

# Grenzen van de low-code tool Mendix.

## Optionele ondertitel.

---

**Senne Timbreur.**

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

**Promotor:** Mevr. F.Spriet

**Co-promotor:** Mevr. J.Alexander

**Academiejaar:** 2024–2025

**Tweede examenperiode**

**Departement IT en Digitale Innovatie .**

**HO  
GENT**



# Woord vooraf

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

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

# Samenvatting

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

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

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

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

# Inhoudsopgave

<b>Lijst van figuren</b>	<b>vii</b>
<b>Lijst van tabellen</b>	<b>viii</b>
<b>Lijst van codefragmenten</b>	<b>ix</b>
<b>1 Inleiding</b>	<b>1</b>
1.1 Probleemstelling	1
1.2 Onderzoeksvraag	2
1.2.1 Probleemdomein	2
1.2.2 Oplossingsdomein	2
1.3 Opzet van deze bachelorproef	3
<b>2 Stand van zaken</b>	<b>4</b>
2.1 Introductie	4
2.2 Model-Driven Development	5
2.2.1 Kernconcepten van Model-Driven Development	5
2.2.2 Historische context en evolutie	5
2.2.3 Voordelen van Model-Driven Development	6
2.2.4 Belangrijkste functionaliteitsgebieden voor MDD-tools	6
2.2.5 De evolutie naar low-code platformen	8
2.3 Low-code platformen	8
2.3.1 OutSystems	9
2.3.2 Joget DX	9
2.3.3 Mendix	10
2.3.4 Keuze voor Mendix als low-code platform	10
2.4 Mendix en high-code	10
2.4.1 Verschillende benaderingen	10
2.4.2 Enterprise Flexibiliteit door Model-Based Engineering (MBE)	11
2.4.3 Low-Code als Brug tussen Business en IT	11
2.4.4 Kunnen beiden elkaar aanvullen?	11
2.5 Keuze tussen high-code en low-code?	12
<b>3 Methodologie</b>	<b>13</b>
3.1 Methodologie	13
3.1.1 Fase 1: Analyse van het probleemdomein	13
3.1.2 Fase 2: Onderzoek naar het oplossingsdomein	14

<b>4 Conclusie</b>	<b>16</b>
<b>A Onderzoeksvoorstel</b>	<b>18</b>
A.1 Inleiding . . . . .	19
A.2 Literatuurstudie . . . . .	21
A.2.1 Low-code en Mendix . . . . .	21
A.2.2 Low-code use cases . . . . .	21
A.2.3 Beperkingen van low-code . . . . .	22
A.2.4 Low-code versus high-code . . . . .	23
A.3 Methodologie . . . . .	23
A.3.1 Fase 1: Analyse van het probleemdomein . . . . .	24
A.3.2 Fase 2: Onderzoek naar het oplossingsdomein . . . . .	25
A.4 Verwacht resultaat, conclusie . . . . .	25
<b>Bibliografie</b>	<b>27</b>

# Lijst van figuren

2.1	Evolution . . . . .	5
2.2	Evolution . . . . .	9

# Lijst van tabellen



# Lijst van codefragmenten

# 1

## Inleiding

Apvine, een toonaangevend IT-consultancybedrijf, maakt intensief gebruik van low-code platformen zoals Mendix om zakelijke applicaties snel en efficiënt te ontwikkelen. Deze aanpak verlaagt de ontwikkeltijd en biedt een flexibele oplossing voor veel projecten. Echter, wanneer applicaties complexe bedrijfslogica, real-time gegevensverwerking of integraties met legacy-systemen vereisen, kunnen de beperkingen van low-code ontwikkeling zichtbaar worden.

In zulke gevallen kan een hybride of high-code aanpak nodig zijn, maar zonder een duidelijk beslissingskader is het lastig om te bepalen wanneer deze overstap gemaakt moet worden. Dit gebrek aan richtlijnen kan leiden tot projectvertragingen, hogere kosten en onverwachte aanpassingen, wat een negatieve impact heeft op zowel Apvine als haar klanten.

Dit onderzoek richt zich op het vaststellen van de grenzen van Mendix en het ontwikkelen van een gestructureerd beslissingskader, zodat projectmanagers en softwarearchitecten beter onderbouwde keuzes kunnen maken tussen low-code en high-code ontwikkeling.

### 1.1. Probleemstelling

De groeiende populariteit van low-code platformen zoals Mendix biedt bedrijven de mogelijkheid om sneller en efficiënter zakelijke applicaties te ontwikkelen. Deze aanpak verlaagt de technische drempel en versnelt het ontwikkelproces. Toch stuiten bedrijven zoals Apvine op beperkingen van deze technologie wanneer ze complexe bedrijfslogica, real-time gegevensverwerking en integraties met legacy-systemen moeten implementeren.

Er ontbreekt momenteel een duidelijk beslissingskader binnen Apvine om te bepalen wanneer low-code een geschikte oplossing is en wanneer high-code ontwikkeling noodzakelijk wordt. Dit leidt tot projectvertragingen, hogere kosten en mo-

gelijke klanttevredenheid door onverwachte aanpassingen in de ontwikkelingsstrategie.

Dit onderzoek richt zich op het identificeren van de grenzen van Mendix en het formuleren van een gestructureerd beslissingskader. Hiermee kunnen projectmanagers en softwarearchitecten beter onderbouwde keuzes maken over de inzet van low-code en high-code technologieën in verschillende scenario's.

## 1.2. Onderzoeksvraag

Dit onderzoek richt zich op de onderzoeksvraag: "Wat zijn de beperkingen van de low-code-tool Mendix bij het ontwikkelen van zakelijke applicaties en welke criteria bepalen wanneer een overstap naar high-code ontwikkeling noodzakelijk wordt?" Om deze centrale vraag te beantwoorden, worden de volgende deelvragen onderzocht:

### 1.2.1. Probleemdomein

- Welke gevolgen heeft het ontbreken van een beslissingskader voor de keuze tussen low-code en high-code binnen Apvine?
- Welke problemen ontstaan er in projecten door het gebrek aan richtlijnen voor de keuze tussen ontwikkelmethoden?
- Wat zijn de huidige criteria die Apvine gebruikt bij het kiezen tussen low-code en high-code ontwikkeling?
- Welke knelpunten ervaren projectmanagers en architecten bij het maken van deze keuze?

### 1.2.2. Oplossingsdomein

- Wat zijn de technische beperkingen van Mendix bij het omgaan met complexe bedrijfslogica, real-time gegevensverwerking en integraties met legacy-systemen?
- In welke specifieke scenario's binnen Mendix-projecten kan een hybride of high-code oplossing nodig zijn, en welke factoren bepalen deze overstap?
- Hoe kunnen projectomvang, tijdslijnen en klantvereisten de beslissing beïnvloeden om wel of niet Mendix (low-code) te gebruiken of over te schakelen naar high-code?
- Welke lessen kunnen worden getrokken uit eerdere projecten bij Apvine waarin Mendix werd ingezet, en hoe kunnen deze inzichten bijdragen aan een effectief beslissingskader?

