronisch aanvraagformulier scherm

PATIËNT HEEFT EEN PAPIEREN AANVRAAGFORMULIER → PATIËNT KLIKT OP “JA”

Wachtzaal scherm:



Afbeelding 16: Wachtzaal scherm

PATIËNT HEEFT GEEN PAPIEREN AANVRAAGFORMULIER → PATIËNT KLIKT OP “NEE”

Meld aan bij balie scherm:



Afbeelding 17: Meld aan bij balie scherm

PATIËNT IS NIET INGESCHREVEN BIJ DE INKOM VAN HET ZIEKENHUIS

Meld aan bij inkom scherm:

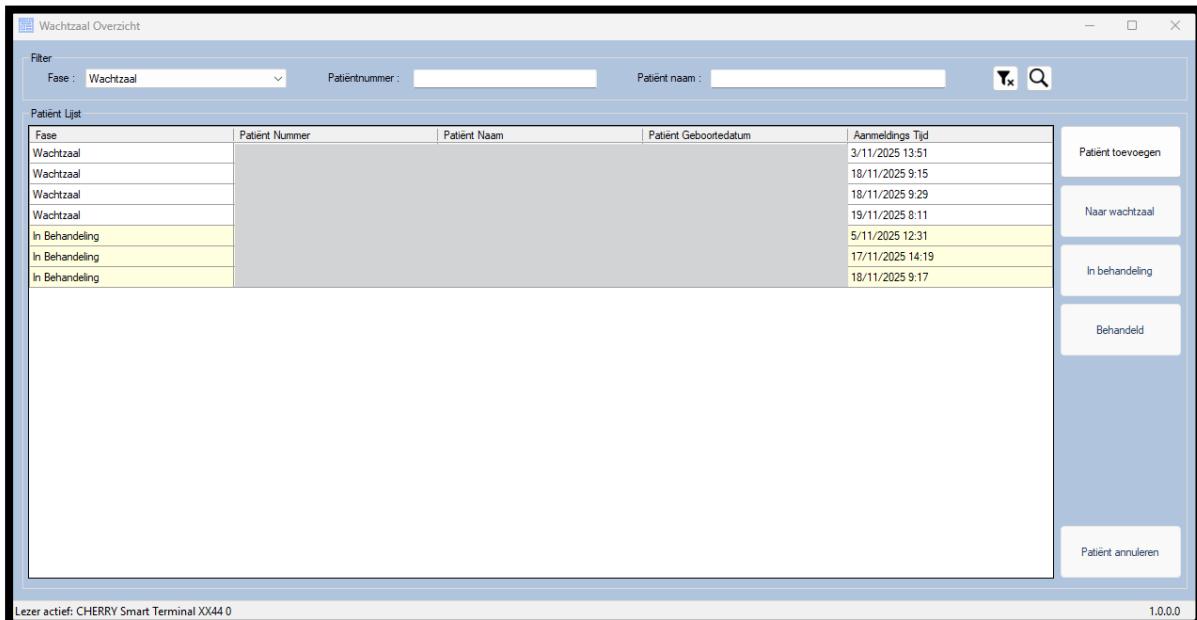


Afbeelding 18: Meld aan bij inkom scherm

LABOMEDEWERKERS DEEL – OVERZICHTSSCHERM

Het overzicht toont een lijst van patiënten die gefilterd kan worden op fase, patiëntnummer en/of patiënt naam. Het is mogelijk om een patiënt te selecteren door erop te klikken, dan heb je de mogelijkheid om de knoppen “Naar wachtzaal”, “In behandeling”, “Behandeld” en “Patiënt annuleren” te gebruiken. Deze knoppen zullen de fase van de patiënt updaten.

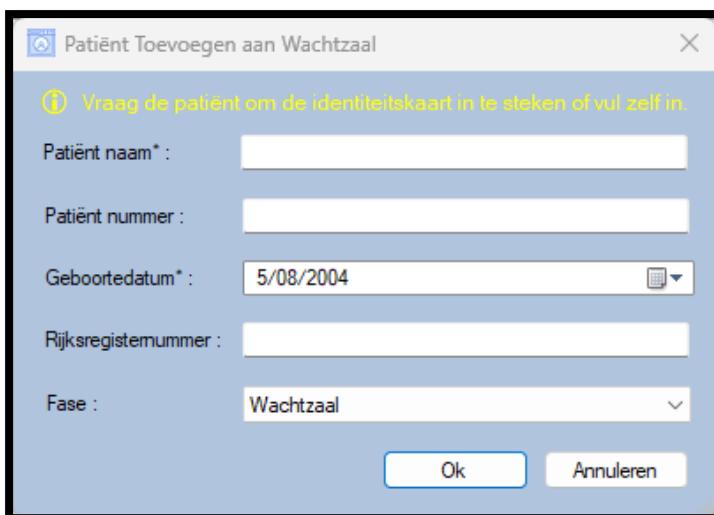
Overzicht scherm:



Afbeelding 19: Wachtzaal overzicht scherm

Daarnaast is er de mogelijkheid om een patiënt die nog niet in het systeem zit toe te voegen. Dit gebeurt via de knop “Patiënt toevoegen”, als er op de knop wordt geklikt opent onderstaande dialoog. Hierin kan de eID reader gebruikt worden of de baliemedewerker vult het zelf in, enkel patiënt naam, geboortedatum en fase zijn verplichte velden. Als alle informatie is ingevuld drukt de baliemedewerker op “Ok” en wordt de patiënt toegevoegd aan de lijst.

Patiënt toevoegen pop-up:



Afbeelding 20: Patiënt toevoegen pop-up

2.2.4. Uitdagingen

Dit was een complex project en bracht verschillende uitdagingen met zich mee. In de beginfase was het niet eenvoudig om volledig te begrijpen welke omstandigheden en parameters invloed hadden op de route die een patiënt moest volgen, wat essentieel was voor een correcte functionele uitwerking. Daarnaast vergde de integratie van de eID-lezer extra tijd, aangezien deze werkte met een externe bibliotheek waarmee ik nog niet vertrouwd was.

Aangezien dit mijn eerste project binnen het ziekenhuis was, moest ik bovendien wennen aan de gehanteerde werkwijze, interne processen en ontwikkelgewoontes. Ook was het de eerste keer dat ik met Windows Forms werkte, wat de nodige gewenning en technische verkenning vereiste. Ondanks deze leercurve heeft dit project een stevige basis gevormd voor de rest van mijn stage.

2.2.5. Eindresultaat

Het eindresultaat is een volledig functionele wachtzaalapplicatie die het volledige proces rond bloedafnames automatiseert en structureert. Patiënten kunnen zich zelfstandig aanmelden via hun eID en worden automatisch naar de juiste fase gedirigeerd. Het laboratoriumpersoneel beschikt over een duidelijk overzicht van alle patiënten en kan het proces efficiënt beheren. Dit leidt tot kortere wachttijden, minder werkdruk aan de balie en een vlottere doorstroming in de wachtzaal. De applicatie draait stabiel in de ziekenhuisomgeving en integreert naadloos met de bestaande infrastructuur.

3. Magazijn etiketten

3.1. Analyse

3.1.1. Klant

De klant voor dit project is de dienst Magazijn van het ziekenhuis. Deze dienst is verantwoordelijk voor het beheren van voorraden en het voorzien van correcte productinformatie en labels voor interne distributie. Het magazijn werkt dagelijks met grote aantallen artikels, waardoor efficiëntie en foutloze verwerking essentieel zijn.

3.1.2. Huidige situatie

Binnen het magazijn moeten medewerkers etiketten printen voor verschillende artikels en producten. Hiervoor zoeken zij gegevens zoals het artikelnummer en de kostenplaats op in de SAP-database. Dit gebeurt volledig handmatig, wat het proces vertraagd en afhankelijk maakt van de medewerker.

3.1.3. Probleem

Doordat het volledige proces manueel verloopt, ontstaan regelmatig problemen zoals tijdsverlies door repetitieve handelingen, een verhoogde kans op fouten bij het zoeken van gegevens in SAP, het printen van etiketten met incorrecte informatie of verkeerde barcodes en inconsistentie in etiketten wanneer meerdere medewerkers ze handmatig samenstellen. Dit leidt tot inefficiëntie en een verhoogde werkdruk binnen het magazijn.

3.1.4. Functionele eisen

Functionele eisen beschrijven welke acties en functies de applicatie moet ondersteunen vanuit het perspectief van de gebruiker.

3.1.4.1. Use cases

Use case 1

Naam: Nieuwe etiketten printen

Actors: Magazijnmedewerker

Functionaliteit: Als magazijnmedewerker kan ik een etiket printen aan de hand van het artikelnummer en kostenplaats.

Voorwaarden: /

Normale flow:

Het systeem geeft een scherm met 2 invulvelden, artikelnummer en kostenplaats. De magazijnmedewerker vult de info van het artikel in en drukt op "Afdrukken". Het systeem zoekt het artikel, maakt het etiket en drukt het af.

Uitzondering:

De magazijnmedewerker vult een onbestaande combinatie in: Het systeem geeft een foutmelding dat er geen artikel gevonden kon worden met deze combinatie.

3.1.5. Technische eisen

Technische eisen bepalen hoe de applicatie moet werken en welke randvoorwaarden of beperkingen moeten worden gerespecteerd.

3.1.5.1. Data-laag

Voor dit project is er geen database nodig, er wordt enkel data opgehaald uit de bestaande SAP-database. Wel moeten we weten welke data we precies nodig hebben uit de grote en onduidelijke database, deze database zal wegens security redenen niet getoond worden.

3.1.5.2. Business-laag

De business-laag bevat de kernlogica van de applicatie en maakt gebruik van één centraal object: Etiket. Dit object omvat alle gegevens die nodig zijn om een correct en uniform magazijnlabel te genereren. Het bevat eigenschappen zoals Artikelnummer en Kostenplaats, op basis waarvan de applicatie automatisch aanvullende informatie ophaalt uit de SAP-database.

Daarnaast omvat het object gegevens zoals Beschrijving, Referentie, Leverancier, AantalLabel en BarcodeCijfers, die gebruikt worden om het etiket volledig en duidelijk op te bouwen.

Door alle relevante informatie te bundelen in één gestructureerd object, kan de applicatie snel, foutloos en consistent etiketten aanmaken, wat de efficiëntie binnen het magazijn aanzienlijk verhoogt.

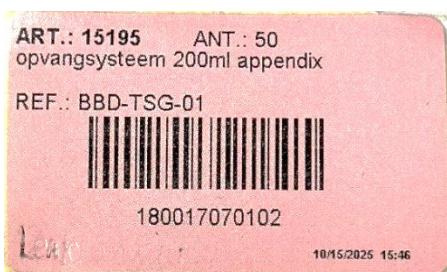
3.1.5.3. Presentatie-laag

De presentatie-laag bestaat uit een gebruiksvriendelijke Windows Forms-interface waarmee magazijnmedewerkers snel etiketten kunnen genereren en printen.

Deze interface bevat:

- Een invoerveld voor artikelnummer en kostenplaats
- Automatische weergave van opgehaalde artikelgegevens
- Een visuele preview van het etiket
- Een printknop die het etiket rechtstreeks naar de labelprinter stuurt
- Foutmeldingen bij ontbrekende gegevens of ongeldige invoer

Voor het etiket heb ik een voorbeeld gekregen van hoe het eruit moet zien, de roze kleur is het etiketpapier en moet geen rekening mee gehouden worden. Wel moet er links onder de leverancier naam op staan:



Afbeelding 21: Voorbeeld magazijn etiket

3.1.6. Technologieën

Voor de ontwikkeling van de etikettenapplicatie is gekozen voor technologieën die betrouwbaar zijn en goed integreren met de bestaande IT-omgeving van het magazijn. De applicatie wordt gemaakt in C# met Windows Forms met de ziekenhuis huisstijl zodat alles meteen herkenbaar is voor de gebruikers.

De koppeling met SAP gebeurt via SAP-databaseconnecties, waarmee de applicatie artikel- en kostenplaatsgegevens op een consistente en veilige manier kan ophalen. Voor het genereren van barcodes wordt .NET-bibliotheek Zen.Barcode gebruikt voor barcodecreatie, zodat elke barcode voldoet aan de standaarden die het magazijn hanteert.

De uiteindelijke etiketten worden rechtstreeks geprint via de drivers van de labelprinter, waardoor medewerkers zonder tussenstappen of extra software snel een correct etiket kunnen afdrukken. Deze combinatie van technologieën vormt een stabiele basis voor een efficiënt en foutloos proces.

3.2. Realisatie

3.2.1. Doel

Het doel van dit project is het ontwikkelen van een snelle en betrouwbare oplossing waarmee magazijnmedewerkers automatisch correcte etiketten kunnen genereren en printen op basis van productgegevens uit de SAP-database. Hierdoor wordt het proces efficiënter, worden fouten verminderd en verbeterd de doorstroom binnen het magazijn.

3.2.2. Functionaliteiten

De applicatie omvat de volgende kernfunctionaliteiten:

- Zoeken van producten op basis van artikelnummer en kostenplaats.
- Automatisch ophalen van alle productinformatie uit SAP.
- Genereren van een etiket aan de hand van de gevonden informatie.
- Direct printen naar een gekoppelde etikettenprinter.
- Foutafhandeling wanneer gegevens ontbreken of niet gevonden worden.

3.2.3. 3-tier architectuur

Ondanks dat de applicatie geen eigen database gebruikt, is er toch gekozen voor een 3-tier architectuur. Deze structuur zorgt ervoor dat alle functionaliteiten duidelijk gescheiden blijven, wat de onderhoudbaarheid en uitbreidbaarheid van het systeem bevordert. Daarnaast maakt deze scheiding het eenvoudiger om fouten op te sporen en te verwerken, doordat elke laag een eigen, afgebakende verantwoordelijkheid heeft.

3.2.3.1. Data-laag

DataSAP.cs:

De DataSAP klasse bevat de functies die verantwoordelijk zijn om alle data op te halen uit de SAP database aan de hand van een SQL connectie.

```
public class DataSAP
{
    private const string ConnectionString = @"[REDACTED]";

    1 reference
    public static DataRow GetArtikel(string artikelnummer, string kostenplaats)...

    1 reference
    public static DataRow GetArtikelViaEina(string artikelnummer, string kostenplaats)...

    2 references
    private static string FormatArtikelnummer(string artikelnummer)...
}
```

Afbeelding 22: Data klasse Magazijn Etiketten

3.2.3.2. Business-laag

De business-laag valideert de ingevoerde waarden, controleert of de SAP-data volledig en correct is en zet deze om in een structuur die geschikt is voor het genereren van een etiket.

