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

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

Samenvatting

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

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

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

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

Inhoudsopgave

Lijst van figuren

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 klantontevredenheid 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. Onderzoeksdoelstelling

Het primaire doel van dit onderzoek is het ontwikkelen van een uitgebreid beslissingskader dat projectmanagers en softwarearchitecten van Apvine in staat stelt om gefundeerde keuzes te maken tussen low-code (Mendix) en high-code ontwikkelingsmethoden. Dit kader zal worden ontworpen om de technische beperkingen van Mendix te identificeren en te bepalen wanneer de overstap naar high-code oplossingen noodzakelijk wordt.

De concrete resultaten van dit onderzoek zullen omvatten:

- Een gedetailleerde analyse van de beperkingen van Mendix bij het verwerken van complexe bedrijfslogica, real-time gegevensverwerking en integraties met legacy-systemen
- Een gestructureerd beslissingskader met duidelijke criteria voor het bepalen van de optimale ontwikkelingsaanpak
- Een reeks praktische richtlijnen en beoordelingsinstrumenten die projectmanagers kunnen toepassen in de beginfase van projectplanning
- Casestudies van eerdere Apvine-projecten, die succesvolle en uitdagende implementaties van low-code oplossingen illustreren
- Aanbevelingen voor het implementeren van hybride benaderingen wanneer dit passend is

Het succes van dit onderzoek zal worden gemeten aan de hand van:

- De toepasbaarheid van het kader op verschillende projecttypen en klantvereisten
- Validatie door technische experts en projectmanagers van Apvine met betrekking tot de praktische bruikbaarheid van het kader
- Het vermogen om potentiële technische knelpunten te voorspellen voordat ze impact hebben op projectplanningen of budgetten
- Duidelijke documentatie van het besluitvormingsproces die met klanten kan worden gedeeld om de transparantie te verbeteren

Het eindresultaat zal een uitgebreid rapport zijn met het beslissingskader, vergezeld van praktische hulpmiddelen (zoals beoordelingschecklists en beslisbomen) die direct kunnen worden geïntegreerd in de projectmethodologie van Apvine.

1.4. Opzet van deze bachelorproef

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

2

Stand van zaken

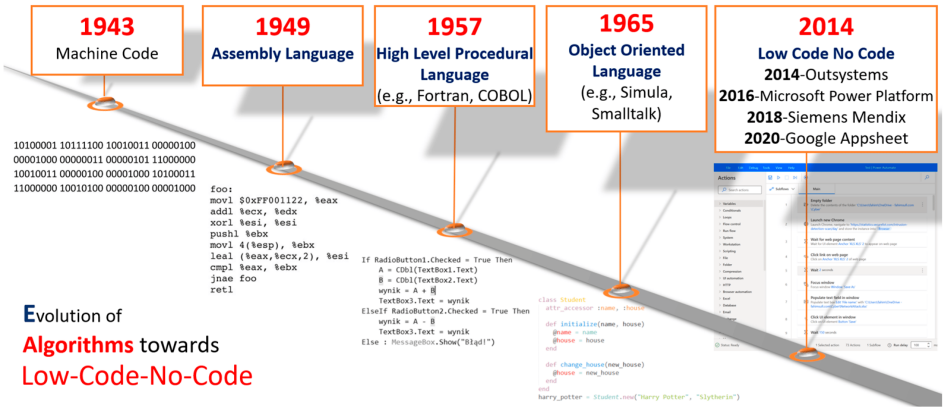
Mendix vertegenwoordigt een van de toonaangevende low-code ontwikkelplatformen in het huidige softwareontwikkelingslandschap (Hermans & Mileff, 2023). Als implementatie van Model-Driven Development (MDD) principes stelt Mendix zowel professionele ontwikkelaars als zakelijke gebruikers in staat om applicaties 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.

2.1. 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.

2.1.1. 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 verminderen.



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

2.1.2. Historische context en evolutie

Model Driven Development heeft een rijke geschiedenis die meer dan 40 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 OptimalJ. Hoewel beide soorten tools MDD ondersteunen door abstractieniveaus te verhogen, bieden tools zoals OptimalJ 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.

2.1.3. 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 codeerinspanning 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.

2.1.4. 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.