Door deze vragen te beantwoorden, beoogt dit onderzoek een onderbouwd beslissingskader te ontwikkelen dat projectmanagers en softwarearchitecten ondersteunt bij het maken van een weloverwogen keuze tussen low-code en high-code ontwikkelmethoden.

### **1.3. Opzet van deze bachelorproef**

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

# 2

## Stand van zaken

### 2.1. Introductie

Mendix vertegenwoordigt een van de toonaangevende low-code ontwikkelplatformen in het huidige softwareontwikkelingslandschap (Hermans & Mileff, [2023](#)). Als implementatie van Model-Driven Development (MDD) principes stelt Mendix zowel professionele ontwikkelaars als zakelijke gebruikers in staat om applicaties te creëren via visuele modellering in plaats van traditionele codering. Het platform heeft aanzienlijke tractie gewonnen onder ondernemingen die hun digitale transformatie-initiatieven willen versnellen door ontwikkeltijd en technische complexiteit te verminderen (van Oosten, [2020](#)).

Hoewel Mendix via zijn modelgestuurde aanpak tal van voordelen biedt, waaronder verhoogde ontwikkelingssnelheid en bredere participatie van niet-technische belanghebbenden, presenteert het ook bepaalde beperkingen die de effectiviteit in verschillende contexten beïnvloeden (Yerukala, [2022](#)). Deze bachelorproef onderzoekt Mendix als een hedendaagse manifestatie van Model-Driven Development en analyseert kritisch de beperkingen ervan op het gebied van technische mogelijkheden, uitdagingen bij organisatorische adoptie en comparatieve nadelen ten opzichte van vergelijkbare platformen.

Door zowel de theoretische onderbouwing van MDD als de praktische implementatie ervan in Mendix te begrijpen, beoogt dit onderzoek een uitgebreide beoordeling te geven van waar en hoe Mendix mogelijk tekortschiet in het aanpakken van complexe enterprise software behoeften, ondanks de innovatieve benadering van applicatieontwikkeling.

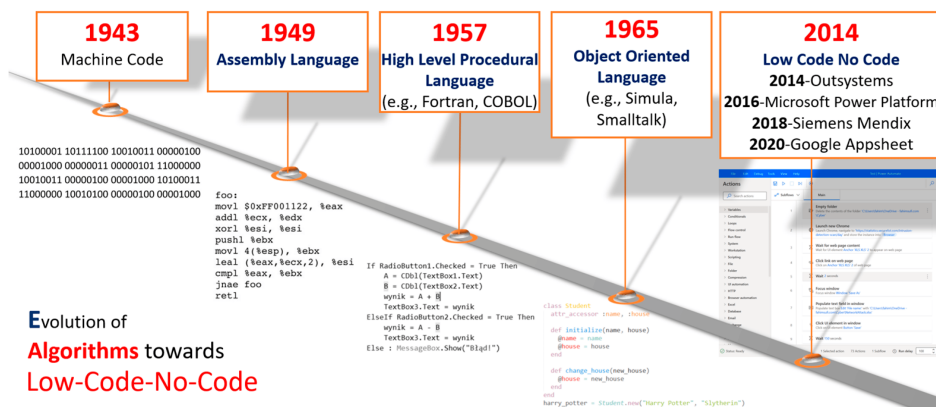
## 2.2. Model-Driven Development

In dit hoofdstuk wordt de theoretische achtergrond van Model-Driven Development (MDD) en low-code ontwikkeling uiteengezet, met een specifieke focus op Mendix. Dit sluit aan bij de introductie van de stand van zaken, waarin het belang van MDD en low-code ontwikkeling werd geïntroduceerd. Dit hoofdstuk biedt een diepgaande analyse van MDD, inclusief de voordelen en beperkingen, en gaat na hoe Mendix binnen dit kader past.

### 2.2.1. Kernconcepten van Model-Driven Development

Volgens het onderzoek Hailpern en Tarr (2006) is MDD “een software engineering aanpak die bestaat uit de toepassing van modellen om het abstractieniveau te verhogen” in softwareontwikkeling. Deze aanpak ontstond als een natuurlijke evolutie in de overgang van low-level programmeertalen zoals assembly naar higher-level talen zoals Java en C#, waarbij modellen de volgende stap in deze abstractie-evolutie vertegenwoordigen.

Het fundamentele principe achter MDD is dat het werken op hogere abstractieniveaus ontwikkelaars in staat stelt om complexe systemen effectiever en met minder inspanning te beheren. Dit komt overeen met de historische evolutie van programmeertalen, waarbij elke nieuwe generatie de abstractie vergrootte om de productiviteit te verbeteren en de cognitieve belasting van ontwikkelaars te vermindere-



**Figuur 2.1:** Evolutie van programmeertalen naar steeds hogere abstractieniveau (Sufi, 2023).

### 2.2.2. Historische context en evolutie

Model Driven Development heeft een rijke geschiedenis die meer dan 20 jaar teruggaat (Henkel & Stirna, 2010). De evolutie verliep in verschillende fasen:

In de jaren '80 ontstonden de eerste geavanceerde modelleringsmethoden om de vereisten van informatiesystemen vast te leggen. Dit vormde de conceptuele basis voor MDD (Henkel & Stirna, 2010).

De jaren '90 brachten propriëtaire CASE-tools (Computer Aided Software Enginee-

ring), die informatiesystemen gedeeltelijk konden genereren op basis van modellen. Deze tools hadden echter beperkingen - vaak konden ze alleen codetemplates genereren waarna programmeurs de rest handmatig moesten aanvullen (Case, 1985).

Modernere benaderingen omvatten OMG's Model Driven Architecture (MDA) en executable UML (xUML). MDA werkt met verschillende modelleringsniveaus en transformaties tussen deze niveaus naar code, waarbij de gegenereerde code verder kan worden uitgewerkt (Soley & the OMG Staff Strategy Group, 2000). xUML daarentegen streeft naar volledige codegeneratie zonder handmatige aanvullingen.

Er bestaan grote verschillen tussen oudere CASE-tools zoals Oracle Forms en Microsoft Access Forms en nieuwere tools gebaseerd op open standaarden zoals OptimaIJ. Hoewel beide soorten tools MDD ondersteunen door abstractieniveaus te verhogen, bieden tools zoals OptimaIJ meer controle over modeltransformaties en codegeneratie, terwijl formulier-gebaseerde tools zoals Oracle Forms minder controle bieden (Henkel & Stirna, 2010).

Ondanks deze verschillen delen alle MDD-benaderingen één fundamenteel doel: het verhogen van het abstractieniveau in softwareontwikkeling, waardoor ontwikkelaars complexere systemen kunnen bouwen met minder inspanning - vergelijkbaar met de evolutie van assembly-taal naar hogere programmeertalen zoals C# en Java.

### 2.2.3. Voordelen van Model-Driven Development

De literatuur identificeert verschillende belangrijke voordelen van het gebruik van MDD-benaderingen (Soley & the OMG Staff Strategy Group, 2000). Door te werken met goed gedefinieerde modellen kunnen ontwikkelaars het aantal implementatiefouten verminderen, waardoor de softwarekwaliteit verbetert. Hogere abstractieniveaus stellen ontwikkelaars in staat om zich te richten op bedrijfslogica in plaats van technische implementatiedetails, wat de productiviteit van ontwikkelaars verhoogt. Modellen geven een duidelijker inzicht in de systeemstructuur en het gedrag, waardoor het ontwikkelproces beter onder controle is.