C# ObjEtiket.cs

ObjEtiket:

```
public class ObjEtiket
{
    1 reference
    public ObjEtiket() { }

    0 references
    public ObjEtiket(string artikelnummer, string kostenplaats)
    {
        Artikelnummer = artikelnummer;
        Kostenplaats = kostenplaats;
    }

    4 references
    public string Artikelnummer { get; set; }
    3 references
    public string Kostenplaats { get; set; }
    3 references
    public string AantalLabel { get; set; }
    8 references
    public string Beschrijving { get; set; }
    3 references
    public string Referentie { get; set; }
    3 references
    public string Leverancier { get; set; }
    5 references
    public string BarcodeCijfers { get; set; }

    1 reference
    public bool IsValid(out string errorText)...
}
```

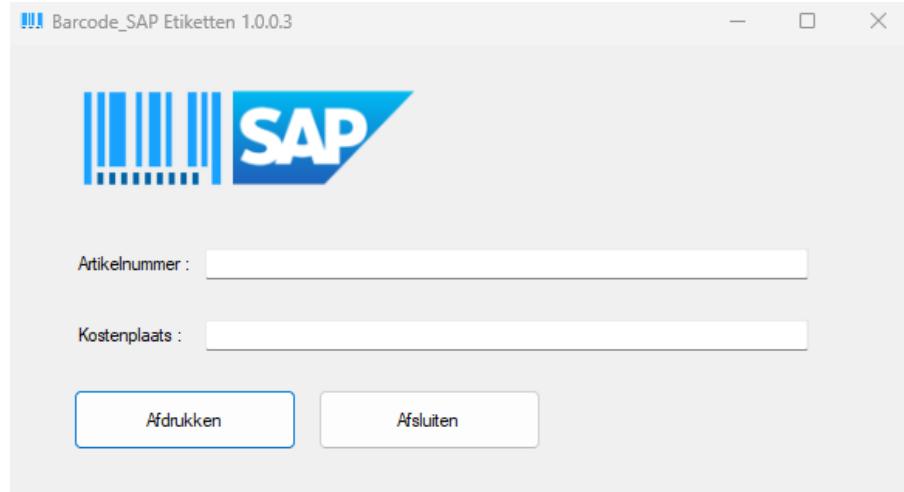
Afbeelding 23: Object Etiket

3.2.3.3. Presentatie-laag

De presentatie-laag bestaat uit Windows Forms en vormt een gebruiksvriendelijke interface voor zowel magazijnmedewerkers.

Startscherm:

Hier kan de magazijnmedewerker de artikelnummer en kostenplaats invullen en het etiket afdrukken.



Afbeelding 24: Startscherm Magazijn Etiketten

3.2.4. Uitdagingen

Dit project bracht verschillende nieuwe uitdagingen met zich mee. Het was de eerste keer dat ik werkte met een etikettenprinter en het genereren van barcodes. De grootste moeilijkheid lag echter bij het raadplegen van de SAP-database. Deze database bevat een zeer uitgebreide structuur met talloze tabellen en kolommen waarvan de benamingen niet altijd duidelijk waren. Voor het samenstellen van één etiket was informatie nodig uit meerdere tabellen, wat het analyseproces complex maakte.

Daarnaast kwam het regelmatig voor dat dezelfde combinatie van artikelnummer en kostenplaats meerdere resultaten opleverde. Het was daarom essentieel om een uniek en consistent kenmerk te vinden dat toeliet om altijd het juiste etiket te genereren. Uiteindelijk bleek dit het veld van de gebruikelijke leverancier te zijn, al was de naamgeving hiervan in de database opnieuw niet herkenbaar, waardoor het zoeken en valideren extra tijd en moeite kostte.

3.2.5. Eindresultaat

Het uiteindelijke resultaat is een stabiele, efficiënte en gebruiksvriendelijke Windows Forms-applicatie die magazijnmedewerkers toelaat om in slechts enkele seconden correcte etiketten te genereren en af te drukken. Dit heeft het volledige werkproces aanzienlijk versneld, de kans op fouten drastisch verminderd en de administratieve werklast merkbaar verlaagd.

Onderstaand een voorbeeld van een etiket dat met de applicatie werd geprint:



Afbeelding 25: Resultaat magazijn etiket

4. Keuken etiketten

4.1. Analyse

4.1.1. Klant

De klant voor dit project is de ziekenhuiskeuken. Deze dienst staat in voor de voedselvoorziening in het ziekenhuis. Wat ze ook doen en voor ons belangrijk is de verwerking van broodjesbestellingen die door het personeel via een externe website worden geplaatst. De keuken is verantwoordelijk voor het correct klaarmaken, verpakken en labelen van de bestellingen.

4.1.2. Huidige situatie

Ziekenhuispersoneel bestelt broodjes via de website Click4Food. In de ziekenhuiskeuken worden deze bestellingen vervolgens verwerkt, waarbij voor elk broodje een etiket moet worden afgedrukt om correct op de verpakking te worden aangebracht. De etiketten bevatten o.a. de naam van de besteller en details over de broodjes. Dit proces gebeurt volledig manueel en kost veel tijd.

4.1.3. Probleem

Het handmatig verwerken van bestellingen en het printen van etiketten verloopt traag en leidt regelmatig tot fouten. Tijdens drukke periodes veroorzaakt dit vertragingen in de workflow en verhoogt het risico op verkeerd gelabelde broodjes. Deze inefficiëntie zorgt voor onnodige werkdruk en kan leiden tot verwarring bij de distributie.

4.1.4. Functionele eisen

Functionele eisen beschrijven welke acties en functies de applicatie moet ondersteunen vanuit het perspectief van de gebruiker.

4.1.4.1. Use cases

Use case 1

Naam: Excel-file importeren

Actor: Keukenmedewerker

Functionaliteit: Als keukenmedewerker kan ik een Excel-file importeren en de data bekijken

Voorwaarden: /

Normale flow:

Het systeem geeft een scherm met een lege tabel. De keukenmedewerker drukt op de knop "Import Excel". Het systeem toont de File Explorer. De keukenmedewerker kiest de juiste Excel-bestand. Het systeem importeert de data in de tabel.

Uitzondering:

De keukenmedewerker kiest een verkeerde Excel-file: Het systeem geeft een foutmelding dat de Excel niet gelezen kan worden.

Use case 2

Naam: Etiketten afdrukken

Actor: Keukenmedewerker

Functionaliteit: Als keukenmedewerker kan ik etiketten afdrukken

Voorwaarden: Use case 1 is voltooid

Normale flow:

Het systeem geeft een scherm met de gevulde tabel. De keukenmedewerker drukt op de knop "Afdrukken". Het systeem drukt alle etiketten in de tabel af.

Uitzondering: /

4.1.5. Technische eisen

Technische eisen bepalen hoe de applicatie moet werken en welke randvoorwaarden of beperkingen moeten worden gerespecteerd.

4.1.5.1. Data-laag

In dit project is geen data-laag voorzien, aangezien er geen gegevens opgeslagen moeten worden. Alle informatie wordt direct ingelezen vanuit een Excel-bestand.

4.1.5.2. Business-laag

De business-laag bevat de logica die nodig is om de juiste gegevens op het etiket te plaatsen. Hiervoor worden twee objecten gebruikt: Bestelling en Etiket.

Het object Bestelling valideert de gegevens afkomstig uit het Excel-bestand. Deze gegevens omvatten de productnaam, de gebruiker, het commentaar en het veld BesteldPer, dat het aantal aanduidt.

Het object Etiket bevat de logica voor het opbouwen van het etiket. Dit gebeurt door het tekenen van een bitmap waarin alle informatie en de volledige structuur van het etiket verwerkt worden.

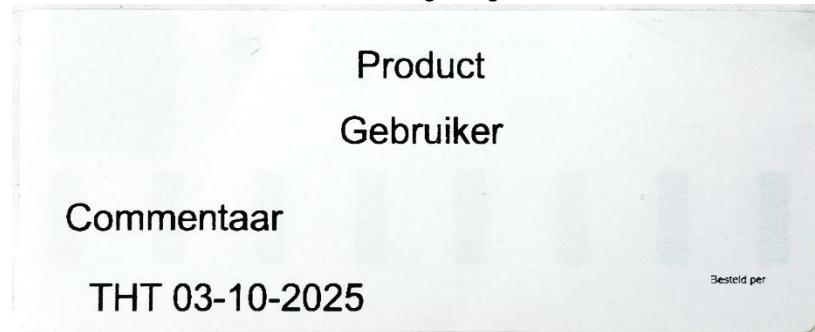
4.1.5.3. Presentatie-laag

De presentatie-laag bestaat uit een gebruiksvriendelijke Windows Forms-interface waarmee keukenmedewerkers hun Excel-bestand kunnen importeren en de bestellingen kunnen afdrukken op etiketten.

Deze interface bevat:

- Een knop om een Excel-file te importeren
- Een tabel voor de Excel data
- Een knop om de etiketten af te drukken
- Checkboxen bij elke rij om te kiezen welk afgedrukt worden
- Een knop om alle checkboxen aan te vinken
- Een knop om alle checkboxen uit te vinken

Voor het etiket heb ik een voorbeeld gekregen van hoe het eruit moet zien:



Afbeelding 26: Voorbeeld keuken etiket

4.1.6. Technologieën

Voor dit project is gebruikgemaakt van C# in combinatie met Windows Forms voor de ontwikkeling van de desktopapplicatie. De verwerking en validatie van Excel-bestanden werd gerealiseerd met de ExcelDataReader-bibliotheek.

4.2. Realisatie

4.2.1. Doel

Het doel van dit project was een snelle, foutloze en geautomatiseerde manier ontwikkelen om broodjesetiketten te genereren. Hierdoor kan de keuken efficiënter werken, zeker tijdens piekmomenten.

4.2.2. Functionaliteiten

De applicatie biedt een reeks praktische functies, waaronder:

- Het selecteren en inlezen van een Excel-bestand.
- Het automatisch verwerken van alle bestellingen.
- Het genereren van etiketten op basis van naam en broodje(s).
- Een overzicht van alle ingeladen bestellingen voor controle.
- De mogelijkheid om etiketten onmiddellijk af te drukken.

4.2.3. 3-tier architectuur

Dit project heeft geen duidelijke 3-tier architectuur, wel zijn alle functionaliteiten gescheiden gehouden om overzicht en foutverwerking te garanderen.

4.2.3.1. Business-laag

De business-laag bevat de logica rond bestellingen en etiketten. Elk ingelezen item wordt omgezet naar een intern Order-object met daarin alle noodzakelijke gegevens. Deze laag beslist hoe de informatie wordt omgezet naar een labelopmaak en controleert op eventuele fouten of ontbrekende waarden.

```
C# ObjBestelling.cs  
C# ObjEtiket.cs
```

ObjBestelling.cs:

```
public class ObjBestelling  
{  
    1 reference  
    public ObjBestelling() { }  
  
    6 references  
    public string Product { get; set; }  
    6 references  
    public string Gebruiker { get; set; }  
    3 references  
    public string Commentaar { get; set; }  
    4 references  
    public int? BesteldPer { get; set; }  
  
    1 reference  
    public bool IsValid(out string errorText)...
```

Afbeelding 27: Object Bestelling

ObjEtiket.cs:

```
public class ObjEtiket
{
    private static readonly Font FontLarge = new Font("Arial", 13);
    private static readonly Font FontSmall = new Font("Arial", 6);
    private static readonly Brush textColor = Brushes.Black;

    private static readonly float EtiketBreedteMm = 86f;
    private static readonly float EtiketHoogteMm = 30f;

    1 reference
    public static void PrintEtiketDymo(ObjBestelling artikel, int kopieIndex = 1, int totaal = 1)...

    1 reference
    private static Bitmap CreateEtiketBitmapDymo(ObjBestelling artikel, int kopieIndex, int totaal)...

    1 reference
    private static void CreateTicketDymo(Graphics g, float breedte, float hoogte, ObjBestelling artikel, int kopieIndex, int totaal)...

    2 references
    private static string GetSelectedPrinter()...

    0 references
    public static void PrintEtiketToshiba(ObjBestelling bestelling, int kopieIndex = 1, int totaal = 1)...

    1 reference
    private static void CreateTicketToshiba(PrintPageEventArgs e, ObjBestelling bestelling, int kopieIndex, int totaal)...

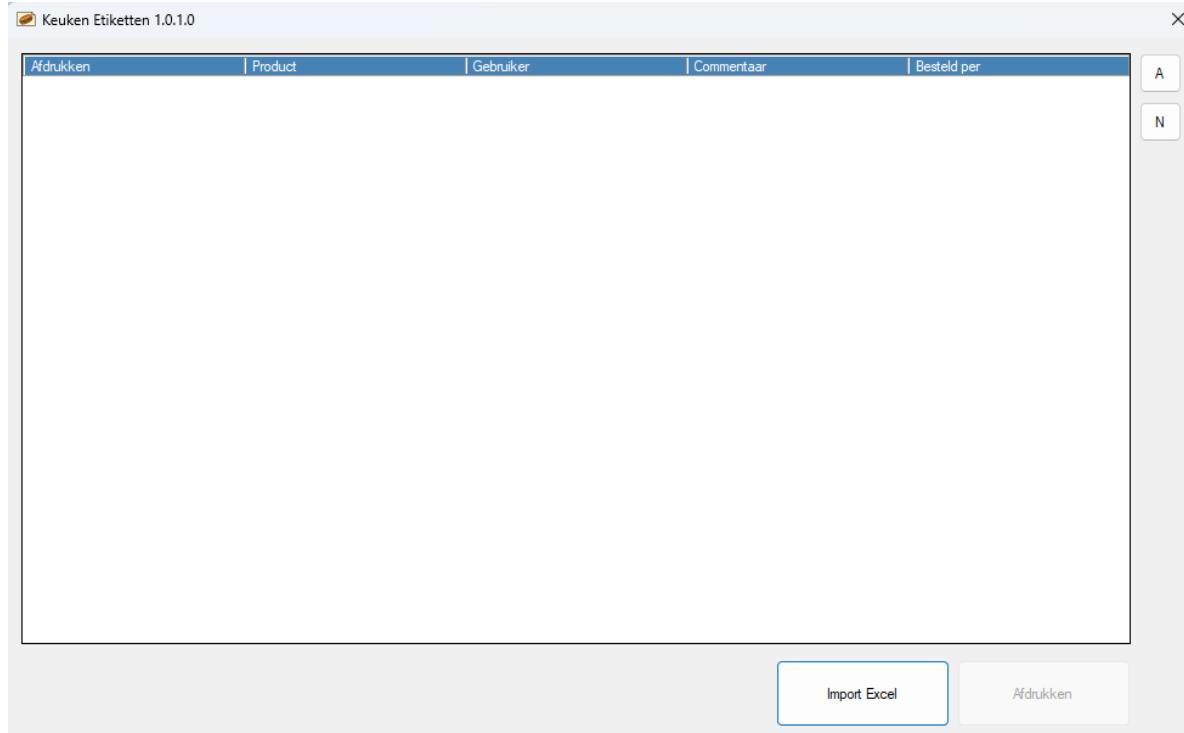
    2 references
    private static int MmToHundredths(float mm)...
}
```

Afbeelding 28: Object Etiket

4.2.3.2. Presentatie-laag

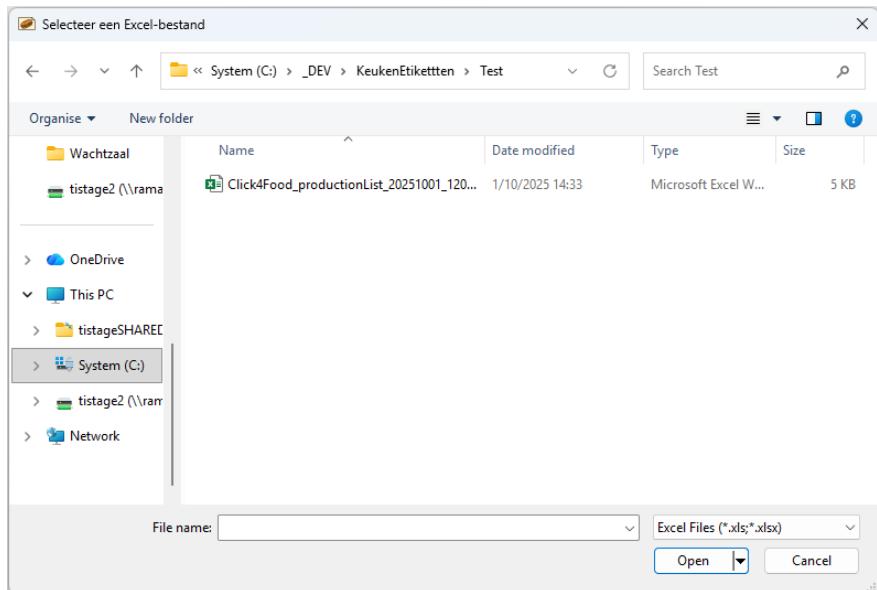
De presentatie-laag bestaat uit een Windows Forms-interface waar gebruikers het Excel-bestand kunnen selecteren, de ingelezen bestellingen kunnen bekijken en etiketten kunnen genereren en printen. De UI is ontworpen om zo eenvoudig mogelijk te zijn, zodat keukenmedewerkers zonder technische kennis ermee kunnen werken.