2.1.5. De evolutie naar low-code platformen

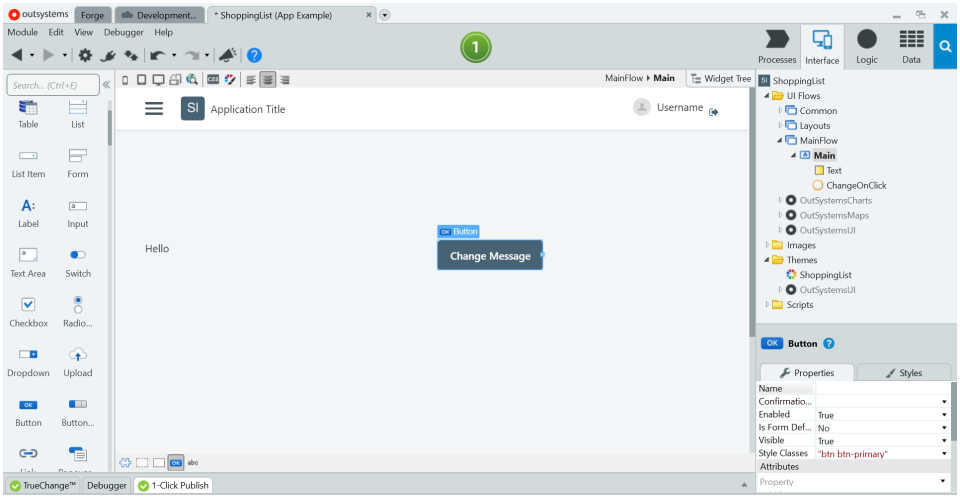
Model-Driven Development (MDD) is de laatste jaren sterk geëvolueerd, vooral met de opkomst van low-code platformen zoals Mendix. Deze platformen vertegenwoordigen een verfijning van MDD-principes, waardoor ze toegankelijker worden voor ontwikkelaars met verschillende vaardigheidsniveaus, terwijl de kernvoordelen van modelgedreven benaderingen behouden blijven.
Zoals Henkel en Stirna (2010) aantoonden 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 platforms heeft het bereik verder vergroot, waardoor het een krachtig hulpmiddel is geworden voor moderne softwareontwikkeling.

2.2. 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.

2.2.1. OutSystems

OutSystems is een robuust low-code platform dat bekend staat om zijn uitgebreide functies en enterprise-gerichte capaciteiten. Het platform onderscheidt zich door een intuïtieve visuele ontwikkelomgeving waarmee ontwikkelaars via drag-and-drop interfaces en voorgedefinieerde componenten snel applicaties kunnen bouwen. Deze gebruiksvriendelijke interface stelt zowel ervaren ontwikkelaars als gebruikers met minder technische kennis in staat om efficiënt te werken (Sido & Emon, 2024). Op het gebied van schaalbaarheid biedt OutSystems sterke prestaties voor enterprise-toepassingen, waarbij de architectuur is ontworpen om mee te groeien met toenemende gebruikersaantallen en datavolumes. Het platform ondersteunt zowel cloud-native als on-premises implementatieopties, waardoor organisaties flexibel kunnen kiezen voor de deploymentstrategie die het beste past bij hun infrastructuur en compliance-vereisten.

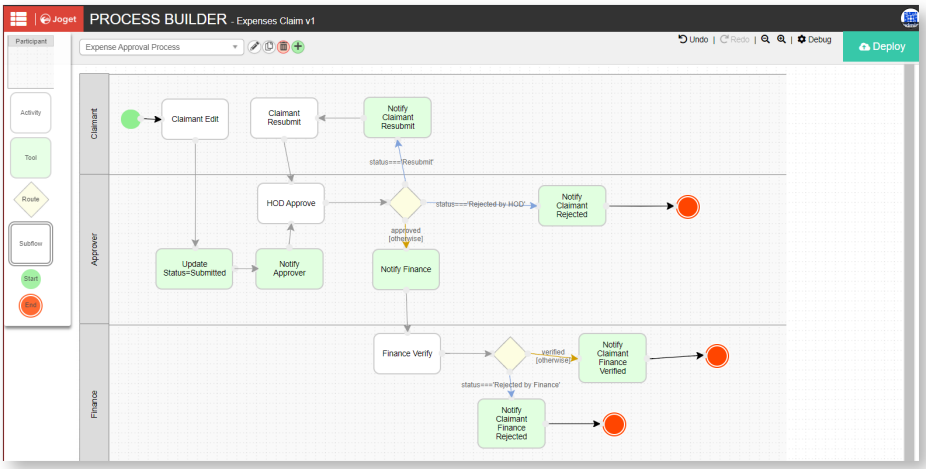


Figuur 2.2: Visuele ontwikkelomgeving van OutSystems (Figueira-Putresza, 2021).