Bovendien vermindert het automatisch genereren van code de handmatige coördineringsinspanning en de bijbehorende fouten, wat leidt tot lagere ontwikkel- en onderhoudskosten. Snellere ontwikkelcycli helpen organisaties efficiënter om te gaan met hun applicatieontwikkelingsbehoeften, waardoor de ontwikkelingsachterstand afneemt. De mogelijkheid om snel iteraties uit te voeren en werkende functionaliteit te demonstreren verbetert de betrokkenheid van belanghebbenden, waardoor de klanttevredenheid toeneemt.

### 2.2.4. Belangrijkste functionaliteitsgebieden voor MDD-tools

Om een MDD-tool effectief te laten zijn, moet het verschillende kritieke functionaliteiten ondersteunen, die kunnen worden gecategoriseerd in ondersteuning van

zowel het modelleren als het ontwikkelproces.

Effectieve ondersteuning bij het modelleren vereist dat tools de juiste abstractieniveaus hanteren, waarbij irrelevante details worden verborgen terwijl essentiële concepten worden blootgelegd.

Dit is vooral belangrijk omdat belanghebbenden modellen voor verschillende doeleinden gebruiken. Modellen moeten begrijpelijk zijn voor zowel technische als niet-technische belanghebbenden, idealiter met behulp van intuïtieve, voorspelbare notatie - veel tools maken om deze reden gebruik van UML, omdat dit wordt beschouwd als de de-facto standaard in de industrie (Marín e.a., 2015). Modellen moeten uitvoerbaar zijn, zelfs als ze incompleet zijn, zodat incrementele ontwikkeling mogelijk is en ontwikkelaars het gedrag van het systeem kunnen voorspellen door middel van experimenten of formele analyse.

Volwassen MDD-tools ondersteunen de verfijning van modellen en transformaties tussen verschillende abstractieniveaus, waardoor aanpasbare model-naar-model en model-naar-code transformaties mogelijk zijn. Volgens Marín e.a. (2015) “moeten MDD-tools de uitvoering van modellen mogelijk maken, ook al zijn ze onvolledig, maar wel geldig”, wat modelcorrectie en -validatie in een vroeg stadium vergemakkelijkt.

Ondersteuning van het ontwikkelproces omvat een reeks functies. Tools moeten duidelijke feedback geven over fouten, idealiter door direct te wijzen naar de modelcomponenten die problemen veroorzaken, vergelijkbaar met hoe compilers problematische code markeren. Omdat software meestal door teams wordt ontwikkeld, moeten MDD-tools modelvergelijking, samenvoeging en versiebeheer ondersteunen.

Marín e.a. (2015) benadrukken dat “versiebeheer van modellen absoluut noodzakelijk is om industriële samenwerkingsprojecten te beheren, waarbij verschillende leden van een ontwikkelteam aan hetzelfde model kunnen werken”. Effectieve tools moeten snel compileren en implementeren, met een bijzonder efficiënte afhandeling van incrementele wijzigingen.

MDD-tools moeten integreren met bestaande systemen en ontwikkelomgevingen, waardoor verbindingen met ERP-systemen, legacy applicaties en andere bedrijfsinfrastructuur mogelijk worden. Een van de belangrijkste voordelen van MDD is dat een breder scala aan mensen kan deelnemen aan de ontwikkeling, waaronder bedrijfskundigen met beperkte programmeervaardigheden. Tools moeten de definitie en het gebruik van herbruikbare componenten, patronen en best practices in projecten vergemakkelijken.

Door deze functies op te nemen kunnen MDD-tools beter aansluiten op de behoeften van de industrie en de succesvolle toepassing van het MDD-paradigma in softwareontwikkelingsprojecten ondersteunen.



### 2.2.5. De evolutie naar low-code platformen

Model-Driven Development (MDD) is de laatste jaren sterk geëvolueerd, vooral met de opkomst van low-code platformen zoals Mendix. Deze platforms vertegenwoordigen een verfijning van MDD-principes, waardoor ze toegankelijker worden voor ontwikkelaars met verschillende vaardigheidsniveaus, terwijl de kernvoordelen van modelgedreven benaderingen behouden blijven.

Zoals Henkel en Stirna (2010) aantoonde in hun analyse van Mendix, proberen moderne MDD-tools een balans te vinden tussen abstractie en controle, waardoor ontwikkelaars op bedrijfsniveau kunnen redeneren terwijl ze toch voldoende controle houden over het systeemgedrag. Deze evolutie heeft MDD-principes toegankelijk gemaakt voor een breder publiek, waardoor softwareontwikkeling democratischer wordt dan alleen voor traditionele programmeerexperts. Henkel en Stirna (2010) benadrukken dat “low-code platforms zoals Mendix met succes de kloof hebben overbrugd tussen abstractie op hoog niveau en technische implementatie, waardoor zowel zakelijke belanghebbenden als ontwikkelaars effectief kunnen samenwerken.”

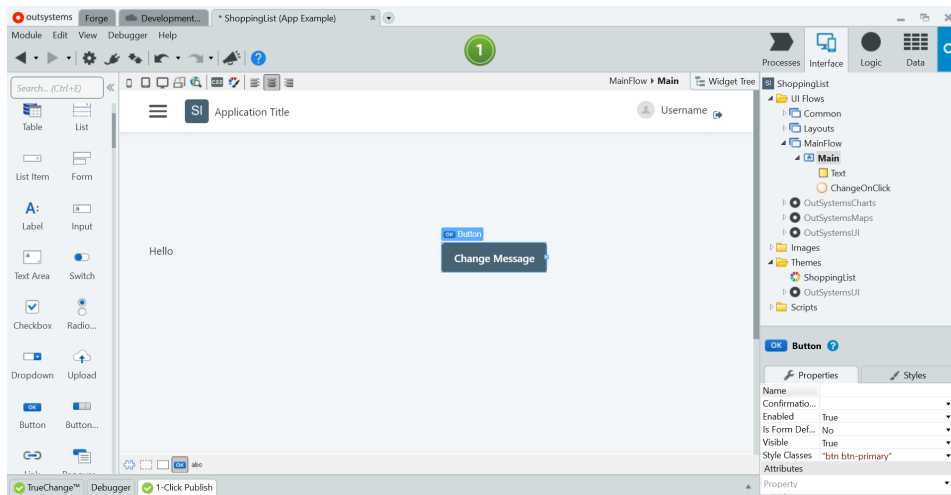
Concluderend kan gesteld worden dat Model-Driven Development een significante vooruitgang betekent in software engineering methodologie, door modellen te verheffen van secundaire documentatie tot primaire ontwikkelartefacten. Door het abstractieniveau te verhogen stelt MDD ontwikkelaars in staat om complexiteit effectiever te managen, waardoor de productiviteit kan toenemen, de kwaliteit kan verbeteren en softwareontwikkeling toegankelijker wordt voor een breder scala aan belanghebbenden. De integratie van MDD-principes in low-code platformen heeft het bereik verder vergroot, waardoor het een krachtig hulpmiddel is geworden voor moderne softwareontwikkeling.