Startscherm (lege tabel):



Afbeelding 29: Startscherm Keuken Etiketten (lege tabel)

Bij klikken "Import Excel" opent volgende pop-up:



Afbeelding 30: Excel importeren scherm

Bij selecteren Excel-file krijg je volgende scherm.

Startscherm (gevulde tabel):

Hier is nog de mogelijkheid om aan te vinken welke etiketten je wil afdrukken, standaard is alles aangevinkt.

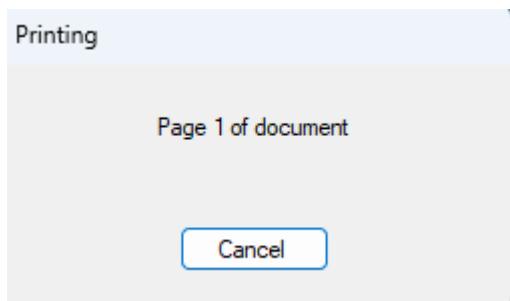
Afdrukken	Product	Gebruiker	Commentaar	Besteld per
<input checked="" type="checkbox"/>	Salade van de dag ASO			1
<input checked="" type="checkbox"/>	Spaghetti Bolognaise ASO			1
<input checked="" type="checkbox"/>	1/2 stokbrood Martino		niet pikant	2
<input checked="" type="checkbox"/>	Middagmaal ASO			1
<input checked="" type="checkbox"/>	Salade van de dag ASO			1
<input checked="" type="checkbox"/>	Middagmaal ASO			1
<input checked="" type="checkbox"/>	Salade van de dag ASO			1

A
N

Import Excel **Afdrukken**

Afbeelding 31: Startscherm Keuken Etiketten (gevulde tabel)

Bij klikken "Afdrukken" krijg je volgende pop-up en worden de etiketten afgedrukt.



Afbeelding 32: Etiket afdrukken scherm

4.2.4. Uitdagingen

Het project verliep aanvankelijk zeer vlot, waarbij de grootste uitdaging in eerste instantie de Excel Data Reader leek te zijn. Tijdens het testen op de etikettenprinter in de keuken bleek echter dat de rotatie van het etiket niet correct was, zoals zichtbaar op de onderstaande afbeelding.



Afbeelding 33: Foutief keuken etiket

Het oplossen van dit probleem bleek aanzienlijk complexer dan verwacht. Niet alleen de rotatie veroorzaakte afwijkingen, maar ook de werking van de printer verschilde van de printer die in het project *Magazijn Etiketten* werd gebruikt. Uiteindelijk is het probleem verholpen door het etiket volledig als bitmap te tekenen en deze rechtstreeks af te drukken. Hierdoor werd de volledige afbeelding consistent en correct op het etiket geplaatst.

4.2.5. Eindresultaat

Het eindresultaat is een gebruiksvriendelijke en stabiele Windows Forms-applicatie die de keuken toelaat om binnen enkele seconden etiketten af te drukken vanuit een Excel-bestand met broodjesbestellingen. Het proces verloopt veel sneller, de foutmarge is sterk verminderd en de workflow van de keuken is merkbaar efficiënter geworden.

Onderstaand een voorbeeld van een etiket dat met de applicatie werd geprint:



Afbeelding 34: Resultaat keuken etiket

5. Kledij uitleningen

5.1. Analyse

5.1.1. Klant

De klant voor dit project is de linnenkamer van het ziekenhuis. Zij zijn verantwoordelijk voor het beheren, verdelen en opvolgen van alle kledingstukken die personeel nodig heeft tijdens de uitoefening van hun functie. Door het ontbreken van een digitaal registratiesysteem is een moderne en efficiënte oplossing noodzakelijk.

5.1.2. Huidige situatie

Binnen de linnenkamer wordt momenteel niet geregistreerd welke personeelsleden welke kledingstukken meenemen. Medewerkers kunnen meerdere stukken meenemen zonder administratieve opvolging. Hierdoor ontbreekt elk overzicht van waar bepaalde kledingstukken zich bevinden en wie deze in bezit heeft.

5.1.3. Probleem

Door het ontbreken van een registratiesysteem ontstaan verschillende operationele problemen. Sommige medewerkers nemen te veel kledingstukken mee, waardoor er tekorten kunnen ontstaan voor anderen. Bij beschadigde of verdwenen kledingstukken is niet te achterhalen wie het kledingstuk als laatste had, waardoor er geen transparantie of controle is. Dit leidt tot inefficiënties en extra werk voor de linnenkamer.

5.1.4. Functionele eisen

De applicatie moet toelaten om kledingstukken te registreren wanneer ze worden uitgeleend en teruggebracht. Scannen gebeurt via barcodes op zowel de kledingstukken als de personeelsbadges. Daarnaast moet de applicatie een duidelijk overzicht bieden van lopende uitleningen en de geschiedenis van uitgeleende kledingstukken.

5.1.4.1. Use cases

Use case 1

Naam: Kleren uitscannen

Actors: Linnenkamer personeel en ziekenhuis personeel

Functionaliteit: Als werknemer kan ik kledingstukken uitlenen

Voorwaarden: Als werknemer gebruik ik mijn eigen badge om mezelf te linken aan een kledingstuk

Normal flow:

Het systeem toont een scherm met invulvelden voor kledingnummer en badgenummer. De linnenkamer werknemer neemt een kledingstuk en scant de code van dat kledingstuk. De werknemer scant zijn badge. De kleding en werknemer worden gelinkt en nu mag de werknemer het kledingstuk meenemen.

Uitzondering:

De werknemer heeft al kleding uitgeleend: Het systeem toont een melding dat de werknemer nog kleding in zijn/haar bezit heeft en de vraag om toch uit te lenen. De linnenkamer medewerker beslist om alsnog uit te lenen of niet.

Use case 2

Naam: Kledij uitscannen

Actors: Linnenkamer personeel

Functionaliteit: Als linnenkamer werknemer kan ik kledij uitscannen

Voorwaarden: Het kledingstuk is binnengebracht door het personeelslid of zat bij de propere was.

Normal flow:

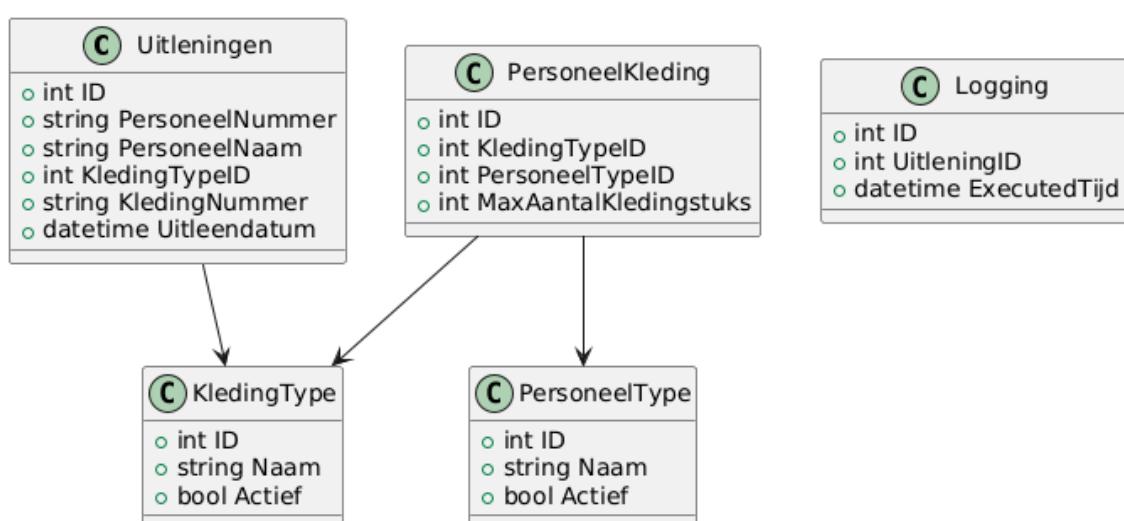
Het systeem toont een scherm met een invulveld voor de kledingnummer. De linnenkamer medewerker scant de kledingnummer. Het systeem verwijdert het record van de werknemer en kledingstuk uit de database.

Uitzondering: /

5.1.5. Technische eisen

De applicatie moet betrouwbaar, performant en gebruiksvriendelijk zijn. Alle registratie moet plaatsvinden in een centrale SQL Server-database. Barcodes moeten snel en zonder fout gescand kunnen worden.

5.1.5.1. Data-laag



Afbeelding 35: Data klasse kledij uitleningen

De data-laag bestaat uit een SQL Server-database die alle informatie over kledinguitleningen structureert. De centrale tabel **Uitleningen** registreert welke medewerker welk kledingstuk heeft meegenomen, inclusief het kledingtype, kledingnummer en de uitleendatum. Via **KledingTypeID** is elke uitlening gekoppeld aan de tabel **KledingType**, waarin alle soorten kledingstukken worden beheerd en aangegeven wordt of een type nog actief is.

Daarnaast bevat de database de tabellen PersoneelType en PersoneelKleding, die samen bepalen hoeveel kledingstukken elk type medewerker in de toekomst maximaal zal mogen uitlenen. Hoewel deze tabellen in de huidige versie van de applicatie nog niet actief gebruikt worden, vormen ze een belangrijke basis voor toekomstige uitbreidingen waarin uitleenlimieten automatisch gecontroleerd kunnen worden.

De tabel Logging registreert tot slot elke uitgevoerde actie, gekoppeld aan een specifieke uitlening. Dit maakt het mogelijk om een historiek op te bouwen en eventuele fouten of misbruik op te volgen.

5.1.5.2. Business-laag

De business-laag bevat alle kernlogica die bepaalt hoe uitleningen worden verwerkt en gevalideerd. Het centrale object binnen deze laag is KledingUitlening, dat de gegevens bundelt van elke uitlening: het kledingstuk, de medewerker die het meeneemt, de naam van de medewerker en de datum van uitlening. Dit object vormt de basis voor het controleren, registreren en weergeven van elke handeling in de applicatie.

Daarnaast bevat de business-laag de objecten KledingType, PersoneelType en PersoneelKleding, die samen de structuur vormen om regels rond uitleningen te ondersteunen. KledingType beschrijft het type kledingstuk en of het type momenteel actief is. PersoneelType definieert toekomstige categorieën van medewerkers, waarbij kan worden aangegeven welke types personeel toegang hebben tot welke kledingstukken. Het object PersoneelKleding koppelt deze twee concepten aan elkaar en bepaalt hoeveel stuks een bepaald type medewerker maximaal mag lenen.

Hoewel PersoneelType en PersoneelKleding in deze eerste versie nog niet actief gebruikt worden, liggen ze reeds klaar om in toekomstige uitbreidingen automatisch limieten en controles in te bouwen. Hierdoor blijft de business-laag schaalbaar en voorbereid op verdere groei of veranderende noden binnen de linnenkamer.

5.1.5.3. Presentatie-laag

De presentatie-laag bestaat uit een gebruiksvriendelijke Windows Forms-interface die het volledige proces van kleding uitlenen en terugbrengen ondersteunt. De interface is ontworpen om medewerkers in de linnenkamer zo snel mogelijk te laten werken met minimale handelingen.

Bij dit project zijn enkele simpele mock-ups om een beeld te hebben hoe het er ongeveer moet gaan uitzien.

Startscherm:

The mock-up shows a user interface for managing clothing loans. At the top, there is a search bar labeled "Filters: Personeelnaam Uitleendatum van/tot". Below this is a table with columns: PersoneelNummer, PersoneelNaam, KledingType, KledingNummer, Uitleendatum, and several buttons on the right: "Nieuw", "Open", and "Verwijderen".

PersoneelNummer	PersoneelNaam	KledingType	KledingNummer	Uitleendatum	Nieuw
					Open
					Verwijderen

Afbeelding 36: Mock-up overzicht scherm kledij uitleningen

Scherm Nieuwe Uitlening:

Personeelnummer:	<input type="text"/>
Personeelnaam:	<input type="text"/>
Kledingtype:	<input type="text"/>
Kledingtype:	<input type="text"/>
<input type="button" value="Annuleer"/> <input type="button" value="Doorgaan"/>	

Afbeelding 37: Mock-up nieuwe uitlening scherm

Scherm Kledij Inleveren:

Kledingnummer:	<input type="text"/>
<input type="button" value="Annuleren"/> <input type="button" value="Doorgaan"/>	

Afbeelding 38: Mock-up kledij inleveren scherm

5.1.6. Technologieën

Voor dit project wordt gebruikgemaakt van C# in combinatie met Windows Forms als front-end technologie. De data opslag gebeurt in SQL Server. Barcodes worden verwerkt via een standaard barcode scanner die fungeert als invoerapparaat binnen de applicatie.

5.2. Realisatie

5.2.1. Doel

Het doel van dit project is het creëren van een overzichtelijk en betrouwbaar systeem voor het registreren van uitgeleende en teruggebrachte kledingstukken. Hierdoor ontstaat transparantie, minder verlies en een hogere efficiëntie binnen de linnenkamer.