Wat betreft maatwerk biedt OutSystems uitgebreide aanpassingsmogelijkheden via eigen extensies en de integratie van custom code wanneer de standaard visuele ontwikkelingstools niet toereikend zijn. Dit stelt ontwikkelaars in staat om complexe bedrijfslogica te implementeren, hoewel dit soms ten koste kan gaan van de ontwikkelsnelheid (Sido & Emon, 2024). Voor integraties beschikt het platform over diverse mogelijkheden om te koppelen met externe systemen en API's, maar er zijn beperkingen bij de integratie met sommige databases en uitdagingen bij migraties, vooral bij complexe legacy-systemen. OutSystems kan verder 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, terwijl de kwaliteit van de documentatie en de geïsoleerde ondersteuning van de community barrières kunnen opwerpen voor ontwikkelaars die het volledige potentieel van het platform willen benutten (Sido & Emon, 2024).

2.2.2. Joget DX

Joget DX is een open-source low-code platform dat zich onderscheidt door zijn toegankelijkheid en focus op snelle applicatieontwikkeling. Op het gebied van visuele ontwikkeling biedt het platform een intuïtieve interface die ontworpen is voor eenvoud, waardoor ook niet-technische gebruikers snel aan de slag kunnen met het bouwen van applicaties. Het is vooral sterk in de ontwikkeling van progressieve webapps (PWA's) en legt veel nadruk op gebruikerservaring (UX), waardoor eindgebruikers kunnen profiteren van moderne, responsieve interfaces (Sido & Emon, 2024). Qua schaalbaarheid kent Joget DX enkele beperkingen door het abonnementsmodel en app-beperkingen, wat uitdagingen kan opleveren voor grootschalige enterprise-toepassingen die hoge prestatie-eisen stellen.

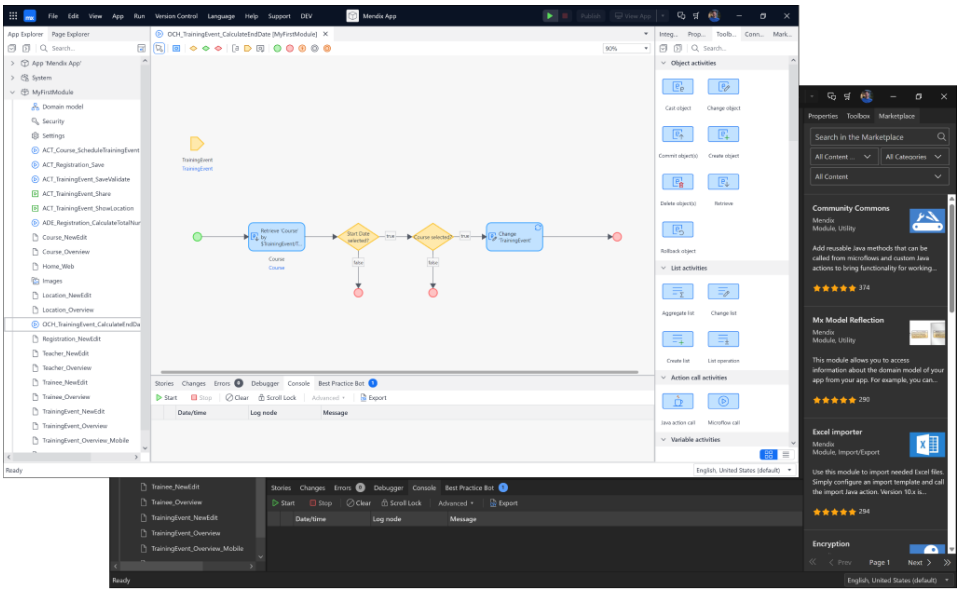


Figuur 2.3: Visuele ontwikkelomgeving van Joget DX (Hugo, 2024).

Voor inzetmogelijkheden biedt het platform goede integratie met DevOps-praktijken, waardoor het goed past binnen moderne ontwikkelworkflows en continuous delivery-processen. Het open-source karakter geeft organisaties flexibiliteit in de implementatie, hoewel de beperkte databasetoegang sommige deployment-scenario's kan bemoeilijken. Op het gebied van maatwerk beschikt Joget DX over uitbreidbaarheid via add-on builders en verbeterde workflowmogelijkheden, waardoor het een flexibele keuze is voor bedrijven die specifieke bedrijfsprocessen willen automatiseren en aanpassen aan hun behoeften (Sido & Emon, 2024). De integratiemogelijkheden zijn redelijk robuust voor standaard use-cases, maar kunnen beperkingen vertonen bij complexere scenario's of legacy-systemen, wat de algehele aanpasbaarheid kan verminderen. Bovendien kunnen de leercurve en aanpassingscomplexiteit van het platform uitdagingen vormen voor gebruikers die overstappen van traditionele ontwikkelmethoden, wat extra training en gewenningstijd kan vereisen.