## 2.3. Low-code platformen

In de volgende paragrafen zullen we de belangrijkste kenmerken, sterke punten en beperkingen van verschillende toonaangevende low-code ontwikkelplatformen onderzoeken en vergelijken, waaronder OutSystems, Joget DX en Mendix. Elk platform biedt unieke mogelijkheden en komt tegemoet aan verschillende organisatorische behoeften en use cases. Door hun benaderingen van visuele ontwikkeling, schaalbaarheid, inzetmogelijkheden, maatwerk en integratiemogelijkheden te onderzoeken, willen we een duidelijk inzicht geven in hoe deze platforms zich tot elkaar verhouden. Uiteindelijk zal deze vergelijking duidelijk maken waarom Mendix de meest uitgebreide en veelzijdige oplossing is, met de beste balans tussen flexibiliteit, schaalbaarheid en geavanceerde functies voor bedrijven die hun digitale transformatie willen versnellen.

### 2.3.1. OutSystems

OutSystems is een robuust low-code platform dat bekend staat om zijn visuele ontwikkelomgeving en mogelijkheden op enterprise-niveau.



**Figuur 2.2:** Visuele ontwikkelomgeving van OutSystems (Figueira-Putresza, 2021).

Het blinkt uit in snelle applicatieontwikkeling en biedt zowel cloud-native als on-premises implementatieopties, die tegemoet komen aan uiteenlopende organisatorische behoeften (Sido & Emon, 2024). OutSystems kan ook bogen op een levendig ecosysteem en community, samen met sterke beveiligingsfuncties en naleving van industriestandaarden. Het licentiemodel en de kostenbeperkingen kunnen echter onbetaalbaar zijn voor sommige organisaties, en de beperkte database-integratie en migratie-uitdagingen kunnen flexibiliteit in de weg staan (Sido & Emon, 2024). Daarnaast kunnen de kwaliteit van de documentatie en de geïsoleerde ondersteuning van de community barrières opwerpen voor ontwikkelaars die het potentieel van het platform willen maximaliseren.

### 2.3.2. Joget DX

Joget DX daarentegen is een open-source platform dat de nadruk legt op eenvoud en snelle applicatieontwikkeling. Het is vooral sterk in de ontwikkeling van progressieve webapps (PWA's), gebruikerservaring (UX) en integratie met DevOps-praktijken (Sido & Emon, 2024). Joget DX biedt ook uitbreidbaarheid via add-on builders en verbeterde workflowmogelijkheden, waardoor het een flexibele keuze is voor bedrijven die processen willen automatiseren. De beperkingen van het abonnementsmodel, de beperkte databasetoegang en de app-beperkingen kunnen de schaalbaarheid en aanpasbaarheid echter beperken (Sido & Emon, 2024). Bovendien kunnen de leercurve en aanpassingscomplexiteit van het platform uitdagingen vormen voor gebruikers die overstappen van traditionele ontwikkelmethoden.

### 2.3.3. Mendix

Mendix onderscheidt zich als het meest uitgebreide en veelzijdige low-code platform en biedt een breed scala aan functies voor zowel technische als niet-technische gebruikers. De modelgedreven ontwikkelaanpak, gecombineerd met microservices en containerisatie, zorgt voor schaalbaarheid, flexibiliteit en draagbaarheid. Mendix ondersteunt zowel cloud-native als on-premise infrastructuren en biedt daarmee flexibiliteit in de inzetmogelijkheden. Het volledige levenscyclusbeheer van het platform, de mogelijkheden voor kunstmatige intelligentie (AI) en machine learning (ML) en de functies voor procesautomatisering maken het een krachtig hulpmiddel voor ondernemingen (Sido & Emon, 2024). Bovendien zorgen de openheid en uitbreidbaarheid van Mendix voor een naadloze integratie met externe systemen en aanpassingen om aan specifieke bedrijfsbehoeften te voldoen. Hoewel Mendix een aantal beperkingen heeft, zoals beperkte aanpasbaarheid van thema's en potentiële vendor lock-in, wegen de algehele mogelijkheden en het gebruiksgemak op tegen deze nadelen (Sido & Emon, 2024).

### 2.3.4. Keuze voor Mendix als low-code platform

Hoewel OutSystems en Joget DX waardevolle functies en mogelijkheden bieden, komt Mendix naar voren als het beste platform vanwege zijn uitgebreide en flexibele benadering van low-code ontwikkeling. Het vermogen om complexe bedrijfsapplicaties te ondersteunen, gecombineerd met de schaalbaarheid, uitbreidbaarheid en sterke focus op gebruikerservaring, maakt het de ideale keuze voor organisaties die op efficiënte en effectieve wijze digitale transformatie willen stimuleren.

## 2.4. Mendix en high-code

### 2.4.1. Verschillende benaderingen

Low-code en high-code representeren twee verschillende benaderingen van softwareontwikkeling, elk met unieke voordelen en uitdagingen. Low-code platforms, zoals Mendix, bieden een visuele ontwikkelomgeving waarin applicaties grotendeels zonder handmatig programmeren kunnen worden gebouwd (Krouwel e.a., 2022). Dit versnelt de ontwikkeltijd en maakt softwareontwikkeling toegankelijker voor niet-technische gebruikers. Daarentegen biedt high-code, waarbij programmeertalen zoals Java of .NET worden gebruikt, maximale flexibiliteit en controle, wat essentieel is voor complexe of sterk aangepaste oplossingen (Krouwel e.a., 2022). Hoewel low-code efficiënter is en snellere iteraties mogelijk maakt, kan het beperkingen hebben in maatwerk en prestaties vergeleken met high-code. De keuze tussen beide hangt af van de behoeften van de organisatie: low-code is ideaal voor snelle, aanpasbare applicaties, terwijl high-code de voorkeur geniet voor diepgaande technische en schaalbare oplossingen.

**2.4.2. Enterprise Flexibiliteit door Model-Based Engineering (MBE)**

Krouwel e.a. (2022) benadrukt dat enterprise agility een cruciale succesfactor is in een steeds dynamischere markt, waarin bedrijven te maken hebben met hyperconcurrentie, veranderende regelgeving en technologische innovaties. Traditionele informatiesystemen vormen vaak een beperkende factor voor deze flexibiliteit, omdat ze hardcoded bedrijfsregels en structuren bevatten die moeilijk aanpasbaar zijn. Door Model-Based Engineering (MBE) te gebruiken, kunnen organisaties hun informatiesystemen ontwikkelen op basis van ontologische modellen, zoals die binnen de DEMO-methodologie worden gehanteerd. Deze modellen beschrijven de essentiële processen en interacties binnen een onderneming, los van specifieke implementaties. Dit maakt het mogelijk om snel en systematisch wijzigingen door te voeren, zonder dat ontwikkelaars handmatig code hoeven aan te passen. Het resultaat is een grotere wendbaarheid van zowel de organisatie als haar IT-systemen, waardoor bedrijven sneller kunnen inspelen op veranderingen zonder dat technologie een beperkende factor vormt