5.2.2. Functionaliteiten

De applicatie biedt:

- Automatische herkenning van personeelsbadges via barcode scanning.
- Scannen van kleding
- Registratie van uitleningen en terugnames.
- Realtime overzicht van alle uitgeleende kledingstukken.
- Een foutbestendige workflow die menselijke fouten minimaliseert.

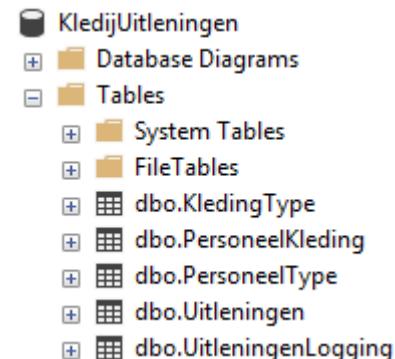
5.2.3. 3-tier architectuur

De applicatie werd gemaakt met een duidelijke 3-tier architectuur om een scheiding te creëren tussen data opslag, logica en presentatie. Zo wordt het maken van extra toevoegingen in de toekomst eenvoudiger.

5.2.3.1. Data-laag

De data-laag bevat klassen en methoden die communiceren met de SQL Server-database. Dit omvat het ophalen van medewerkers, kledingstukken en uitleningen, evenals het toevoegen en bijwerken van uitleenrecords.

SQL Server structuur:



Afbeelding 39: SQL database structuur

dbo.Uitleningen:

	Column Name	Data Type	Allow Nulls
!	ID	int	<input type="checkbox"/>
	PersoneelNummer	nchar(25)	<input checked="" type="checkbox"/>
	PersoneelNaam	nvarchar(250)	<input checked="" type="checkbox"/>
	KledingTypeID	int	<input checked="" type="checkbox"/>
	KledingNummer	nvarchar(250)	<input checked="" type="checkbox"/>
	Uitleendatum	datetime	<input checked="" type="checkbox"/>
	InsertedOn	datetime	<input checked="" type="checkbox"/>
	InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
	UpdatedOn	datetime	<input checked="" type="checkbox"/>
	UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 40: Database tabel Uitleningen

dbo.KledingType:

	Column Name	Data Type	Allow Nulls
!	ID	int	<input type="checkbox"/>
	Naam	nvarchar(250)	<input checked="" type="checkbox"/>
	Actief	bit	<input checked="" type="checkbox"/>
	InsertedOn	datetime	<input checked="" type="checkbox"/>
	InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
	UpdatedOn	datetime	<input checked="" type="checkbox"/>
	UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 41: Database tabel KledingType

dbo.PersoneelType:

	Column Name	Data Type	Allow Nulls
!	ID	int	<input type="checkbox"/>
	Naam	nvarchar(250)	<input checked="" type="checkbox"/>
	Actief	bit	<input checked="" type="checkbox"/>
	InsertedOn	datetime	<input checked="" type="checkbox"/>
	InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
	UpdatedOn	datetime	<input checked="" type="checkbox"/>
	UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 42: Database tabel PersoneelType

dbo.PersoneelKleding:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
KledingTypeID	int	<input checked="" type="checkbox"/>
PersoneelTypeID	int	<input checked="" type="checkbox"/>
MaxAantalKledingstuks	int	<input checked="" type="checkbox"/>
InsertedOn	datetime	<input checked="" type="checkbox"/>
InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
UpdatedOn	datetime	<input checked="" type="checkbox"/>
UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 43: Database tabel PersoneelKleding

dbo.Logging:

Column Name	Data Type	Allow Nulls
ID	int	<input type="checkbox"/>
UitleningID	int	<input checked="" type="checkbox"/>
ExecutedTijd	datetime	<input checked="" type="checkbox"/>
SQLCommandText	nvarchar(MAX)	<input checked="" type="checkbox"/>
InsertedOn	datetime	<input checked="" type="checkbox"/>
InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
UpdatedOn	datetime	<input checked="" type="checkbox"/>
UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 44: Database tabel Logging

5.2.3.2. Business-laag

De business-laag bepaalt hoe scans worden geïnterpreteerd. Wanneer een barcode wordt gescand, wordt hier beslist of het om een medewerker of een kledingstuk gaat en welke actie moet worden uitgevoerd. Ook worden validatieregels toegepast, zoals dubbele uitleningen of foutieve terugnames.

```
C# ObjKledingType.cs  
C# ObjPersoneelKleding.cs  
C# ObjPersoneelType.cs  
C# ObjUitlening.cs
```

ObjUitlening:

```
public class ObjUitlening
{
    1 reference
    public ObjUitlening()
    {
        ID = null;
    }

    1 reference
    public ObjUitlening(long id)
    {
        ID = id;
        Load();
    }

    12 references
    public long? ID { get; set; }
    9 references
    public string PersoneelNummer { get; set; }
    11 references
    public string PersoneelNaam { get; set; }
    10 references
    public string KledingNummer { get; set; }
    5 references
    public DateTime? Uitleendatum { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    1 reference
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    2 references
    public bool IsValid(out string errorText, out int uitleningenCount)...
    1 reference
    public bool Load()...
    2 references
    public bool Save(out string errorText, out int uitleningenCount)...
    0 references
    public bool Delete(out string errorText)...
}
```

Afbeelding 45: Object Uitlening

ObjKledingType:

```
public class ObjKledingType
{
    0 references
    public ObjKledingType()
    {
        ID = null;
    }

    0 references
    public ObjKledingType(long id)
    {
        ID = id;
        Load();
    }

    7 references
    public long? ID { get; set; }
    4 references
    public string Naam { get; set; }
    2 references
    public bool? Actief { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    1 reference
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    1 reference
    public bool IsValid (out string errorText)...

    1 reference
    public bool Load()...

    0 references
    public bool Save(out string errorText)...
}
```

Afbeelding 46: Object KledingType

ObjPersoneeelType:

```
public class ObjPersoneeelType
{
    0 references
    public ObjPersoneeelType()
    {
        ID = null;
    }

    0 references
    public ObjPersoneeelType(long id)
    {
        ID = id;
        Load();
    }

    7 references
    public long? ID { get; set; }
    4 references
    public string Naam { get; set; }
    2 references
    public bool? Actief { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    1 reference
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    1 reference
    public bool IsValid (out string errorText)...
}

1 reference
public bool Load()...
}

0 references
public bool Save(out string errorText)...
}
```

Afbeelding 47 Object PersoneeelType

ObjPersoneelKleding:

```
public class ObjPersoneelKleding
{
    0 references
    public ObjPersoneelKleding()
    {
        ID = null;
    }

    0 references
    public ObjPersoneelKleding(long id)
    {
        ID = id;
        Load();
    }

    6 references
    public long? ID { get; set; }
    4 references
    public long? PersoneelTypeID { get; set; }
    4 references
    public long? KledingTypeID { get; set; }
    4 references
    public int? MaxAantalKledingstuks { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    1 reference
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    1 reference
    public bool IsValid(out string errorText)...
}

1 reference
public bool Load()...
}

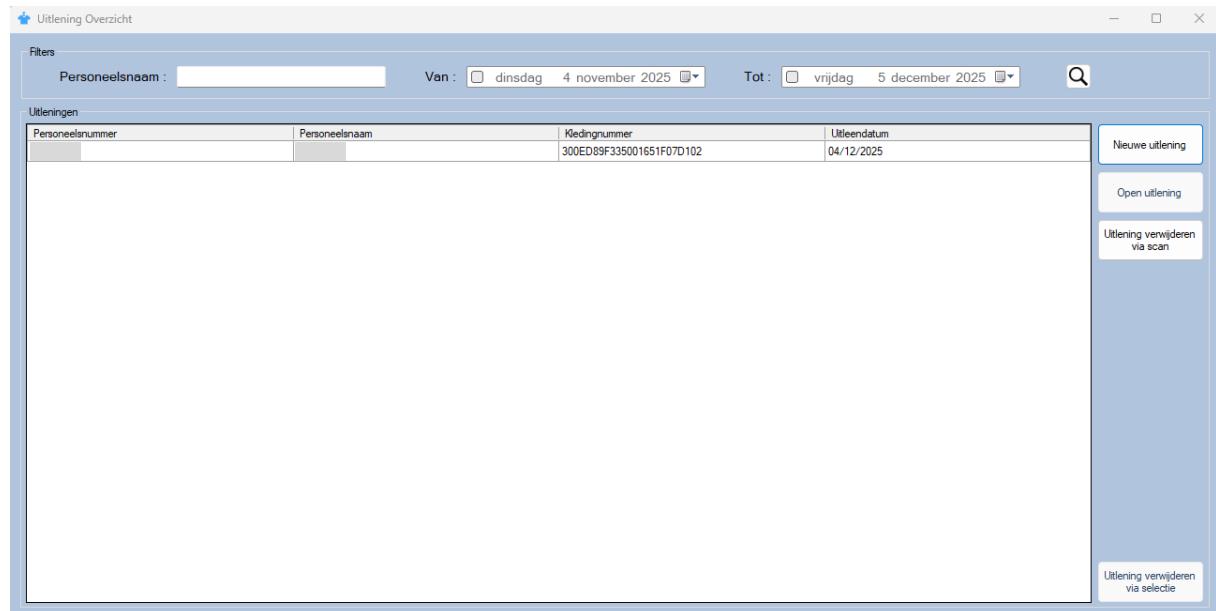
0 references
public bool Save(out string errorText)...
}
```

Afbeelding 48: Object PersoneelKleding

5.2.3.3. Presentatie-laag

De presentatie-laag biedt een vlot te gebruiken interface met duidelijke knoppen, tabellen en meldingen. De workflow is bewust eenvoudig gehouden zodat de linnenkamer zonder opleiding met het systeem kan werken.

Startscherm:

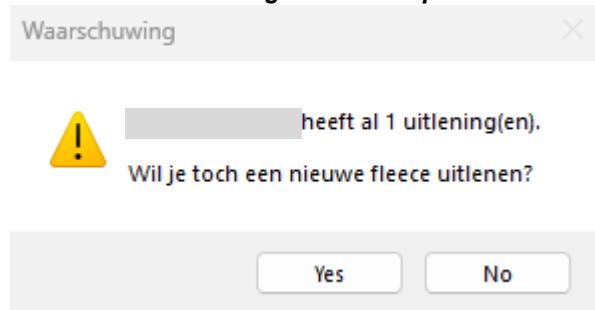


Afbeelding 49: Startscherm Kledij Uitleningen

Nieuwe uitlening scherm:

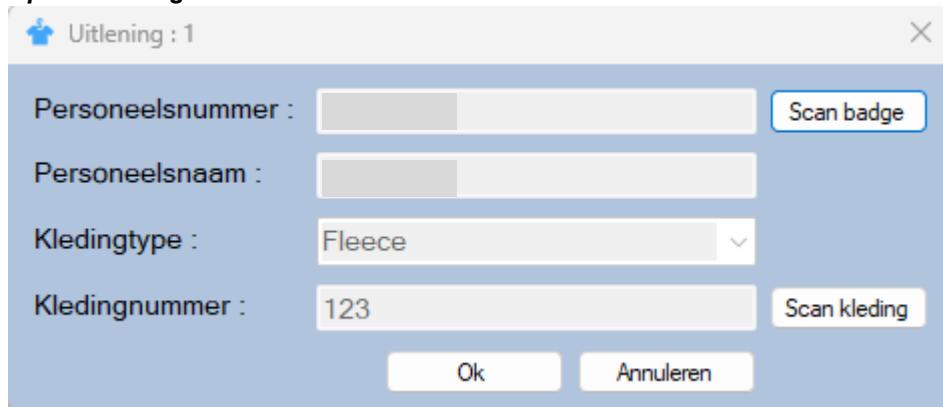
Afbeelding 50: Nieuwe uitlening scherm

Als de werknemer nog 1 of meer openstaande uitleningen heeft:



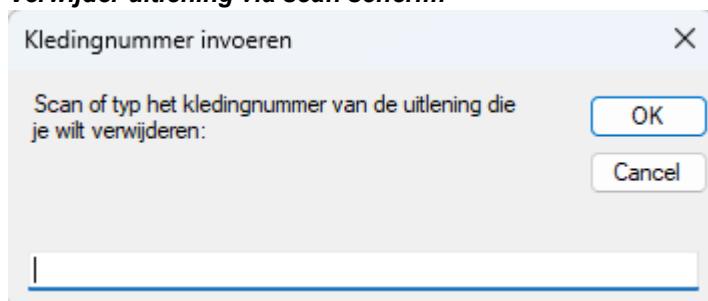
Afbeelding 51: Waarschuwing meerdere kledingstukken

Open uitlening scherm:



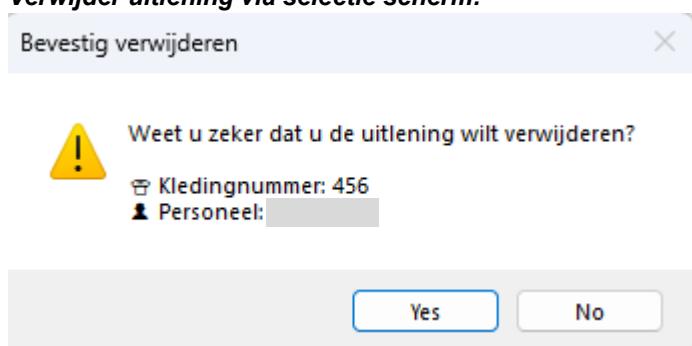
Afbeelding 52: Open uitlening scherm

Verwijder uitlening via scan scherm:



Afbeelding 53: Verwijder uitlening via scan scherm

Verwijder uitlening via selectie scherm:



Afbeelding 54: Bevestiging uitlening verwijderen

5.2.4. Uitdagingen

De grootste uitdagingen in dit project waren onder andere het opzetten van een correcte database structuur en de applicatie gebruiksvriendelijk houden zodat de linnenkamer medewerkers er zonder problemen mee overweg kunnen.

5.2.5. Eindresultaat

Het eindresultaat is een stabiele en efficiënte Windows Forms-applicatie waarmee de linnenkamer op een gebruiksvriendelijke manier kledingstukken kan beheren. Het systeem biedt een volledig overzicht van alle uitleningen, vermindert fouten en verhoogt de transparantie. Hierdoor beschikt het ziekenhuis voortaan over een professioneel en betrouwbaar registratiesysteem voor kledij-uitleningen.

6. Badge uitleningen

6.1. Analyse

6.1.1. Klant

De klant is de technische dienst van het ziekenhuis, die verantwoordelijk is voor de techniek in het ziekenhuis en het beheer, de uitlening en de opvolging van toegangsbadges.

6.1.2. Huidige situatie

Badges worden momenteel handmatig uitgeleend en op papier of in Excel geregistreerd. Er wordt niet automatisch bijgehouden wie een badge in gebruik heeft, wanneer deze werd uitgeleend of wanneer deze moet worden teruggebracht. Hierdoor ontbreekt een betrouwbaar overzicht.

