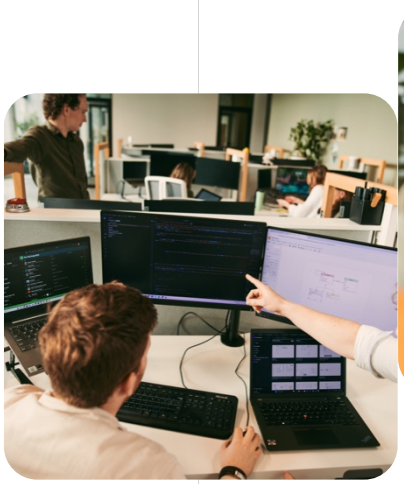
BEYONDER

PRO-CODE OF LOW-CODE



Keuzegids met inzichten van onze experts

INHOUDSOPGAVE

01.

INTRODUCTIE

In de snel veranderende wereld van softwareontwikkeling staan er voortdurend bedrijven voor de uitdaging om de meest geschikte ontwikkelingsstrategie te kiezen die aansluit bij hun specifieke behoeften en doelstellingen. Bij Beyonder, waar we gespecialiseerd zijn in zowel pro-code als low-code softwareontwikkeling, worden we vaak geconfronteerd met de vraag: "Welke aanpak is de beste keuze voor ons project?"

Het antwoord op deze vraag is niet altijd eenvoudig en hangt af van tal van factoren, waaronder de complexiteit van de applicatie, de beschikbare middelen, de gewenste doorlooptijd en de toekomstige schaalbaarheid. In deze whitepaper willen we de nuances van beide benaderingen verkennen en bedrijven helpen een weloverwogen beslissing te nemen.

Met Pro-Code bedoelen we het gebruik van programmeertalen zoals C#, Javascript, Python om software te bouwen. Met Low Code bedoelen we omgevingen als Mendix, OutSystems of MS Powerapps. Ook dit zijn omgevingen waarmee software ontwikkeld kan worden, echter de bouwblokken zijn 'kant-en-klaar' en de constructie is veelal visueel georiënteerd.

Pro-code ontwikkeling biedt flexibiliteit en controle, waardoor het ideaal is voor complexe, op maat gemaakte oplossingen. Aan de andere kant maakt low-code ontwikkeling het mogelijk om snel en kostenefficiënt applicaties te bouwen, zelfs met beperkte technische kennis, wat het aantrekkelijk maakt voor een breed scala aan projecten.

Door onze ervaring en expertise in beide domeinen, zijn we goed gepositioneerd om de voordelen en nadelen van pro-code en low-code te analyseren. We zullen ook verschillende use cases bespreken om te illustreren wanneer elke benadering de voorkeur verdient, en de belangrijkste overwegingen belichten die bedrijven moeten meenemen bij hun beslissing.

Of je nu een snel een Minimum Viable Product (MVP) wil lanceren of een complexe enterprise-oplossing nodig hebt, deze whitepaper biedt waardevolle inzichten en richtlijnen om je te helpen de juiste keuze te maken.

02.

ONTSTAAN VAN LOW-CODE

Low-code is ontstaan uit de behoefte aan snellere en efficiëntere manieren om software te ontwikkelen, in een tijd waarin traditionele ontwikkelingsmethoden vaak traag, kostbaar en foutgevoelig waren om aan de snel veranderende eisen van bedrijven te voldoen. In de jaren 2000 begonnen bedrijven als Mendix, OutSystems en Appian met het ontwikkelen van platforms die visuele, drag-and-drop interfaces boden voor applicatieontwikkeling, waardoor zowel technische als niet-technische gebruikers betrokken konden worden bij het

ontwikkelproces. Deze platforms maakten het mogelijk om zonder diepgaande programmeerkennis snel werkende applicaties te bouwen, te testen en te implementeren. De beweging werd verder versterkt door de toenemende vraag naar digitale transformatie, de noodzaak om de kloof tussen IT en zakelijke afdelingen te overbruggen, en de groeiende schaarste aan ervaren softwareontwikkelaars.

Hierdoor is low-code een aantrekkelijke oplossing geworden voor veel organisaties die wendbaarder en innovatiever willen zijn.

03.

WAT IS LOW-CODE?

Low-code ontwikkeling is een moderne benadering van softwareontwikkeling die gebruikmaakt van visuele interfaces, drag-and-drop tools, en minimale codering om snel en efficiënt applicaties te bouwen. Low-code platforms zijn ontworpen om de technische complexiteit te verminderen en de ontwikkeltijd te verkorten, waardoor zowel professionele ontwikkelaars als niet-technische gebruikers (zoals business analisten) in staat zijn om software te creëren.