**2.4.3. Low-Code als Brug tussen Business en IT**

Een belangrijke conclusie uit Krouwel e.a. (2022) is dat low-code technologie niet alleen zorgt voor een snellere ontwikkeling van software, maar ook de samenwerking tussen business en IT aanzienlijk verbetert. Traditionele softwareontwikkeling vereist vaak uitgebreide specificaties en langdurige communicatie tussen ontwikkelaars en business stakeholders, wat leidt tot vertragingen en misinterpretaties. Low-code platforms, zoals Mendix, bieden een visuele ontwikkelomgeving waarin bedrijfsgebruikers direct kunnen meedenken en bijdragen aan het ontwerp van applicaties (Mendix, 2023). Dit verlaagt de drempel om wijzigingen door te voeren en zorgt ervoor dat IT-systemen beter aansluiten op de behoeften van de organisatie. Doordat bedrijfsprocessen en bijbehorende applicaties sneller en effectiever kunnen worden aangepast, wordt de time-to-market verkort en kunnen organisaties flexibeler inspelen op nieuwe kansen en uitdagingen

**2.4.4. Kunnen beiden elkaar aanvullen?**

Volgens Krouwel e.a. (2022) kunnen low-code en high-code elkaar versterken door de flexibiliteit van model-based engineering (MBE) te combineren met de aanpasbaarheid van traditionele codering. Low-code platforms zoals Mendix maken gebruik van abstracties en modelgestuurde technieken, waardoor applicaties sneller kunnen worden ontwikkeld zonder dat complexe code nodig is. Tegelijkertijd biedt high-code de mogelijkheid om de gegenereerde applicaties verder te verfijnen, bijvoorbeeld door aangepaste logica, integraties of prestatie-optimalisaties toe te voegen. In het onderzoek wordt benadrukt dat low-code vooral geschikt is om snel te reageren op veranderingen in bedrijfsprocessen, terwijl high-code nodig blijft voor diepgaande aanpassingen en geavanceerde automatiseringen. Door

beide methoden te combineren, ontstaat een hybride aanpak waarbij organisaties profiteren van de snelheid en wendbaarheid van low-code zonder in te boeten op de kracht en maatwerkopties van high-code.

## **2.5. Keuze tussen high-code en low-code?**

Bij het kiezen tussen high-code, low-code en no-code ontwikkelmethoden is het essentieel om vooraf duidelijk te bepalen welke functionaliteiten en mate van maatwerk je voor je applicatie nodig hebt (Ballejos, 2024). High-code ontwikkeling biedt maximale flexibiliteit en is geschikt voor complexe, op maat gemaakte oplossingen, maar vereist aanzienlijke tijd en technische expertise. Low-code platforms versnellen het ontwikkelproces door gebruik te maken van visuele tools en vooraf gebouwde componenten, wat ideaal is voor applicaties die snel moeten worden ontwikkeld met een zekere mate van aanpasbaarheid (Ballejos, 2024). No-code oplossingen stellen zelfs niet-technische gebruikers in staat om eenvoudige applicaties te creëren zonder enige programmeerkennis, wat handig is voor basisbehoeften maar beperkingen kent in complexiteit en maatwerk. Door vooraf je specifieke eisen en doelen te definiëren, kun je de ontwikkelmethode kiezen die het beste aansluit bij de behoeften van je project en organisatie.

# 3

## Methodologie

### 3.1. Methodologie

Dit onderzoek is opgedeeld in twee fasen: eerst het in kaart brengen van het probleemdomein (het ontbreken van een beslissingskader), gevolgd door onderzoek naar het oplossingsdomein (de ontwikkeling van het kader).

#### 3.1.1. Fase 1: Analyse van het probleemdomein

Om een grondig inzicht te krijgen in het probleemdomein worden de volgende onderzoeksmethoden gebruikt:

##### Analyse van historische projectdata

Om deelvragen 1 en 2 te beantwoorden ("Wat zijn de gevolgen van het ontbreken van een beslissingskader?" en "Welke problemen ontstaan er in projecten?"), wordt een gestructureerde analyse uitgevoerd van afgesloten projecten. Deze analyse omvat:

- Vergelijking van initiële projectschattingen versus werkelijke uitkomsten om impact op kosten en doorlooptijd te kwantificeren
- Analyse van projectdocumentatie om momenten te identificeren waar keuzes tussen low-code en high-code tot problemen leidden
- Inventarisatie van situaties waarin late overschakeling naar alternatieve ontwikkelmethoden nodig was
- Evaluatie van de financiële en tijdsimpact van deze late aanpassingen

##### Interviews met experts

Voor het beantwoorden van deelvragen 3 en 4 ("Wat zijn de huidige criteria?" en "Welke knelpunten ervaren projectmanagers?") worden diepte-interviews gehouden met:

- Projectmanagers over hun huidige besluitvormingsproces
- Architecten over hun ervaringen met technologie-keuzes
- Pre-sales consultants over hun aanpak bij het inschatten van projectgeschiedenis
- Ontwikkelaars over de uitdagingen die zij ervaren bij technologie-keuzes

### **Documentatieonderzoek**

Om deelvraag 3 verder te onderbouwen wordt bestaande interne documentatie geanalyseerd:

- Huidige richtlijnen en procedures voor projectaanpak
- Pre-sales documentatie en offertes
- Project kick-off documenten
- Architectuurbeslissingen en design documents

### **3.1.2. Fase 2: Onderzoek naar het oplossingsdomein**

Na het volledig in kaart brengen van het probleem, richt het onderzoek zich op het ontwikkelen van een oplossing middels:

### **Literatuuroverzicht**

Een diepgaande analyse van bestaande documentatie over Mendix en andere low-code platforms voor het beantwoorden van deelvraag 5 en 7:

- Officiële Mendix productgidsen
- Casestudies van derden en rapporten uit de industrie
- Online forums en ontwikkelaarsgemeenschappen
- Academische publicaties over low-code ontwikkeling

### **Praktisch onderzoek**

Uitvoering van praktische tests voor deelvraag 6 om de grenzen van het Mendix-platform te onderzoeken:

- Schaalbaarheid en prestaties
- Integratiemogelijkheden
- Aanpasbaarheid en uitbreidbaarheid
- Ontwikkelingssnelheid
- Beveiliging en compliance

**Ontwikkeling beslissingskader**

Voor het beantwoorden van deelvraag 8 ("Welke best practices en richtlijnen voor de keuze tussen Mendix en traditionele ontwikkeling kunnen worden gedestilleerd uit de projectervaringen en literatuur?") wordt op basis van alle verzamelde inzichten een gestructureerd beslissingskader ontwikkeld dat projectmanagers en architecten ondersteunt bij het maken van weloverwogen keuzes over de inzet van Mendix versus traditionele ontwikkelmethoden.