6.1.3. Probleem

Het manuele proces is foutgevoelig en inefficiënt. Er zijn regelmatig situaties waarbij badges onvindbaar zijn of te lang in omloop blijven zonder dat de verantwoordelijke medewerker dit weet. Er is geen centrale logging, waardoor het moeilijk is om na te gaan wie een badge als laatste in bezit had.

6.1.4. Functionele eisen

Functionele eisen beschrijven welke acties en functies de applicatie moet ondersteunen vanuit het perspectief van de gebruiker.

6.1.4.1. Use cases

Use case 1

Naam: Uitlening aanmaken

Actors: Externe werknemers

Functionaliteit: Als werknemer kan ik een badge van de technische dienst uitlenen

Voorwaarden: De badge is niet meer gelinkt aan een openstaande uitlening

Normal flow:

De werknemer scant de badge die hij gekregen heeft van de technische dienst. Het systeem toont een scherm met invulvelden om info in te vullen. De werknemer vult zijn informatie in en drukt op "Ok". Het systeem start een nieuwe uitlening.

Uitzondering: /

Use case 2:

Naam: Uitlening stoppen

Actors: Externe werknemers

Functionaliteit: Als werknemer kan ik een badge van de technische dienst terugbrengen

Voorwaarden: /

Normal flow:

De werknemer scant de badge. Het systeem herkent dat er een openstaande uitlening was en beëindigt de uitlening. De werknemer geeft de badge terug aan de technische dienst.

Uitzondering:

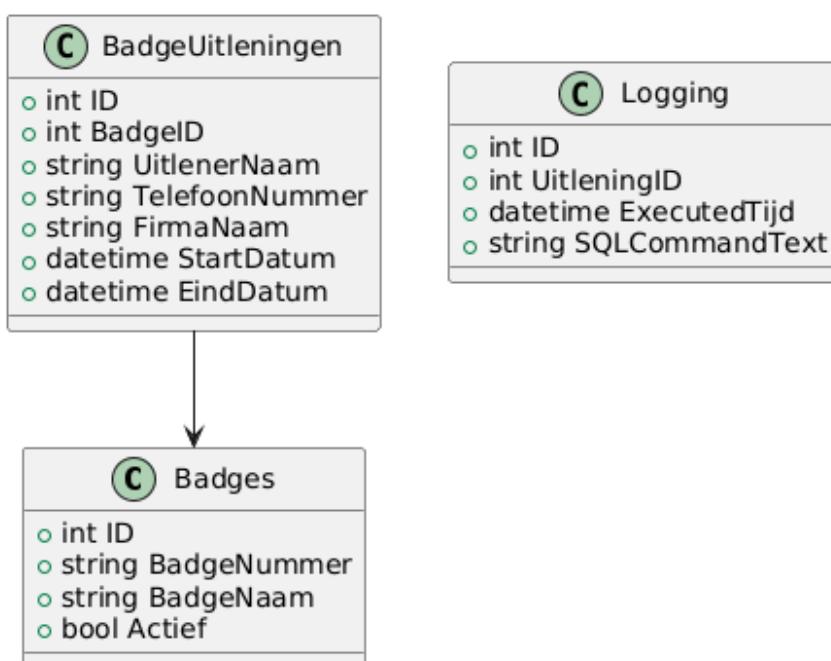
Als een werknemer de badge fysiek niet terugbrengt zit zijn info in het systeem en kan er achterhaald worden wie de badge niet teruggestaan heeft.

6.1.5. Technische eisen

Technische eisen beschrijven hoe de applicatie moet functioneren op technisch vlak. Ze leggen vast welke systemen, databases, architectuur en beperkingen nodig zijn om de software betrouwbaar, performant en onderhoudbaar te laten werken.

6.1.5.1. Data-laag

De data-laag bestaat uit drie kernetiteiten. De tabel Badges bewaart alle beschikbare badges, inclusief hun nummer, naam en de status of ze actief zijn. De tabel BadgeUitleningen registreert elke uitlening met gegevens zoals de betrokken badge, de naam van de uitlener, het bedrijf, het telefoonnummer en de start- en einddatum van de uitlening, hierdoor kan de applicatie bepalen of een badge momenteel in gebruik is. Tot slot legt de tabel Logging elke uitgevoerde actie vast, inclusief het tijdstip en de gebruikte SQL-opdracht, zodat alle bewerkingen traceerbaar blijven en administratieve opvolging mogelijk is.



Afbeelding 55: Data UML Badge Uitleningen

6.1.5.2. Business-laag

De business-laag bevat de logica voor het interpreteren van scans, het aanmaken of afsluiten van uitleningen en het valideren van badgegegevens.

De 2 objecten zijn Badge en BadgeUitlening. Het Badge-object beschrijft een badge met een ID, een uniek nummer, een naam en de status of deze actief is. Het BadgeUitlening-object stelt een uitlening voor en bevat een ID, een verwijzing naar de betrokken badge via BadgID, de naam en het telefoonnummer van de persoon die de badge leent, de firmanaam, en de start- en stopperiode van de uitlening.

6.1.5.3. Presentatie-laag

De presentatielaag bestaat uit twee gebruikersinterfaces:

- **WinForms-configuratiescherm:**

Dit scherm maakt het mogelijk om nieuwe badges toe te voegen, bestaande badges te beheren en een overzicht van alle uitleningen te bekijken. Er kan gefilterd worden op geschiedenis en huidige situatie.

Overzicht scherm:

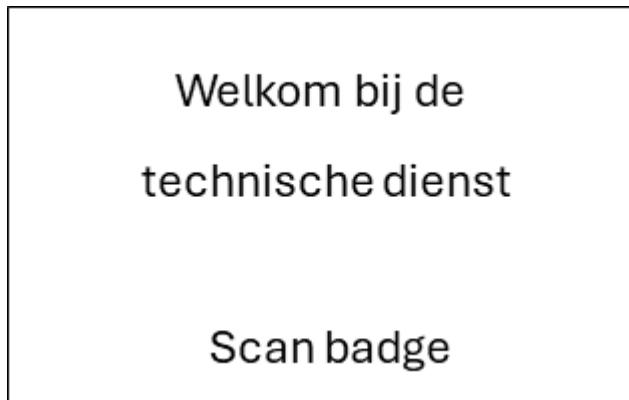
The screenshot shows a Windows application window titled 'Overzicht scherm: Badge Uitleningen configuratie'. At the top left is a filter input labeled 'Filter op: Huidige situatie'. Below it is a search bar labeled 'Geschiedenis: badge, technieker, firma, datum'. A table lists badge details: Badge, technieker, Firma, Gsm, Startdatum, Einddatum. To the right of the table are three buttons: 'Nieuwe badge', 'Open badge', and 'Verwijderen'.

Afbeelding 56: Overzicht scherm Badge Uitleningen configuratie

- **ASP.NET webapplicatie op Zebra-tablet:**

Dit is een eenvoudige webinterface die geoptimaliseerd is voor touchbediening en barcode scanning. Na het scannen toont het scherm automatisch of de badge wordt uitgeleend of teruggebracht. De gebruiker hoeft slechts minimale gegevens in te vullen, waardoor het proces zeer snel verloopt.

Startscherm:



Afbeelding 57: Startscherm Badge Uitleningen

Badge uitlenen scherm:

The image shows a rectangular form with a white background. On the left side, there are five input fields with labels: "Badgenummer:" (with value "000000"), "Badge naam:" (with value "Badge 1"), "Uw Firma:" (empty), "Uw Naam:" (empty), and "GSM nummer:" (empty). At the bottom of the form are two buttons: "Annuleer" (left) and "Doorgaan" (right). The entire form is enclosed in a thin black border.

Afbeelding 58: Badge uitlenen scherm

6.1.6. Technologieën

Voor dit project werd een combinatie van Windows Forms en ASP.NET Web Forms gebruikt, aangevuld met een SQL Server-database voor centrale opslag. De Zebra-tablets gebruiken een ingebouwde barcode scanner die via de web interface wordt aangesproken. De applicaties zijn ontwikkeld in C# en ASPX voor het web gedeelte.

6.2. Realisatie

6.2.1. Doel

Het doel van de realisatie is het implementeren van een betrouwbaar, efficiënt en gebruiksvriendelijk systeem voor de uitlening van badges binnen het ziekenhuis. Het systeem moet handmatige registratie vervangen, real-time inzicht bieden in de status van badges en de traceerbaarheid van uitleningen verbeteren.

6.2.2. Functionaliteiten

Het systeem biedt de volgende kernfunctionaliteiten:

- **Badge uitlenen:** Externe werknemers kunnen badges scannen en hun gegevens registreren, waarna een uitlening wordt aangemaakt.
- **Badge terugbrengen:** Bij het terugbrengen van de badge wordt de uitlening automatisch afgesloten en wordt de status van de badge bijgewerkt.
- **Overzicht en rapportage:** De technische dienst kan inzien welke badges in gebruik zijn, welke teruggebracht moeten worden en historische uitleningen raadplegen.
- **Logging en traceerbaarheid:** Alle acties worden centraal geregistreerd, inclusief tijdstempels, zodat achteraf gecontroleerd kan worden wie een badge in gebruik had.

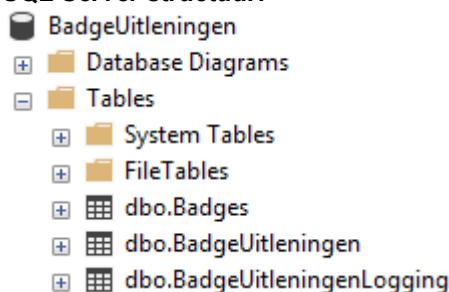
6.2.3. 3-tier architectuur

Het project is opgebouwd volgens een 3-tier architectuur, waarbij de functionaliteit is opgesplitst in drie lagen: data-laag, business-laag en presentatie-laag. Deze structuur zorgt voor een duidelijke scheiding van verantwoordelijkheden, maakt onderhoud eenvoudiger, verhoogt de betrouwbaarheid en maakt toekomstige uitbreidingen eenvoudiger. De 2 applicaties maken gebruik van één gemeenschappelijke data- en business-laag en een aparte presentatie-laag.

6.2.3.1. Data-laag

De data-laag vormt de kern van de applicatie en is verantwoordelijk voor het opslaan, ophalen en beheren van gegevens. In deze laag bevinden zich de databases en de bijbehorende tabellen.

SQL Server structuur:



Afbeelding 59: SQL database structuur

dbo.Badges:

	Column Name	Data Type	Allow Nulls
Y	ID	int	<input type="checkbox"/>
	BadgeNummer	nvarchar(25)	<input checked="" type="checkbox"/>
	BadgeNaam	nvarchar(250)	<input checked="" type="checkbox"/>
	Actief	bit	<input checked="" type="checkbox"/>
	InsertedOn	datetime	<input checked="" type="checkbox"/>
	InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
	UpdatedOn	datetime	<input checked="" type="checkbox"/>
	UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 60: Database tabel Badges

dbo.BadgeUitleningen:

	Column Name	Data Type	Allow Nulls
Y	ID	int	<input type="checkbox"/>
	BadgeID	int	<input checked="" type="checkbox"/>
	UitlenerNaam	nvarchar(250)	<input checked="" type="checkbox"/>
	Telefoonnummer	nvarchar(250)	<input checked="" type="checkbox"/>
	FirmaNaam	nvarchar(250)	<input checked="" type="checkbox"/>
	StartDatum	datetime	<input checked="" type="checkbox"/>
	EindDatum	datetime	<input checked="" type="checkbox"/>
	InsertedOn	datetime	<input checked="" type="checkbox"/>
	InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
	UpdatedOn	datetime	<input checked="" type="checkbox"/>
	UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 61: Database tabel BadgeUitleningen

dbo.BadgeUitleningenLogging:

	Column Name	Data Type	Allow Nulls
	ID	int	<input checked="" type="checkbox"/>
	BadgeUitleningenID	int	<input checked="" type="checkbox"/>
	ExecutedTijd	datetime	<input checked="" type="checkbox"/>
	SQLCommandText	nvarchar(MAX)	<input checked="" type="checkbox"/>
	InsertedOn	datetime	<input checked="" type="checkbox"/>
	InsertedBy	nchar(50)	<input checked="" type="checkbox"/>
	UpdatedOn	datetime	<input checked="" type="checkbox"/>
	UpdatedBy	nchar(50)	<input checked="" type="checkbox"/>

Afbeelding 62: Database tabel BadgeUitleningenLogging

Data.cs:

De data klasse bevat een reeks aan functies die data ophalen, updaten, invoegen en verwijderen aan de hand van een SQL connectie.

```
public class Data
{
    private const string ConnectionString = @"

// Badges
6 references
public static DataTable GetBadges(long id = 0, string badgeNummer = null, string badgeNaam = null,
                                    bool? actief = null, bool? inGebruik = null, bool? alleenActieveUitlening = false)...

1 reference
private static string BuildWhereBadge(long id = 0, string badgeNummer = null, string badgeNaam = null, bool? actief = null,
                                       bool? inGebruik = null, string badgeAlias = "b", string uitleningAlias = "bu")...

1 reference
public static DataRow GetBadge(long id)...

1 reference
public static long InsertBadge(string badgeNummer, string badgeNaam, bool? actief)...

1 reference
public static int UpdateBadge(long id, string badgeNummer, string badgeNaam, bool? actief)...

1 reference
public static int DeleteBadge(long id)...

//BadgeUitleningen
2 references
public static DataTable GetBadgeUitleningen(long id = 0, long? badgeID = null, string badgeNaam = null, string uitlenerNaam = null,
                                             string telefoonnummer = null, string firmaNaam = null,
                                             DateTime? startDatum = null, DateTime? eindDatum = null)...

1 reference
private static string BuildWhereBadgeUitleningen(long id = 0, long? badgeID = null, string badgeNaam = null, string uitlenerNaam = null,
                                                 string telefoonnummer = null, string firmaNaam = null,
                                                 DateTime? startDatum = null, DateTime? eindDatum = null)...

1 reference
public static DataRow GetBadgeUitlening(long id)...

2 references
public static DataTable GetActieveBadgeUitleningen(long badgeID)...

1 reference
public static long InsertBadgeUitlening(long badgeID, string uitlenerNaam, string telefoonnummer, string firmaNaam,
                                         DateTime? startDatum, DateTime? eindDatum)...

1 reference
public static int UpdateBadgeUitlening(long id, long badgeID, string uitlenerNaam, string telefoonnummer, string firmaNaam,
                                         DateTime? startDatum, DateTime? eindDatum)...

2 references
public static int UpdateBadgeUitleningEinddatum(long id, DateTime? eindDatum)...

0 references
public static int DeleteBadgeUitlening(long id)...

// Methode om nieuwe ID op te halen
4 references
public static long GetNewID(string tableName)...
```

Afbeelding 63: Data klasse Badge Uitleningen

6.2.3.2. Business-laag

De business-laag bevat de logica van de applicatie. Hier wordt bepaald hoe gegevens uit de data-laag worden geïnterpreteerd en verwerkt, en hoe de applicatie reageert op gebruikersacties.