2.2.3. 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).



Figuur 2.4: Visuele ontwikkelomgeving van Mendix (Mendix, 2025).

2.2.4. 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. Zoals de vergelijkingstabel laat zien, blinkt Mendix uit op alle belangrijke criteria: visuele ontwikkeling, schaalbaarheid, inzetmogelijkheden, maatwerk en integratiemogelijkheden. Het vermogen van Mendix om complexe bedrijfsapplicaties te ondersteunen met zijn geavanceerde visuele interface is superieur aan de concurrentie. De uitstekende schaalbaarheid voor enterprise-toepassingen overtreft de beperkingen die bij Joget DX worden er-

Criteria	**Mendix**	**OutSystems**	**Joget DX**
Visuele ontwikkeling	Zeer gebruiksvriendelijk; geschikt voor alle niveau's	Intuïtieve drag-and-drop interface	Eenvoudige interface gericht op PWA's
Schaalbaarheid	Uitstekend voor enterprise; hoge belasting	Sterk; ontworpen voor groei	Beperkt door abonnementsmodel
Inzetmogelijkheden	Veelzijdig (cloud/on-premises/hybrid); CI/CD-integratie	Cloud en on-premises opties	DevOps-integratie; beperkte database-opties
Maatwerk	Uitgebreid; Java-extensies; krachtige Microflow	Extensies en custom code mogelijk	Add-on builders; goede workflows
Integratiemogelijkheden	Uitstekende API's; veel connectors; legacy-ondersteuning	Diverse opties; enkele database-beperkingen	Adequaat voor standaard gebruik; beperkingen bij complexiteit

Tabel 2.1: Vergelijking Low-code platformen.

varen. Daarnaast biedt Mendix meer veelzijdige deployment-opties dan beide concurrenten, terwijl de uitgebreide aanpassingsmogelijkheden en krachtige Microflow-editor meer flexibiliteit bieden dan de alternatieven. Waar OutSystems worstelt met databaseintegratie en Joget DX beperkt wordt in complexere integratiescenario's, biedt Mendix uitstekende API-integratie met uitgebreide connectors voor externe systemen. Deze combinatie van sterke punten maakt Mendix de ideale keuze voor organisaties die op efficiënte en effectieve wijze digitale transformatie willen stimuleren, ongeacht de complexiteit van hun bedrijfsprocessen of technische vereisten.

2.3. Mendix en high-code

2.3.1. 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 groten-

deels 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.

2.3.2. 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

2.3.3. 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

2.3.4. 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.

2.4. Keuze tussen high-code en low-code?

Bij het kiezen tussen high-code en low-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. Bij deze analyse worden initiële projectschattingen vergeleken met werkelijke uitkomsten om de impact op kosten en doorlooptijd nauwkeurig te kwantificeren. Daarnaast wordt de projectdocumentatie systematisch doorgenomen met als doel het identificeren van specifieke momenten waarop keuzes tussen low-code en high-code ontwikkelmethoden tot problemen hebben geleid. Het onderzoek inventariseert ook situaties waarin projectteams gedwongen waren om tijdens het project over te schakelen naar alternatieve ontwikkelmethoden vanwege onvoorziene beperkingen. Ten slotte wordt de financiële en tijdsimpact van deze late aanpassingen grondig geëvalueerd om de werkelijke kosten van suboptimale initiële technologiekeuzes inzichtelijk te maken.

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 verschillende belanghebbenden binnen het ontwikkelproces. Projectmanagers worden geïnterviewd over hun huidige besluitvormingsproces bij technologiekeuzes, waarbij specifiek wordt gefocust op de impliciete en expliciete criteria die zij hanteren. Daarnaast delen architecten hun ervaringen met technologiekeuzes, inclusief de technische overwegingen die doorslaggevend zijn bij het maken van platformkeuzes. Pre-sales consultants worden bevraagd over hun methodiek bij het inschatten van projectgeschiktheid voor verschillende ontwikkelplatformen in de offertefase. Tot slot worden ontwikkelaars geïnterviewd over de praktische uitdagingen die zij ervaren wanneer technologiekeuzes zijn gemaakt en zij deze moeten implementeren, met bijzondere aandacht voor de discrepantie tussen verwachtingen en werkelijkheid.