3.1 | Kenmerken

Kenmerken van Low-Code Ontwikkeling:

- **Visuele ontwikkelomgeving:** Low-code platforms bieden een visuele ontwikkelomgeving waarin gebruikers applicaties kunnen bouwen door componenten te slepen en neer te zetten, in plaats van volledige code te schrijven. Dit maakt het proces intuïtiever en toegankelijker.
- **Snellere ontwikkelcycli:** Door gebruik te maken van herbruikbare componenten en vooraf gebouwde modules, kunnen ontwikkelaars snel prototypes en volledige applicaties bouwen. Dit versnelt de time-to-market aanzienlijk.
- **Minder technische vaardigheden vereist:** Low-code platforms zijn ontworpen om eenvoudig te gebruiken te zijn, zelfs voor personen zonder diepgaande programmeerkennis. Dit democratiseert de ontwikkelingsmogelijkheden binnen een organisatie.
- **Automatisering en Integratie:** Veel low-code platforms bieden ingebouwde mogelijkheden voor automatisering van workflows en integratie met andere systemen en databases, wat de ontwikkeling van end-to-end oplossingen vereenvoudigt.

3.2 | Use-cases

Low-code ontwikkeling wordt vaak gekozen voor projecten die:

- **Snelle prototyping en MVP:** Wanneer een organisatie snel een Minimum Viable Product (MVP) moet lanceren om een idee te testen of marktkansen te benutten. Beperkte baten en korte levensduur; een snelle oplossing van tijdelijke aard.

- **Business Process Automation:** Low-code leent zich voor het snel automatiseren van repetitieve bedrijfsprocessen en workflows. Processen die beschreven kunnen worden in een lineair model, zoals BPMN, zijn bij uitstek geschikt.
- **Applicaties met beperkte complexiteit:** Voor toepassingen die geen complexe business logica of maatwerk functionaliteiten vereisen, bijvoorbeeld een maatwerk HRMS, ERP of CRM systeem - feitelijk vaak applicaties die gedreven worden door recht toe, recht aan CRUD operaties.

Low-code ontwikkeling biedt een efficiënte en kosteneffectieve manier om applicaties te bouwen, vooral voor projecten met beperkte complexiteit en strakke deadlines. Het democratiseert softwareontwikkeling door, tot op zekere hoogte, het toegankelijk te maken voor een breder scala aan gebruikers, van professionele ontwikkelaars tot zakelijke gebruikers zonder diepgaande kennis van softwareontwikkeling.

04.

WAT IS PRO-CODE?

Pro-code is de meest voorkomende benadering van softwareontwikkeling waarbij ontwikkelaars gebruik maken van conventionele programmeertalen zoals C#, Java, Python en anderen om op maat gemaakte applicaties te bouwen. Deze methode staat in contrast met low-code platforms die visuele ontwikkeltools en minimale codering bieden om applicaties te creëren.

4.1 | Kenmerken

Kenmerken van Pro-Code:

- **Volledige controle en flexibiliteit:** Pro-code ontwikkeling biedt ontwikkelaars volledige controle over alle aspecten van de applicatie, van de gebruikersinterface tot de backend-logica en database-interacties. Dit maakt het mogelijk om zeer specifieke en op maat gemaakte functionaliteiten te implementeren die aansluiten bij de unieke behoeften van een organisatie, zonder concessies te hoeven doen.
- **Gebruik van traditionele programmeertalen:** Pro-code projecten worden geschreven in traditionele, tekstgebaseerde programmeertalen. Dit vereist diepgaande kennis van de gebruikte taal en de bijbehorende frameworks en libraries.
- **Flexibele architecturen:** Pro-code ontwikkeling maakt het mogelijk om flexibelere softwarearchitecturen te creëren, zoals microservices, gedistribueerde systemen, en applicaties met hoge prestaties en schaalbaarheid.
- **Robuuste integraties:** Pro-code biedt veel meer mogelijkheden om naadloze integraties te realiseren met andere systemen, databases, en externe services, wat essentieel is voor veel enterprise-oplossingen. Denk bijvoorbeeld aan integraties met een zorg hardware-systemen; zoals alarmbellen of valdetectie.

4.2 | Use-cases

Pro-code ontwikkeling wordt vaak gekozen voor projecten die:

- **Complexe functionaliteit:** applicaties die ingewikkelde bedrijfslogica, geavanceerde algoritmen, of speciale functies vereisen die niet eenvoudig te realiseren zijn met low-code platforms. Bijvoorbeeld planningsalgoritmen voor de kinderopvang of voorspellende algoritmiek in de palletindustrie.
- **Pro performance:** Systemen die optimale prestaties en schaalbaarheid nodig hebben, zoals real-time data processing, financiële transactiesystemen, of grote e-commerce platforms.
- **Strikte compliance en beveiliging:** Projecten waarbij strikte naleving van beveiligingsnormen en regelgeving essentieel is, zoals in de gezondheidszorg, financiële sector, of overheidsinstellingen. Hierbij kan het bijvoorbeeld gaan over scheiding van data per tenant, strikte encryptie vereisten met eigen sleutel, data backup requirements of uitgebreide audit trails voor impactvolle besluitprocessen.
- **Legacy-systemen:** Als een systeem moet werken met bestaande legacy-systemen en -architecturen, kan pro-code de flexibiliteit en compatibiliteit bieden met deze bestaande systemen. Bijvoorbeeld door maatwerk integraties op binaire interfaces te ondersteunen.