# 4

## Conclusie

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

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

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

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



# Onderzoeksvoorstel

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

## Samenvatting

In het digitale landschap van vandaag zoeken bedrijven steeds naar manieren om zakelijke applicaties efficiënter te ontwikkelen. Low-code ontwikkelplatformen, zoals Mendix, zijn de laatste jaren sterk gegroeid als een vervanging voor traditionele high-code ontwikkeling. Deze platformen beloven snellere ontwikkelingstijden en toegankelijkheid door visuele interfaces en herbruikbare componenten. Er is echter nog onduidelijkheid over wat de grenzen van de low-code-tool Mendix zijn bij het ontwikkelen van zakelijke applicaties, en wanneer high-code-ontwikkeling meer geschikt is.

Dit onderzoek duikt diep in de technische mogelijkheden en beperkingen van Mendix om deze centrale vraag te beantwoorden. Door de complexiteit van deze vraag te ontrafelen, zal de studie een genuanceerd beeld schetsen van de huidige stand van low-code technologie.

Een gemengde onderzoeksmethodologie zal worden gehanteerd, bestaande uit een diepgaande literatuurstudie, praktische experimenten en interviews met ervaren ontwikkelaars. Deze veelzijdige onderzoeksaanpak stelt ons in staat om de technische grenzen van Mendix grondig te analyseren, de prestaties in verschillende scenario's te evalueren en concrete richtlijnen te formuleren voor organisaties die overwegen low-code te implementeren.

Het onderzoek zal een gedetailleerde analyse geven van de technische beperkingen van Mendix en use cases identificeren waarin de ontwikkeling van high-code meer geschikt is. Het zal ook een beslissingskader bieden, zodat bedrijven en an-

dere instellingen kunnen kiezen tussen low-code en high-code methoden. Uiteindelijk zal het richtlijnen bevatten voor het optimaal combineren van de twee methoden om de beste resultaten te behalen.

Gezien de groeiende adoptie van low-code platforms is er dringend behoefte aan objectief onderzoek naar hun mogelijkheden en beperkingen. Deze bachelorproef helpt om deze behoefte aan kennis te vervullen en biedt handige adviezen voor organisaties die overwegen om low-code of traditionele ontwikkelmethoden toe te passen.

## **A.1. Inleiding**

Apvine, een toonaangevend IT-consultancybedrijf, richt zich op het creëren van applicaties met low-code platforms, voornamelijk Mendix. Deze aanpak is zeer effectief gebleken voor de meeste van hun projecten en maakt snelle ontwikkeling en implementatie mogelijk. Toch zijn er situaties waarin projecten uitdagingen bieden die de grenzen van low-code platforms opzoeken. Denk hierbij aan ingewikkelde bedrijfslogica, intensieve real-time verwerking of complexe integraties met legacy-systemen, die de mogelijkheden van een puur low-code methode kunnen overstijgen.

In dergelijke situaties kan het noodzakelijk zijn om over te stappen op een hybride of high-code methode. Het is echter niet eenvoudig om te beslissen wanneer deze overgang moet gebeuren. Zonder duidelijke richtlijnen loopt Apvine het risico op vertragingen, hogere uitgaven en ontevreden klanten door late of onverwachte aanpassingen in de ontwikkelingsstrategie. Om deze valkuilen te vermijden, richt dit onderzoek zich op de centrale vraag: “Wat zijn de grenzen van de low-code-tool Mendix bij het ontwikkelen van zakelijke applicaties, en wanneer is high-code-ontwikkeling meer geschikt?”

Dit onderzoek heeft als doel een raamwerk voor besluitvorming te ontwikkelen dat projectmanagers en architecten van Apvine ondersteunt bij het beoordelen of ze moeten blijven met low-code of overstappen naar high-code voor een specifiek project. Het raamwerk zal worden gebaseerd op:

- Systematische analyse van eerdere projecten om gemeenschappelijke succesfactoren en uitdagingen te identificeren.
- Richtlijnen voor het implementeren van hybride methoden om zowel flexibiliteit als complexiteit te combineren.
- Een hulpmiddel dat beslissingen ondersteunt tijdens de fases van pre-sales en planning.

Om de centrale vraag te beantwoorden, worden de volgende deelvragen onder-

zocht:

**Probleemdomein:**

1. Wat zijn de gevolgen van het ontbreken van een beslissingskader voor de keuze tussen low-code en high-code bij Apvine?
2. Welke problemen ontstaan er in projecten door het gebrek aan richtlijnen voor de keuze tussen ontwikkelmethoden?
3. Wat zijn de huidige criteria die Apvine gebruikt bij het kiezen tussen low-code en high-code ontwikkeling?
4. Welke knelpunten ervaren projectmanagers en architecten bij het maken van de keuze tussen low-code en high-code?

**Oplossingsdomein:**

5. Wat zijn de technische beperkingen van Mendix bij het omgaan met complexe bedrijfslogica, intensieve real-time verwerking en integraties met legacy-systemen?
6. In welke specifieke scenario's binnen Mendix-projecten kan een hybride of high-code oplossing nodig zijn, en wat zijn de triggers voor het maken van deze overstap?
7. Hoe kunnen projectomvang, tijdslijnen en klantvereisten de beslissing beïnvloeden om wel of niet met Mendix (low-code) te werken, of over te schakelen naar high-code?
8. Welke lessen kunnen worden getrokken uit de ervaring van eerdere projecten bij Apvine waarin Mendix werd ingezet, en hoe kunnen deze inzichten helpen bij het bepalen wanneer een hybride of high-code oplossing nodig is?

Het beslissingskader dat uit dit onderzoek voortkomt, biedt een gestructureerde aanpak om organisaties te helpen bij het maken van een weloverwogen keuze tussen low-code en high-code ontwikkelmethoden. Met dit kader zijn bedrijven zoals Apvine in staat om niet alleen effectiever te plannen, maar ook de risico's van late veranderingen in projecten te verlagen, de ontwikkelingskosten effectiever te beheersen en hoogwaardige oplossingen te bieden die voldoen aan de unieke wensen van hun klanten.

## A.2. Literatuurstudie

### A.2.1. Low-code en Mendix

Het landschap van digitale klantervaringen evolueert snel, aangewakkerd door de groeiende verwachtingen van klanten en de behoefte voor bedrijven om steeds consistente en hoogwaardige ervaringen te bieden. Zoals beschreven in de tekst “Deliver Standout Digital Customer Experiences with Low-Code” door het bedrijf Mendix (2023), worden bedrijven geconfronteerd met aanzienlijke uitdagingen om aan deze eisen te voldoen vanwege de complexiteit van traditionele ontwikkelpraktijken en beperkte middelen.

Ongelijksoortige processen en afzonderlijke oplossingen leiden vaak tot onsamenvhangende ervaringen die niet aansluiten bij de wensen van de klant (**Mendix2023**). Ondernemingen moeten worstelen met het beheer van meerdere complexe technologieën, afzonderlijke applicaties en een vertraagde time-to-market. Dit alles hindert hen erin om zich snel aan te passen en de ervaringen te bieden die klanten verwachten.

Low-code ontwikkelplatforms zijn aangetoond als een effectieve oplossing voor deze problemen, waardoor bedrijven de controle kunnen nemen over de klantervaring. Low-code platformen maken zowel technische als niet-technische teams mogelijk om samen te werken aan de snelle ontwikkeling van multi-ervaringstoepassingen door snelle ontwikkelingskansen en een klantgerichte benadering te bieden.