C# ObjBadge.cs
C# ObjBadgeUitlening.cs

ObjBadge:

```
public class ObjBadge
{
    1 reference
    public ObjBadge()
    {
        ID = null;
    }

    1 reference
    public ObjBadge(long id)
    {
        ID = id;
        Load();
    }

    13 references
    public long? ID { get; set; }
    9 references
    public string BadgeNummer { get; set; }
    8 references
    public string BadgeNaam { get; set; }
    5 references
    public bool Actief { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    1 reference
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    2 references
    public bool IsValid(out string errorText)...

    1 reference
    public bool Load()...

    1 reference
    public bool Save(out string errorText)...

    0 references
    public bool Delete(out string errorText)...
}
```

Afbeelding 64: Object Badge

ObjBadgeUitlening:

```
public class ObjBadgeUitlening
{
    1 reference
    public ObjBadgeUitlening()
    {
        ID = null;
    }

    0 references
    public ObjBadgeUitlening(long id)
    {
        ID = id;
        Load();
    }

    8 references
    public long? ID { get; set; }
    5 references
    public long? BadgeID { get; set; }
    5 references
    public string UitlenerNaam { get; set; }
    6 references
    public string Telefoonnummer { get; set; }
    5 references
    public string FirmaNaam { get; set; }
    5 references
    public DateTime? StartDatum { get; set; }
    4 references
    public DateTime? EindDatum { get; set; }
    1 reference
    public DateTime? InsertedOn { get; set; }
    2 references
    public string InsertedBy { get; set; }
    1 reference
    public DateTime? UpdatedOn { get; set; }
    1 reference
    public string UpdatedBy { get; set; }

    1 reference
    public bool IsValid(out string errorText)...

    1 reference
    public bool Load()...

    1 reference
    public bool Save(out string errorText)...
}
```

Afbeelding 65: Object BadgeUitlening

6.2.3.3. Presentatie-laag

De presentatie-laag is verantwoordelijk voor de interactie met de gebruiker en zorgt ervoor dat de informatie uit de business-laag op een gebruiksvriendelijke manier wordt weergegeven. De presentatie-laag is opgedeeld in 2 delen: het configuratie deel en het gebruikers deel.

CONFIGURATIE DEEL

Hiermee kan de technische dienst badges beheren, nieuwe badges toevoegen en een overzicht van alle uitleningen bekijken. Ook kunnen filters worden toegepast om snel de benodigde informatie te vinden.

Overzicht scherm (huidige situatie):

The screenshot shows a Windows application window titled 'Badges en uitleningen'. At the top left is a 'Filters' section with two radio buttons: 'Huidige situatie' (selected) and 'Geschiedenis'. Below these are dropdown menus for 'In gebruik' and 'Actief', and input fields for 'Badge naam' and 'Firma naam'. To the right are date pickers for 'Startdatum' (dinsdag 4 november 2025) and 'Einddatum' (vrijdag 5 december 2025), along with a search icon. A large table below the filters displays badge information. The table has columns: Badge naam, Uitleners naam, Telefoonnummer, Firma naam, Start datum, and Eind datum. It contains three rows: Badge 1 (Mathijs Ooms, 0479047885, Ziekenhuis Geel, 04/12/2025 12:15), Badge 2 (<Momenteel niet uitgeleend>, <Momenteel niet uitgeleend>, <Momenteel niet uitgeleend>), and Badge 3 (<Momenteel niet uitgeleend>, <Momenteel niet uitgeleend>, <Momenteel niet uitgeleend>). On the right side of the table are two buttons: 'Nieuwe badge' and 'Open badge'.

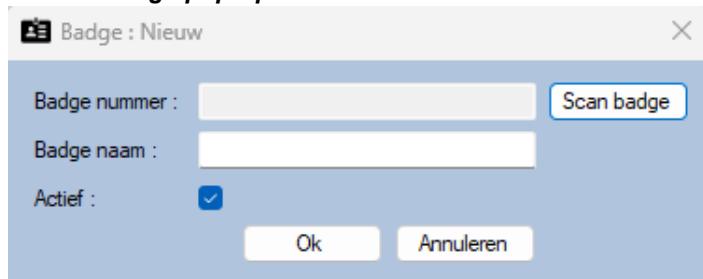
Afbeelding 66: Overzicht scherm (huidige situatie) Badge Uitleningen

Overzicht scherm (geschiedenis):

The screenshot shows the same application window as above, but with the 'Geschiedenis' radio button selected in the 'Filters' section. The table below shows a different set of badge records. The columns are the same: Badge naam, Uitleners naam, Telefoonnummer, Firma naam, Start datum, and Eind datum. The data includes: Badge 1 (Mathijs Ooms, 0479047885, Ziekenhuis Geel, 04/12/2025 12:15), Badge 1 (Ruben thaels, 0491112233, Test firma, 27/11/2025 14:51, 27/11/2025 14:51), Badge 2 (Mo, 764689097, Mo, 27/11/2025 14:44, 27/11/2025 14:45), Badge 2 (test2, 02161561, test2, 26/11/2025 15:07, 27/11/2025 07:51), and Badge 1 (test, 026516165, test, 26/11/2025 15:03, 26/11/2025 15:03). The right side features the same 'Nieuwe badge' and 'Open badge' buttons.

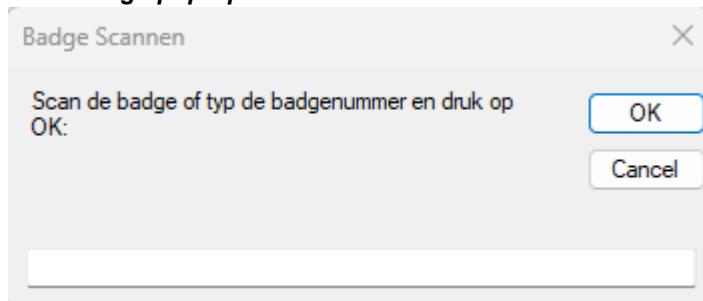
Afbeelding 67: Overzicht scherm (geschiedenis) Badge Uitleningen

Nieuwe badge pop-up:



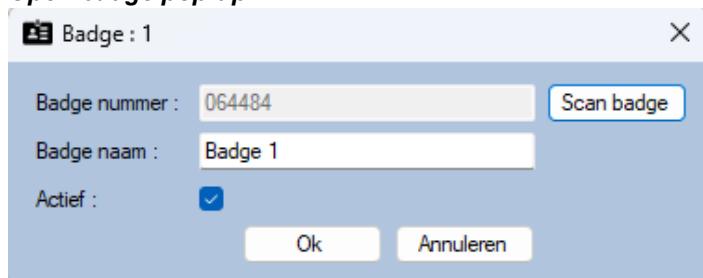
Afbeelding 68: Nieuwe badge pop-up

Scan badge pop-up:



Afbeelding 69: Scan badge pop-up

Open badge pop-up:

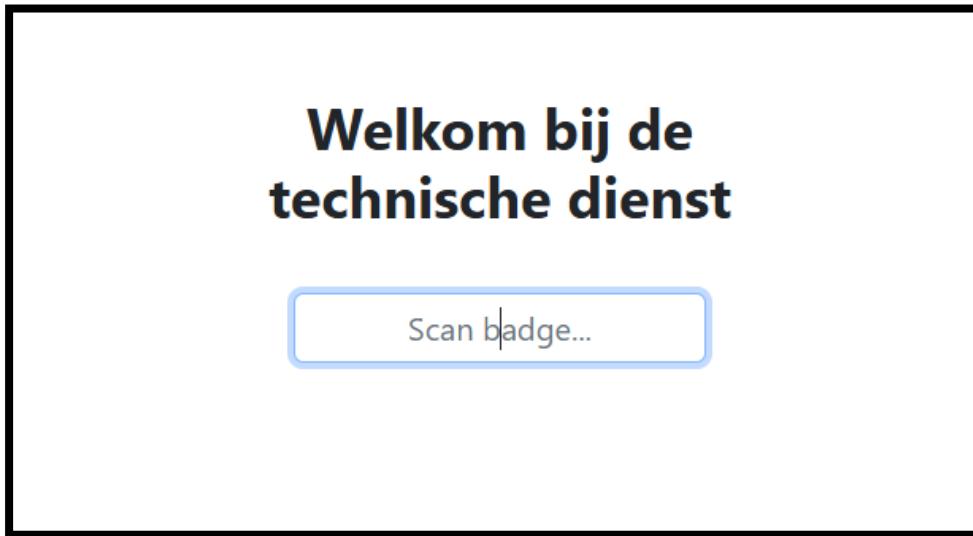


Afbeelding 70: Open badge pop-up

GEBRUIKERS DEEL

Web applicatie die compatibel is voor Zebra-tablets. Na het scannen van een badge toont het scherm automatisch of een badge wordt uitgeleend of teruggebracht, en kan de gebruiker de minimale gegevens invoeren voor registratie.

Startscherm:



Afbeelding 71: Startscherm web app

Bij scannen van badge zonder openstaande uitlening

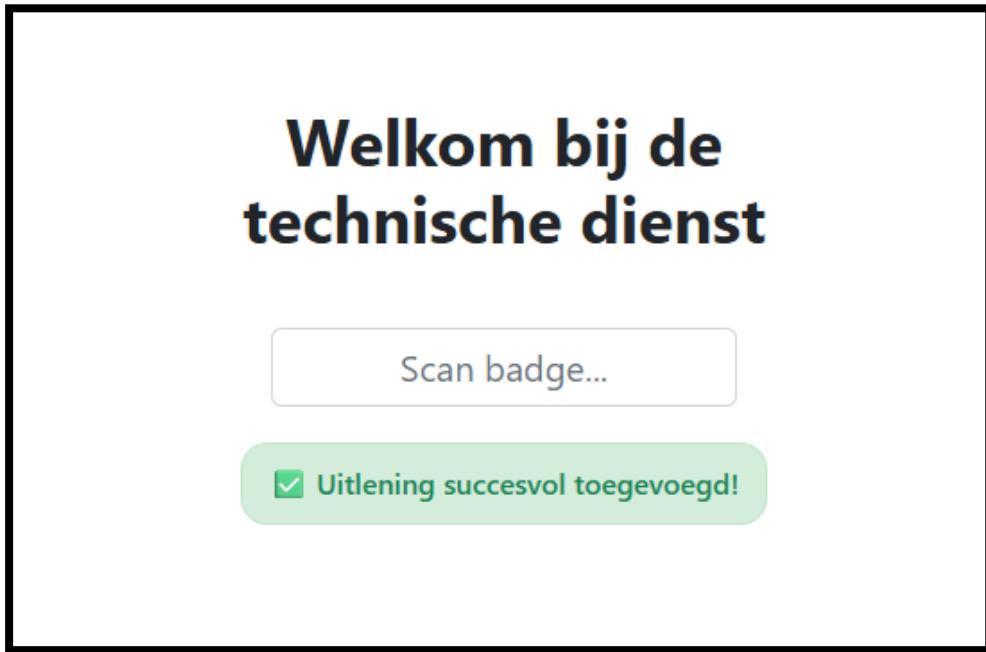
Badge uitlenen scherm:

A screenshot of a tablet screen displaying the badge loaning form. The screen has a white background with a black border. It contains five input fields: "Badge nummer:" with value "064488", "Badge naam:" with value "Badge 2", "Uw naam:" (empty), "Telefoonnummer:" (empty), and "Uw firma:" (empty). At the bottom, there are two buttons: "Bevestig uitlening" on the left and "Annuleren" on the right.

Afbeelding 72: Badge uitlenen scherm

Uitlening succesvol opgeslagen

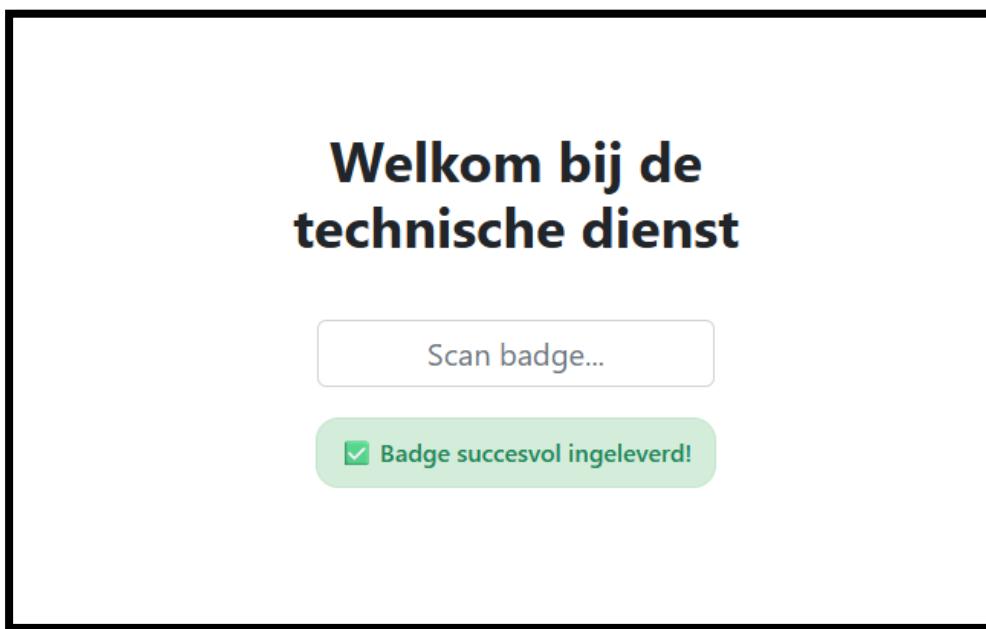
Startscherm met confirmatie van uitlening badge:



Afbeelding 73: Startscherm met confirmatie van uitlening badge

Bij scannen van badge met openstaande uitlening

Startscherm met confirmatie van inlevering badge:



Afbeelding 74: Startscherm met confirmatie van inlevering badge

6.2.4. Uitdagingen

De grootste uitdaging in dit project was het maken van de web applicatie. Dit was de eerste keer dat ik werkte met web forms in combinatie met ASPX. Al bij al ging me dit goed af en is het een mooi eindresultaat.

6.2.5. Eindresultaat

Het resultaat is een stabiel, gebruiksvriendelijk en efficiënt badgesysteem dat handmatige registratie overbodig maakt. De technische dienst heeft een duidelijk overzicht van alle uitleeningen, kan eenvoudig controleren wie verantwoordelijk is voor een badge en kan historisch rapporteren. De combinatie van Windows Forms en een web interface op Zebra-tablets zorgt voor optimale toegankelijkheid en snelheid bij het dagelijks gebruik.

7. Device manager

7.1. Analyse

7.1.1. Klant

De klant is de IT-afdeling van het ziekenhuis, verantwoordelijk voor het beheer, de monitoring en de registratie van netwerkapparaten en hun configuraties.

7.1.2. Huidige situatie

Apparaat gegevens zoals MAC-adressen, internetsnelheid, duplex mode en switchpoort worden momenteel nauwelijks in de database bijgehouden. Informatie wordt grotendeels via Excel-scripts verzameld en opgeslagen in Excel-sheets, die snel verouderen. Hierdoor is het moeilijk om actuele en betrouwbare gegevens op te vragen.