Documentatieonderzoek

Om deelvraag 3 verder te onderbouwen wordt bestaande interne documentatie geanalyseerd. Dit omvat een grondige bestudering van de huidige richtlijnen en procedures voor projectaanpak, waarin impliciet of expliciet keuzes worden gemaakt over ontwikkelmethoden. Ook worden pre-sales documentatie en offertes onder de loep genomen om inzicht te krijgen in de initiële afwegingen en beloftes die worden gedaan voordat een project daadwerkelijk start. Project kick-off documenten worden onderzocht om de uitgangspunten en verwachtingen aan het begin van projecten te identificeren, met specifieke aandacht voor de technologiekeuzes die in deze vroege fase worden vastgelegd. Tenslotte worden architectuurbeslissingen en design documents geanalyseerd om te begrijpen hoe technische overwegingen worden gedocumenteerd en gecommuniceerd binnen projectteams, en hoe deze documenten bijdragen aan het besluitvormingsproces rondom low-code versus high-code ontwikkeling.

Analyse van kritieke use cases

Om inzicht te krijgen in de beperkingen van Mendix in praktijksituaties worden specifieke use cases onderzocht waar Mendix in zijn standaardvorm tekortschoot. Dit onderzoek begint met de identificatie van projecten waarin aanvullende modules of custom ontwikkeling noodzakelijk waren om aan de projectvereisten te voldoen. Vervolgens worden de onderliggende redenen geanalyseerd waarom de standaard Mendix-functionaliteit in deze gevallen niet toereikend bleek, waarbij zowel technische als functionele beperkingen worden gedocumenteerd. De impact van deze tekortkomingen op projectdoorlooptijd en budget wordt nauwkeurig beoordeeld om de werkelijke kosten van deze beperkingen te kwantificeren. Daarnaast wordt de effectiviteit van de geïmplementeerde aanvullende oplossingen geëvalueerd om te bepalen in hoeverre deze de oorspronkelijke beperkingen succesvol hebben

geadresseerd. Tot slot worden de gevonden scenario's gecategoriseerd om patronen te identificeren in situaties waarin Mendix-uitbreidingen typisch nodig zijn, wat waardevolle input vormt voor het te ontwikkelen beslissingskader.

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 vormt een essentieel onderdeel voor het beantwoorden van deelvraag 5 en 7. Deze analyse omvat diverse bronnen, waaronder officiële Mendix productgidsen die gedetailleerde technische specificaties en functionaliteiten beschrijven, casestudies van derden en rapporten uit de industrie die praktijkvoorbeelden en onafhankelijke evaluaties bieden, online forums en ontwikkelaarsgemeenschappen waar praktijkervaringen en knelpunten worden gedeeld, en academische publicaties over low-code ontwikkeling die theoretische onderbouwing en wetenschappelijke inzichten verschaffen over de bredere context van deze technologie.

Praktisch onderzoek

Voor de beantwoording van deelvraag 6 worden praktische tests uitgevoerd om de grenzen van het Mendix-platform te onderzoeken. Dit onderzoek richt zich op verschillende aspecten, waaronder schaalbaarheid en prestaties, waarbij de reactietijd, belastingstestresultaten en het vermogen om te schalen bij toenemende gebruikersaantallen worden geëvalueerd. Daarnaast worden de integratiemogelijkheden getest om de compatibiliteit met externe systemen, API's en databases te beoordelen. De aanpasbaarheid en uitbreidbaarheid van het platform worden geanalyseerd door te onderzoeken in hoeverre maatwerkfunctionaliteiten en uitbreidingen kunnen worden geïmplementeerd. Ook de ontwikkelingssnelheid wordt onder de loep genomen, waarbij wordt gekeken naar de efficiëntie van de ontwikkelomgeving en de snelheid waarmee applicaties kunnen worden gebouwd en aangepast. Tot slot wordt de beveiliging en compliance beoordeeld om vast te stellen in hoeverre het platform voldoet aan relevante regelgeving en beveiligingsstandaarden. Deze tests bieden een diepgaand inzicht in de sterke en zwakke punten van Mendix en helpen bij het bepalen van de geschiktheid van het platform voor specifieke toepassingen.