Met name het low-code Mendix-platform is ontworpen om bedrijven te ondersteunen bij digitalisering en om consistente, toekomstbestendige contactpunten voor klanten te creëren voor diverse kanalen en apparaten. Dankzij functies zoals automatisering, AI-ondersteuning en cloud-native schaalbaarheid stelt Mendix bedrijven in staat om snel applicaties te bouwen en te implementeren, terwijl ze tegelijkertijd flexibel en consistent blijven. (Mendix, 2023)

### A.2.2. Low-code use cases

Low-code ontwikkelplatforms maken het voor organisaties mogelijk om een breed scala aan bedrijfsapplicaties te creëren, zodat ze hun bedrijfsprocessen kunnen faciliteren en de gebruikerservaringen kunnen optimaliseren. Zo zijn er veelvoorkomende use cases:

- Legacy modernisering

Met low-code hebben bedrijven de mogelijkheid om legacy-systemen te combineren met nieuwere technologieën, bestaande kansen te vergroten of verouderde systemen te vervangen om in te spelen op veranderende bedrijfsbehoeften. Banco de Occidente maakte gebruik van een low-code platform om hun verouderde systemen te integreren en hun processen te verbeteren, wat leidde tot een betere ervaring voor zowel klanten als medewerkers (Bunce, 2024b).

- Portalen  
Low-code stelt gebruikers in staat om op maat gemaakte, webgebaseerde portals te creëren die hen van belangrijke informatie en acties voorzien, wat de efficiëntie en gebruikerservaring ten goede komt. DHL Group heeft een platform ontwikkeld voor het beheer van leveranciersstamgegevens, gebaseerd op een low-code platform, waarmee het werk wordt geautomatiseerd. (Bunce, 2024b).
- Mobiele apps  
Low-code applicaties functioneren op diverse apparaten zonder dat er hercodering vereist is, waardoor gebruikers toegang krijgen tot het apparaat dat ze verkiezen. Super Bock Group maakte gebruik van low-code voor het ontwikkelen van een mobiel goedkeuringsproces voor aankoopaanvragen, wat resulteerde in kortere responstijden (Bunce, 2024a).
- Integratie  
Low-code platforms maken het mogelijk om verschillende systemen en gegevensbronnen te integreren, zodat gebruikers in één interface toegang krijgen tot informatie. Zo maakte bijvoorbeeld Unilever gebruik van een low-code platform om hun SAP-organisaties en prijsstrategieën voor klanten te integreren, wat resulteerde in een hogere efficiëntie en precisie (Bunce, 2024b).

Dit zijn slechts een paar voorbeelden van de veelzijdigheid en brede toepassing van low-code ontwikkelplatformen in diverse industrieën en gebruikssituaties. Bij een volledige literatuurstudie kan/zal dit verder uitgewerkt worden.

### A.2.3. Beperkingen van low-code

Hoewel low-code ontwikkelplatformen veel voordelen opleveren bij het versnellen van de applicatielevering, hebben ze ook enkele opvallende nadelen waar organisaties rekening mee moeten houden. Het artikel “9 Low Code Limitations in 2024 to Know About” (Malak, 2024) beschrijft verschillende belangrijke beperkingen die verband houden met low-code methoden. Deze beperkingen omvatten:

- Een tekort aan controle over de automatisch gegenereerde code kan een obstakel vormen voor ingewikkelde applicaties die codeoptimalisatie vereisen.
- Potentiële vendor lock-in door eigen frameworks of programmeertalen maakt het lastig om van platform te veranderen of met andere systemen te integreren.
- Er zijn beperkte mogelijkheden voor aanpassing, omdat low-code platforms beter geschikt zijn voor algemene functies dan voor ingewikkelde, specifieke vereisten binnen een bepaalde sector.

- Beveiligingsproblemen ontstaan doordat low-code ontwikkeling sommige beveiligingsaspecten kan over het hoofd zien, vooral in streng gereguleerde sectoren.
- Complexiteit van integratie komt voort uit het feit dat low-code platforms mogelijk niet perfect kunnen integreren met de huidige infrastructuur van een organisatie en externe diensten.
- Beperkingen in de schaalbaarheid, omdat low-code applicaties problemen kunnen ondervinden bij het aanpassen aan de groeiende vraag van gebruikers en het volume aan gegevens.

De geschiktheid van low-code voor een specifiek project zal uiteindelijk afhankelijk zijn van elementen zoals de ingewikkeldheid van de applicatie, de technische expertise van de organisatie, de behoefte aan beveiliging en naleving, evenals de langetermijnschaalbaarheidseisen. Door de in dit artikel genoemde beperkingen grondig te beoordelen, kunnen bedrijven beter geïnformeerde keuzes maken over het gebruik van low-code ontwikkeling en wanneer traditionele codering beter aansluit.

#### **A.2.4. Low-code versus high-code**

Hoewel ze fundamenteel van elkaar verschillen, kunnen low-code en high-code ontwikkelmethoden elkaar aanvullen om aan diverse projecteisen te voldoen. High-code ontwikkeling levert ongeëvenaarde controle, flexibiliteit en de optie om complexe, aangepaste toepassingen vanaf nul te ontwikkelen, wat het perfect maakt voor grote of ingewikkelde projecten die afhankelijk zijn van geavanceerde functies en integraties. Low-code-platforms vergemakkelijken het ontwikkelproces met visuele hulpmiddelen en kant-en-klare componenten, wat ervoor zorgt dat prototypes en de ontwikkeling van middelmatig complexe toepassingen sneller verlopen.

Door deze methoden in evenwicht te brengen, kunnen bedrijven hun productiviteit maximaliseren, de time-to-market voor bepaalde projecten verkorten en ervaren ontwikkelaars inzetten waar ze het meest nodig zijn, zoals Ballejos (2024) benadrukt in uitgebreide overzicht van de verschillen en mogelijke wisselwerking tussen deze methodologieën.

### **A.3. Methodologie**

Dit onderzoek is opgedeeld in twee fasen: eerst het in kaart brengen van het probleemdomain (het ontbreken van een beslissingskader), gevolgd door onderzoek naar het oplossingsdomain (de ontwikkeling van het kader).