7.1.3. Probleem

De database is incompleet, waardoor het lastig is om een overzicht van alle apparaten en hun netwerkconfiguratie op te vragen. Dit veroorzaakt inefficiëntie bij onderhoud, troubleshooting en planning van netwerkupgrades.

7.1.4. Functionele eisen

Het project zal bestaan uit 2 projecten, één om MAC adressen te updaten uit een Excel bestand en één om switchpoorten up te daten van apparaten gevonden op switches via SSH.

7.1.4.1. Use cases

Use case 1

Naam: Apparaten importeren

Actors: IT-medewerkers

Functionaliteit: Als IT-medewerker kan ik een lijst van apparaten en MAC-adressen vanuit Excel in het systeem importeren.

Voorwaarden: /

Normal flow:

De IT-medewerker selecteert het Excel-bestand. Het systeem leest het bestand in via ExcelDataReader. Gegevens worden gevalideerd en gecontroleerd op duplicaten. Nieuwe apparaten worden toegevoegd aan de database, bestaande apparaten worden geüpdateet.

Uitzondering:

Het Excel bestand kan niet gelezen worden of bevat foute data: Het systeem geeft de melding dat het lezen van het Excel bestand mislukt is.

Use case 2

Naam: Switchpoorten updaten

Actors: IT-medewerkers

Functionaliteit: Als IT-medewerker kan ik via SSH netwerkconfiguratie ophalen en updaten voor elk apparaat.

Voorwaarden: De switch is bereikbaar via het netwerk en SSH is ingeschakeld.

Normal flow:

Het systeem leest het Excel bestand met switches en stuurt SSH-commando's voor elke switch. Het systeem interpreteert de output en haalt gegevens op zoals duplex, snelheid, switch en interface. De database wordt bijgewerkt met de nieuwe configuratiegegevens.

Uitzondering:

Het Excel bestand kan niet gelezen worden of bevat foute data: Het systeem geeft de melding dat het lezen van het Excel bestand mislukt is.

Op een switchpoort zitten 5 of meer apparaten: Het systeem slaat deze switchpoort over en update de data van deze apparaten niet.

7.1.5. Technische eisen

Technische eisen beschrijven hoe de applicatie moet functioneren op technisch vlak, inclusief systemen, databases, architectuur en beperkingen voor betrouwbaarheid, performance en onderhoudbaarheid.

7.1.5.1. Data-laag

De data moet gezocht en opgeslagen worden in de bestaande Device Manager database. De nodige tabellen voor de MAC adres updaten applicatie zijn PC, MedischApparaat, Server en Telefoon. Enkel Telefoon zal gebruikt worden voor inserts en updates, de rest enkel updates. Voor de switchpoort updaten applicatie zijn dit AccesPoint, MedischApparaat, OtherDevice, PC, Printer, Server en Telefoon om te updaten en tabellen Switch en Switchpoort voor validatie.

7.1.5.2. Business-laag

In de business-laag voor de MAC adres updaten zal enkel het object Telefoon gebruikt worden om de telefoondata te valideren om daarna te inserten in de database via de data-laag.

Voor de switchpoort updaten zal het object Switch gemaakt worden om in de presentatie-laag een tijdelijk object Switch aan te maken om zo één voor één de nodige commando's uit te voeren.

7.1.5.3. Presentatie-laag

Voor de MAC adres updaten zullen er 2 delen komen voor de presentatie-laag, een gebruiksvriendelijke Windows Forms applicatie en een console applicatie die kan werken als script. In de Windows Forms app is het nodig om een knop te hebben om een Excel bestand te kiezen waarvan de data dan in een tabel komt en een knop om met deze data de database te updaten. Voor de console app zal alles hetzelfde werken, enkel zal je moeten kiezen tussen de Telefoon Excel en Andere apparaten Excel omdat dit pad hard gecodeerd in de applicatie komt.

Voor de Switchpoort updaters komt er enkel een console app met hard gecodeerd Excel pad. In deze Excel zit een lijst van switches die gebruikt zal worden door de applicatie om de SSH commando's op uit te voeren. Er komt dan per switch een tabel met resultaten per switchpoort.

7.1.6. Technologieën

Het systeem zal geschreven worden in C#, met een SQL Server-database voor centrale opslag. Excel-bestanden worden gelezen met ExcelDataReader. SSH-commando's worden uitgevoerd via de SSH.NET bibliotheek. De WinForms-interface biedt een gebruiksvriendelijke bediening voor IT-medewerkers.

7.2. Realisatie

7.2.1. Doel

Het doel van de realisatie is het ontwikkelen van een betrouwbaar en efficiënt systeem dat automatisch apparaten en netwerkconfiguraties kan importeren, controleren en updaten. Het systeem vervangt handmatige Excel-verwerking en biedt een actueel overzicht van alle apparaten en hun netwerkstatus.

7.2.2. Functionaliteiten

De belangrijkste functionaliteiten in dit project zijn:

- **Apparaten importeren:** Lees Excel-bestanden in en update de database automatisch.
- **Switchpoorten updaten:** Voer SSH-commando's uit om configuratiegegevens te verzamelen en bij te werken.
- **Overzicht en beheer:** IT-medewerkers kunnen apparaten en hun configuraties bekijken en aanpassen.
- **Mails versturen:** alle MAC adressen en switchpoorten die niet worden gevonden worden in een lijst gezet en deze lijst wordt verstuurd via mail.

7.2.3. 3-tier architectuur

Het project maakt gebruik van een 3-tier architectuur, met een scheiding tussen data-, business- en presentatie-laag. Dit zorgt voor onderhoudbaarheid, schaalbaarheid en duidelijke verantwoordelijkheden. De 2 MAC adres updaters maken gebruik van dezelfde data- en business-laag.

7.2.3.1. Data-laag

MAC adres updater – Data.cs:

```
public class Data
{
    public static string ConnectionString = @"[REDACTED]";

    private readonly List<string> tabelLijst = new List<string>
    {
        "Pc",
        "MedischApparaat",
        "Server",
        "Telefoon"
    };

    4 references
    public int UpdateMacAdres(string deviceNaam, string macAdres)[...]

    3 references
    public static long InsertTelefoon(
        string deviceNaam,
        string plaats = "",
        int? modelId = 1,
        string serieNummer = "",
        string macAdres = "",
        string opmerkingen = ""
    )[...]

    1 reference
    public static long GetNewID(string tableName)[...]
}
```

Afbeelding 75: Data klasse MAC adres updater

Switchpoort updater – Data.cs:

```
public class Data
{
    public static string ConnectionString = @"[REDACTED]";

    private readonly List<string> tabelLijst = new List<string>
    {
        "AccessPoint",
        "MedischApparaat",
        "OtherDevice",
        "Pc",
        "Printer",
        "Server",
        "Telefoon"
    };

    1 reference
    public int UpdateDevice(string macAdres, int snelheid, string duplex, string switchPoort)[...]

    1 reference
    public bool IsCorrectSwitchPort(string netwerkSwitchPoort)[...]
}
```

Afbeelding 76: Data klasse Switchpoort updater

7.2.3.2. Business-laag

ObjTelefoon:

In de MAC adres updaer wordt gebruik gemaakt van object Telefoon om deze data te valideren.

```
public class Telefoon
{
    3 references
    public int? ID { get; set; }
    2 references
    public string Naam { get; set; }
    2 references
    public string MAC_adres { get; set; }
    2 references
    public string Plaats { get; set; }
    2 references
    public int? ModelID { get; set; }
    1 reference
    public string SerieNummer { get; set; }
    1 reference
    public string Opmerkingen { get; set; }
    4 references
    public bool IsNew { get; set; }

    0 references
    public Telefoon(int? id)
    {
        ID = id;
        IsNew = id == null;
    }

    0 references
    public Telefoon()
    {
        ID = null;
        IsNew = true;
    }

    1 reference
    public string Error{...}

    1 reference
    private string GetErrorForProp(string columnName){...}

    1 reference
    public bool IsValid(){...}

    0 references
    public bool Save(){...}
}
```

Afbeelding 77: Object Telefoon

ObjSwitch:

In de switchpoort updaer is er enkel object Switch dat enkel gebruikt wordt om tijdelijke switch data bij te houden.

```
public class ObjSwitch
{
    7 references
    public string Naam { get; set; }
    3 references
    public string Ip { get; set; }
    2 references
    public string Username { get; set; }
    2 references
    public string Password { get; set; }
}
```

Afbeelding 78: Object Switch

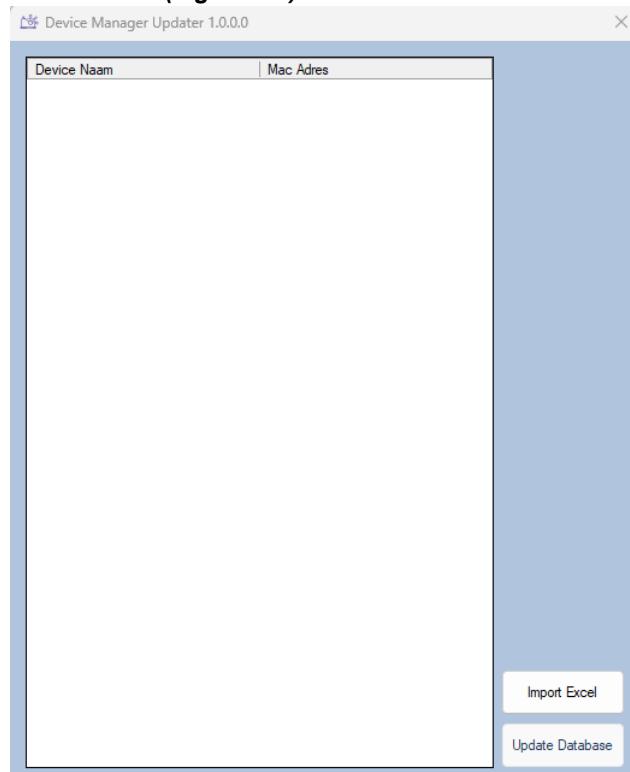
7.2.3.3. Presentatie-laag

MAC ADRES UPDATER

Windows Forms:

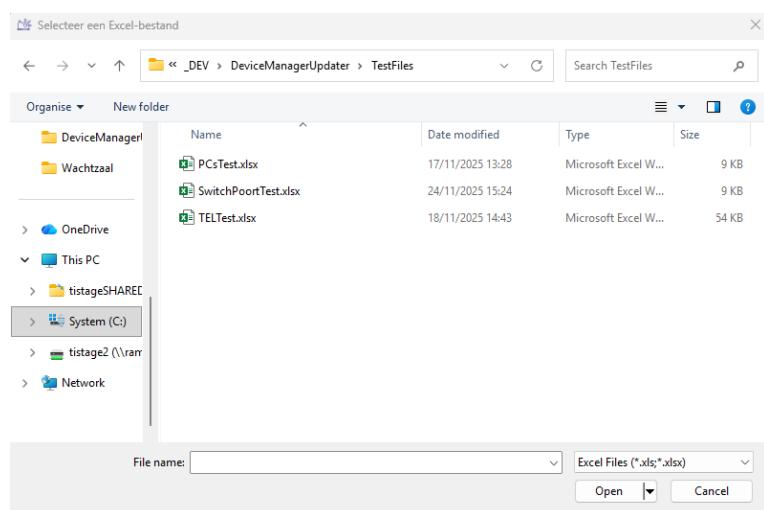
Dit is de gebruiksvriendelijke applicatie bedoeld om manueel een Excel bestand te kiezen om de database te updaten.

Startscherm (lege tabel):



Afbeelding 79: Startscherm (lege tabel) MAC adres updater

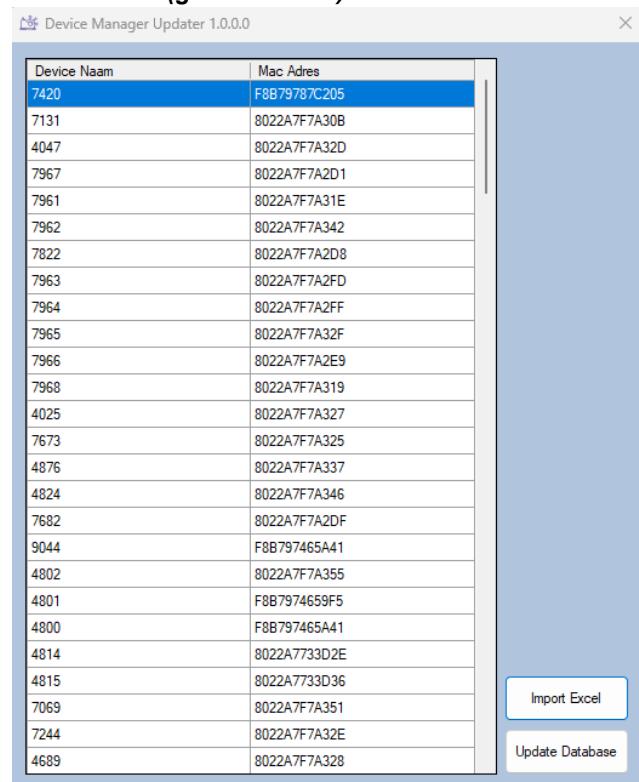
Bij klikken op knop “Import Excel”



Afbeelding 80: Excel importeren scherm

Na kiezen Excel bestand

Startscherm (gevulde tabel):



Device Naam	Mac Adres
7420	F8B79787C205
7131	8022A7F7A30B
4047	8022A7F7A32D
7967	8022A7F7A2D1
7961	8022A7F7A31E
7962	8022A7F7A342
7822	8022A7F7A2D8
7963	8022A7F7A2FD
7964	8022A7F7A2FF
7965	8022A7F7A32F
7966	8022A7F7A2E9
7968	8022A7F7A319
4025	8022A7F7A327
7673	8022A7F7A325
4876	8022A7F7A337
4824	8022A7F7A346
7682	8022A7F7A2DF
9044	F8B797465A41
4802	8022A7F7A355
4801	F8B7974659F5
4800	F8B797465A41
4814	8022A7733D2E
4815	8022A7733D36
7069	8022A7F7A351
7244	8022A7F7A32E
4689	8022A7F7A328