Reflectie op eigen ervaringen

In dit onderzoek wordt een gestructureerde reflectie uitgevoerd op de transitie van high-code naar low-code ontwikkeling binnen een lopend Mendix-project. Hierbij worden persoonlijke ervaringen gedocumenteerd, waarbij zowel de uitdagin-

gen als successen in de praktijk worden belicht. Er wordt een vergelijkende analyse gemaakt van de ontwikkelingsefficiëntie tussen traditionele high-code methoden en de Mendix-aanpak, met aandacht voor de verschillen in snelheid, flexibiliteit en onderhoudbaarheid. Daarnaast worden specifieke situaties geïdentificeerd waarin high-code kennis een toegevoegde waarde biedt binnen een Mendix-omgeving, bijvoorbeeld bij complexe logica of integraties met externe systemen. Tot slot wordt de leercurve geëvalueerd en worden de benodigde aanpassingen in denkwijze besproken die gepaard gaan met de overstap naar een low-code platform. Deze reflectie biedt waardevolle inzichten in de impact van low-code ontwikkeling op bestaande programmeerervaringen en werkwijzen.

Evaluatie van Mendix-uitbreidingen

In dit onderzoek wordt de effectiviteit van verschillende uitbreidingsstrategieën voor Mendix geanalyseerd. Een belangrijk aspect daarbij is de vergelijking tussen standaard Marketplace-modules en custom ontwikkeling, waarbij wordt gekeken naar factoren zoals flexibiliteit, implementatietijd en onderhoudsgemak. Daarnaast wordt de impact van Java-uitbreidingen onderzocht, met specifieke aandacht voor de invloed op onderhoudbaarheid en toekomstbestendigheid van het platform. Ook wordt een analyse uitgevoerd van de kosten-batenverhouding van verschillende uitbreidingsmethoden, waarbij zowel ontwikkel- als beheerkosten worden meegenomen. Tot slot worden de integratie-uitdagingen beoordeeld die zich voordoen bij het combineren van Mendix met externe systemen en services, om zo inzicht te krijgen in de haalbaarheid en complexiteit van verschillende uitbreidingsopties. Deze analyse draagt bij aan een beter begrip van de meest effectieve strategieën voor het uitbreiden van Mendix-toepassingen binnen verschillende bedrijfscontexten.

Ontwikkeling beslissingskader

Voor het beantwoorden van deelvraag 8, die zich richt op het destilleren van best practices en richtlijnen voor de keuze tussen Mendix en traditionele ontwikkeling, wordt een gestructureerd beslissingskader ontwikkeld. Dit kader ondersteunt projectmanagers en architecten bij het maken van weloverwogen keuzes over de inzet van Mendix in vergelijking met conventionele ontwikkelmethoden. Het biedt beslissingspunten voor de initiële keuze tussen Mendix en traditionele ontwikkeling en stelt criteria vast om te bepalen welke projectonderdelen beter geschikt zijn voor high-code ontwikkeling. Daarnaast worden richtlijnen geformuleerd voor het effectief combineren van low-code en high-code in hybride projecten, zodat beide benaderingen optimaal kunnen worden ingezet. Verder bevat het kader aanbevelingen voor de vroege identificatie van potentiële beperkingen en risico's, waardoor projectrisico's proactief kunnen worden gemitigeerd. Tot slot worden strategieën ontwikkeld om op basis van eerdere ervaringen en literatuur weloverwogen keuzes te maken bij toekomstige projecten. Dit beslissingskader draagt bij aan een sys-

tematische en onderbouwde aanpak voor het selecteren van de meest geschikte ontwikkelmethode binnen uiteenlopende projectcontexten.

4

Conclusie

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

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

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

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

A

Onderzoeksvoorstel

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

Samenvatting

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 beheren 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 onsamenhangende 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 leverancierssstamgegevens, 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 genereerde 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 probleemdomein (het ontbreken van een beslissingskader), gevolgd door onderzoek naar het oplossingsdomein (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

Hermans, M. P., & Mileff, P. (2023). Short introduction to Mendix. https://doi.org/https://doi.org/10.32968/psaie.2023.3.5

Hugo. (2024, februari 9). *Joget DX 7 Knowledge Base - Process Builder*. https://dev.joget.org/community/display/DX7/Process+Builder

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

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 Software Development* (pp. 183–196). Springer International Publishing. 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/

Mendix. (2025, februari 26). *Download Studio Pro*. https://marketplace.mendix.com/link/studiopro

Sido, N., & Emon, E. A. (2024, mei 31). *Low/No Code Development andGenerative 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

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