**A.3.1. Fase 1: Analyse van het probleemdomein**

Om een grondig inzicht te krijgen in het probleemdomein worden de volgende onderzoeksmethoden gebruikt:

**Analyse van historische projectdata**

Om deelvragen 1 en 2 te beantwoorden ("Wat zijn de gevolgen van het ontbreken van een beslissingskader?" en "Welke problemen ontstaan er in projecten?"), wordt een gestructureerde analyse uitgevoerd van afgesloten projecten. Deze analyse omvat:

- Vergelijking van initiële projectschattingen versus werkelijke uitkomsten om impact op kosten en doorlooptijd te kwantificeren
- Analyse van projectdocumentatie om momenten te identificeren waar keuzes tussen low-code en high-code tot problemen leidden
- Inventarisatie van situaties waarin late overschakeling naar alternatieve ontwikkelmethoden nodig was
- Evaluatie van de financiële en tijdsimpact van deze late aanpassingen

**Interviews met experts**

Voor het beantwoorden van deelvragen 3 en 4 ("Wat zijn de huidige criteria?" en "Welke knelpunten ervaren projectmanagers?") worden diepte-interviews gehouden met:

- Projectmanagers over hun huidige besluitvormingsproces
- Architecten over hun ervaringen met technologie-keuzes
- Pre-sales consultants over hun aanpak bij het inschatten van projectgeschiktheid
- Ontwikkelaars over de uitdagingen die zij ervaren bij technologie-keuzes

**Documentatieonderzoek**

Om deelvraag 3 verder te onderbouwen wordt bestaande interne documentatie geanalyseerd:

- Huidige richtlijnen en procedures voor projectaanpak
- Pre-sales documentatie en offertes
- Project kick-off documenten
- Architectuurbeslissingen en design documents

**A.3.2. Fase 2: Onderzoek naar het oplossingsdomein**

Na het volledig in kaart brengen van het probleem, richt het onderzoek zich op het ontwikkelen van een oplossing middels:

**Literatuuroverzicht**

Een diepgaande analyse van bestaande documentatie over Mendix en andere low-code platforms voor het beantwoorden van deelvraag 5 en 7:

- Officiële Mendix productgidsen
- Casestudies van derden en rapporten uit de industrie
- Online forums en ontwikkelaarsgemeenschappen
- Academische publicaties over low-code ontwikkeling

**Praktisch onderzoek**

Uitvoering van praktische tests voor deelvraag 6 om de grenzen van het Mendix-platform te onderzoeken:

- Schaalbaarheid en prestaties
- Integratiemogelijkheden
- Aanpasbaarheid en uitbreidbaarheid
- Ontwikkelingssnelheid
- Beveiliging en compliance

**Ontwikkeling beslissingskader**

Voor het beantwoorden van deelvraag 8 ("Welke best practices en richtlijnen voor de keuze tussen Mendix en traditionele ontwikkeling kunnen worden gedestilleerd uit de projectervaringen en literatuur?") wordt op basis van alle verzamelde inzichten een gestructureerd beslissingskader ontwikkeld dat projectmanagers en architecten ondersteunt bij het maken van weloverwogen keuzes over de inzet van Mendix versus traditionele ontwikkelmethoden.

**A.4. Verwacht resultaat, conclusie**

Door het combineren van de kennis uit het literatuuronderzoek, de praktische bevindingen uit de experimenten en de inzichten uit de interviews met experts, heeft het onderzoek als doel een uitgebreid begrip te ontwikkelen van de mogelijkheden en beperkingen van Mendix in de context van de ontwikkeling van bedrijfsapplicaties.

Als afsluiting van dit onderzoek wordt ook een beslissingskader opgesteld dat organisaties een gestructureerde aanpak biedt om een best passende keuze te maken tussen low-code en high-code ontwikkelmethoden. Dit kader stelt bedrijven zoals Apvine in staat om niet alleen effectiever te plannen, maar ook de risico's van late veranderingen in projecten te beperken. Hierdoor worden de ontwikkelingskosten beter beheerd en hoogwaardige oplossingen opgeleverd die aansluiten bij de unieke wensen van hun klanten.

# Bibliografie

- Ballejos, L. (2024, juni 6). *High Code vs. Low Code vs. No Code: Navigating the Best Coding Solutions for Your Needs*. Verkregen november 15, 2024, van <https://www.ninjaone.com/blog/high-code-vs-low-code-vs-no-code/>
- Bunce, C. (2024a, april 2). *Low-Code Use Cases: What Can You Build With a Low-Code Platform?* Verkregen november 15, 2024, van <https://www.bizagi.com/en/blog/low-code-use-cases>
- Bunce, C. (2024b, mei 21). *Low-Code vs High-Code: Decoding the Two Development Approaches*. Verkregen november 15, 2024, van <https://www.bizagi.com/en/blog/low-code-vs-high-code>
- Case, A. F. (1985). Computer-aided software engineering (CASE): technology for improving software development productivity. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 17(1), 35–43. <https://doi.org/https://doi.org/10.1145/1040694.1040698>
- Figueira-Putresza, Y. (2021). OutSystems tutorial: Learn low-code development and get your first certificate. Verkregen maart 7, 2025, van <https://pretius.com/blog/outsystems-tutorial/>
- Hailpern, B., & Tarr, P. (2006). Model-driven development: The good, the bad, and the ugly. In *IBM Systems Journal* (pp. 451–461, Deel 45). IBM. <https://doi.org/https://doi.org/10.1147/sj.453.0451>
- Henkel, M., & Stirna, J. (2010). Pondering on the Key Functionality of Model Driven Development Tools: The Case of Mendix. In *Perspectives in Business Informatics Research* (pp. 146–160). Springer Berlin Heidelberg. [https://doi.org/https://doi.org/10.1007/978-3-642-16101-8\\_12](https://doi.org/https://doi.org/10.1007/978-3-642-16101-8_12)
- Hermans, M. P., & Mileff, P. (2023). Short introduction to Mendix. <https://doi.org/https://doi.org/10.32968/psaie.2023.3.5>
- Krouwel, M. R., Op 't Land, M., & Proper, H. A. (2022). Generating Low-Code Applications from Enterprise Ontology. In *The Practice of Enterprise Modeling* (pp. 18–32). Springer International Publishing. [https://doi.org/https://doi.org/10.1007/978-3-031-21488-2\\_2](https://doi.org/https://doi.org/10.1007/978-3-031-21488-2_2)
- Malak, H. A. (2024, oktober 13). *9 Low Code Limitations in 2024 to Know About*. Verkregen november 15, 2024, van <https://theecmconsultant.com/low-code-limitations/>
- Marín, B., Salinas, A., Morandé, J., Giachetti, G., & de la Vara, J. L. (2015). Main Features for MDD Tools: An Exploratory Study. In *Model-Driven Engineering and Soft-*

- ware Development (pp. 183–196). Springer International Publishing. [https://doi.org/10.1007/978-3-319-25156-1\\_12](https://doi.org/10.1007/978-3-319-25156-1_12)
- Mendix. (2023, maart 14). *Deliver Standout Digital Customer Experiences with Low-Code*. <https://www.mendix.com/strategies/digital-customer-experiences/>
- Sido, N., & Emon, E. A. (2024, mei 31). *Low/No Code Development and Generative AI*.
- Soley, R., & the OMG Staff Strategy Group. (2000). Model Driven Architecture. Verkregen maart 7, 2025, van [https://www.vico.org/aRecursosMDA/MDA\\_paper3\\_2.pdf](https://www.vico.org/aRecursosMDA/MDA_paper3_2.pdf)
- Sufi, F. (2023). Algorithms in Low-Code-No-Code for Research Applications: A Practical Review. *Algorithms*, 16(2), 108. <https://doi.org/https://doi.org/10.3390/a16020108>
- van Oosten, A. (2020, juni 11). *Achieve New Levels of Business Value with Low Code*. Mendix. <https://www.mendix.com/blog/achieve-new-levels-of-business-value-with-low-code/>
- Yerukala, M. (2022, oktober 10). *What is Mendix*. MindMajix. <https://mindmajix.com/mendix-tutorial#benefits>