Afbeelding 81: Startscherm (gevulde tabel) MAC adres updater

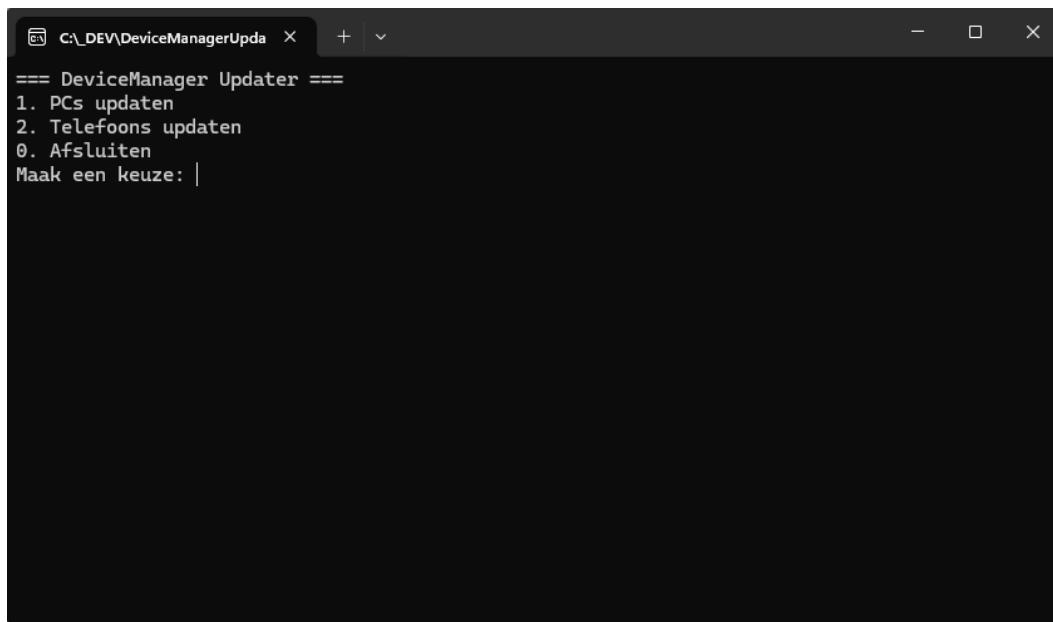
Bij klikken knop "Update Database"



Afbeelding 82: Bevestiging pop-up MAC adres updater

Console applicatie:

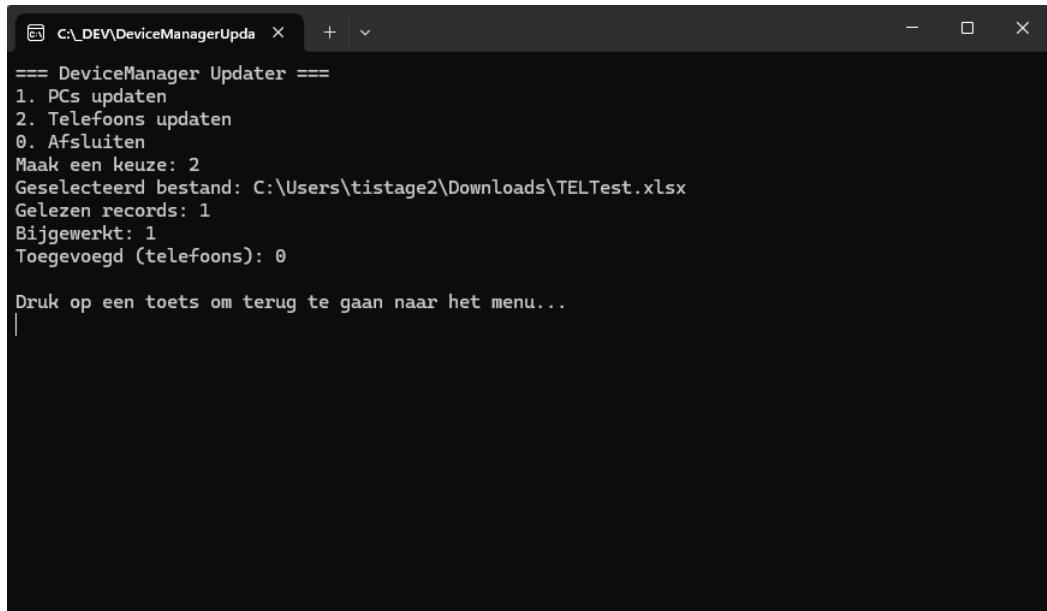
Deze console applicatie is bedoeld om te gebruiken als script dat wekelijks of maandelijks automatisch uit te voeren.



```
== DeviceManager Updater ==
1. PCs updaten
2. Telefoons updaten
0. Afsluiten
Maak een keuze: |
```

Afbeelding 83: Startschermscherm console applicatie MAC adres updater

Na het maken van een keuze (1 of 2)



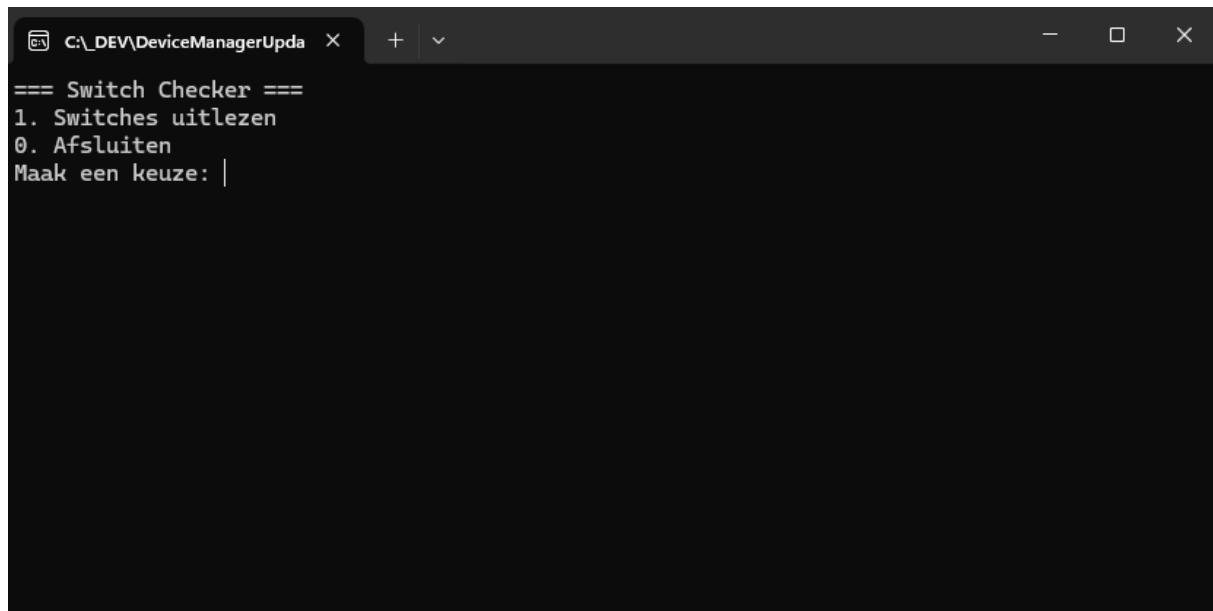
```
== DeviceManager Updater ==
1. PCs updaten
2. Telefoons updaten
0. Afsluiten
Maak een keuze: 2
Geselecteerd bestand: C:\Users\tistage2\Downloads\TELTest.xlsx
Gelezen records: 1
Bijgewerkt: 1
Toegevoegd (telefoons): 0

Druk op een toets om terug te gaan naar het menu...
|
```

Afbeelding 84: Update voltooid scherm

SWITCHPOORT UPDATER

Deze console applicatie is ook bedoeld om te gebruiken als script dat wekelijks of maandelijks herhaalt wordt om automatisch de database te updaten.



Afbeelding 85: Startscherf Switchpoort updater

Na het maken van een keuze (1)

A screenshot of the same application window after a choice was made. It now displays a table of network port configurations:

Ch	Slot	Port	MAC 1	MAC 2	MAC 3	MAC 4	MAC 5	Speed	Duplex		
1	1	1	0000E000102		0002E70AB5F7		0002E70AB61B		0002E70AB64A	0002E70AF486	1000 Full
1	1	2	2CFAA2F30BA8		B0227A3107DF		B0227A317551		B0227A3189E4	F4390918DFFD	1000 Full
1	1	3	0068EBB7E90E					1000	Full		
MAC 0068EBB7E90E geüpdateet (1000/Full)											
1	1	4	00206B414A0D					100	Full		
1	1	6	B0227A31763B					1000	Full		
MAC B0227A31763B geüpdateet (1000/Full)											
1	1	8	8022A7F7A338		9C7BEFC06164		8022A7F7A338			100	Full
MAC 8022A7F7A338 geüpdateet (100/Full)											
MAC 9C7BEFC06164 geüpdateet (100/Full)											
MAC 8022A7F7A338 geüpdateet (100/Full)											
1	1	11	E8D8D1CFB2C1					1000	Full		
MAC E8D8D1CFB2C1 geüpdateet (1000/Full)											
1	1	15	5410ECCB6770					100	Full		
1	1	16	5410ECCB69D1					100	Full		
1	1	18	644ED76C059C					1000	Full		
MAC 644ED76C059C geüpdateet (1000/Full)											
1	1	20	E8D8D1CEFF77					1000	Full		
MAC E8D8D1CEFF77 geüpdateet (1000/Full)											
1	1	22	644ED76C04F1					1000	Full		
MAC 644ED76C04F1 geüpdateet (1000/Full)											
1	1	25	B0227A317719					1000	Full		
MAC B0227A317719 geüpdateet (1000/Full)											

Afbeelding 86: Update voltooid scherm

Voorbeeld van een mail die verstuurd wordt als er MAC adressen zijn die niet gevonden kunnen worden:

NoReplyDeviceManager@azstdimpna.be
To: Stagiair Informatica 2

De volgende MAC-adressen werden niet gevonden:
E1ACCE-1/1/4: 00206B414A0D
E1ACCE-1/1/15: 5410ECCB6770
E1ACCE-1/1/16: 5410ECCB69D1
E1ACCE-1/1/28: 3C2AF4226F0F
E1ACCE-1/1/30: 000732A8361B
E1ACCE-1/1/31: B8A44FAEE939
E1ACCE-1/1/32: 000732A835E9
E1ACCE-1/1/40: 689CE2574826
E1ACCE-1/1/41: 000E3D208303
E1ACCE-1/1/42: 000732A835DB
E1ACCE-1/1/48: 00E04B1D60FB
E1ACCE-2/1/3: 30055CFD02A5
E1ACCE-2/1/4: 9CAED3B4A6E4
E1ACCE-2/1/14: 5410ECCB69D7
E1ACCE-2/1/15: FC0FE76DFCE8
E1ACCE-2/1/16: 5410ECCB6734
E1ACCE-2/1/22: DCCD2FA36547
E1ACCE-2/1/25: 00187DE32154

Afbeelding 87: Mail voorbeeld

7.2.4. Uitdagingen

De grootste uitdaging was de maken van de SSH verbinding en het verkrijgen van de juiste gegevens door de juiste commando's uit te voeren. Ook om met deze data dan de database te updaten was een moeilijkheid. Uiteindelijk zijn alle eisen tegemoetgekomen met een werkend eindresultaat.

7.2.5. Eindresultaat

Het resultaat is een stabiel en gebruiksvriendelijk Device Manager-systeem. IT-medewerkers kunnen apparaten efficiënt beheren, netwerkconfiguraties automatisch bijwerken en foutieve of ontbrekende gegevens snel opsporen. Het systeem vervangt handmatige Excel-verwerking en biedt een betrouwbaar overzicht van alle apparaten en hun netwerkstatus.

8. Besluit

8.1. Samenvatting

Tijdens mijn stageperiode in het ziekenhuis heb ik aan een breed scala van projecten gewerkt die elk gericht waren op het optimaliseren van interne processen. Deze projecten varieerden van applicaties voor patiëntaanmelding en logistieke etiketten tot systemen voor uitleningen, netwerkbeheer en badgebeheer. Elk project kende een eigen analyse-, ontwikkelings- en testtraject, steeds opgebouwd volgens een gestructureerde en onderbouwde aanpak met duidelijke scheiding tussen data-, business- en presentatielagen. Doorheen de realisatie van de verschillende applicaties heb ik gebruikgemaakt van technologieën zoals C#, Windows Forms, ASP.NET, SQL Server, ExcelDataReader, barcodescanners, eID-lezers en SSH.NET.

Alle oplossingen die ik ontwikkelde hebben als doel om medewerkers efficiënter, sneller en foutloos te laten werken. Of het nu gaat om het automatiseren van patiëntflows, het genereren van etiketten, het registreren van uitleningen of het actualiseren van netwerkdata: elk project draagt tastbaar bij aan de digitale werking van het ziekenhuis.

8.2. Conclusies

Uit deze stage en het realiseren van de verschillende projecten kan ik enkele belangrijke conclusies trekken:

- **Digitalisering levert directe meerwaarde op:** Bij elk project werd duidelijk dat manuele processen foutgevoelig zijn en veel tijd vragen. De ontwikkelde applicaties zorgen voor tijdsinst, minder fouten en een betere workflow.
- **Een goede analyse is essentieel:** Veel projecten vereisten diepgaand begrip van bestaande processen, databanken en uitzonderingssituaties. Een correcte analyse zorgde telkens voor een vlottere ontwikkeling.
- **3-tier architectuur is zeer nuttig:** Door een duidelijke scheiding tussen databanktoegang, logica en interface bleven de projecten overzichtelijk, onderhoudbaar en uitbreidbaar.
- **Flexibiliteit is noodzakelijk in een ziekenhuisomgeving:** Elk project had te maken met bestaande software, hardware of werkwijzen. Aanpassingsvermogen en pragmatische keuzes waren cruciaal.
- **Mijn technische vaardigheden zijn sterk gegroeid:** Door met nieuwe technologieën zoals eID-lezen, barcode-integraties en SSH-automatisatie te werken, heb ik mijn technische kennis aanzienlijk verbreed en verdiept.

8.3. Evaluatie

Ik kijk terug op een bijzonder leerrijke en uitdagende stageperiode. Elk project bood nieuwe inzichten en liet mij kennismaken met realistische problemen en professionele ontwikkelstandaarden. Vooral de diversiteit van de projecten zorgde ervoor dat ik mijn kennis in verschillende domeinen tegelijk kon uitbreiden.

Daarnaast heb ik geleerd om zelfstandig problemen te analyseren, oplossingen te bedenken en applicaties te bouwen die effectief gebruikt worden in de dagelijkse werking van het ziekenhuis. De feedbackmomenten met medewerkers waren zeer waardevol en hielpen mij om de gebruikerservaring continu te verbeteren.

Tot slot ben ik trots op de gerealiseerde toepassingen en de impact die ze hebben op de interne werking. Deze stage vormde een uitstekende voorbereiding op mijn verdere loopbaan als programmeur en gaf mij het vertrouwen om complexere projecten zelfstandig aan te pakken.

9. Verwijzingen

Websites

- PlantUML. (2025). PlantUML Editor Online: <https://www.planttext.com/>
- Microsoft. (2025). SQL Server 2022: <https://www.microsoft.com/en-us/sql-server/sql-server-2022>
- Swelio Library. (2024, november 14). Github: <https://github.com/perevoznyk/swelio-sdk>
- NuGet. (2021, januari 24). SSH.NET: <https://www.nuget.org/packages/SSH.NET>
- NuGet. (2025). ExcelDataReader: <https://www.nuget.org/packages/ExcelDataReader>

AI-assistenten

- OpenAI. (2025). ChatGPT: <https://chatgpt.com>