Pro-code ontwikkeling biedt de ultieme flexibiliteit en kracht voor het bouwen van op maat gemaakte applicaties. Voor organisaties die specifieke, niet-standaard oplossingen nodig hebben of complexe integraties moeten realiseren, blijft pro-code de voorkeurskeuze.

05.

TRENDS & ONTWIKKELINGEN

Het speelveld van pro- en low-code verandert voortdurend. De ontwikkelingen zorgen ervoor dat pro-code ontwikkeling sneller wordt en de low-code platforms nog meer flexibiliteit verkrijgen. Hieronder worden de belangrijkste ontwikkelingen uit elkaar gezet.

5.1 | Integratie platformen

Voor enterprise koppelingen met een hoge complexiteit is vaak maatwerk noodzakelijk. De opkomst van geavanceerde integratieplatformen heeft echter deze beperking aanzienlijk verminderd en de flexibiliteit van low-code ontwikkeling versterkt.

Integratieplatformen zoals Zapier, MuleSoft, en Microsoft Power Automate bieden krachtige mogelijkheden om verschillende systemen en diensten naadloos met elkaar te verbinden. Deze tools maken het mogelijk om data en functionaliteiten uit diverse bronnen te integreren zonder dat er diepgaande programmeerkennis nodig is. Door gebruik te maken van vooraf gebouwde connectors en visuele workflow-ontwerpers, kunnen pro- en low-code ontwikkelaars snel en eenvoudig integraties opzetten die voorheen alleen mogelijk waren met complexe, handmatig geschreven code.

Deze integratieplatformen vergroten de reikwijdte en mogelijkheden van low-code applicaties aanzienlijk. Ze stellen bedrijven in staat om bestaande systemen, zoals ERP's, CRM's, en andere enterprise-software, te verbinden met nieuwe low-code toepassingen, waardoor een naadloze gegevensstroom en procesautomatisering mogelijk wordt. Dit verhoogt niet alleen de operationele efficiëntie, maar maakt het ook eenvoudiger om snel in te spelen op veranderende zakelijke behoeften en technologische innovaties.

Door de inzet van integratieplatformen kunnen organisaties de voordelen van low-code ontwikkeling – snelheid, gebruiksgemak en lagere kosten – combineren met een verhoogde flexibiliteit en schaalbaarheid. Dit maakt low-code een nog krachtiger hulpmiddel voor het ontwikkelen van robuuste, geïntegreerde oplossingen die aan de complexiteit en dynamiek van moderne bedrijfsomgevingen voldoen.

5.2 | Low-code cloud development platforms

De ontwikkeling van low-code platforms blijft evolueren, met leveranciers die voortdurend streven naar het vergroten van het gebruiksgemak en de efficiëntie voor hun gebruikers. Een belangrijke trend hierbij is het aanbieden van alles-in-een oplossingen die volledig cloud-native zijn. Deze platforms draaien in de cloud van de low-code leverancier, waardoor ontwikkelaars snel en eenvoudig omgevingen kunnen uitrollen zonder uitgebreide kennis van cloudinfrastructuren te hoeven hebben. Dit verhoogt de productiviteit en stelt teams in staat om zich te concentreren op het bouwen van applicaties in plaats van het beheren van onderliggende infrastructuur.

Echter, deze beweging naar cloud-gebaseerde alles-in-een oplossingen brengt ook enkele belangrijke nadelen met zich mee, waaronder vendor lock-in en compliance-uitdagingen. Wanneer organisaties hun applicaties en gegevens volledig onderbrengen in de cloud van een specifieke low-code leverancier, kunnen ze afhankelijk worden van de technologieën en diensten van die leverancier. Dit kan de flexibiliteit beperken om in de toekomst over te stappen naar andere platforms of omgevingen, en kan leiden tot hogere kosten op de lange termijn als gevolg van licentie- en servicekosten die door de leverancier worden opgelegd. Voor pro-code oplossingen zijn er abstractielagen zoals Docker, Terraform en Kubernetes, daarnaast kan er gekozen worden voor services (zoals databases) die cloud-agnostisch zijn. Daarmee zijn pro-code oplossingen veel minder hard verbonden met de cloud omgeving waar ze op draaien.

Daarnaast kunnen er compliance-uitdagingen ontstaan doordat de low-code oplossing niet in de cloudomgeving van de organisatie zelf draait. Dit kan problematisch zijn voor bedrijven die moeten voldoen aan strikte regelgeving en data governance policies, zoals GDPR, NEN7510 of andere branche-specifieke normen. De controle over dataopslag, beveiliging en toegang kan beperkt zijn, wat het moeilijker maakt om aan de vereiste compliance-standaarden te voldoen. Door compliance, maar ook kosten, is er daarnaast een omslag gaande voor organisaties in diverse speelvelden, van public naar private cloud/on premise. De cloudplatform lock-in van de verschillende low-code platformen kan daarbij in de weg komen te staan.

5.3 | AI in pro-code ontwikkeling

De integratie van kunstmatige intelligentie (AI) in pro-code ontwikkeling heeft een aanzienlijke impact op de manier waarop software wordt ontworpen, ontwikkeld en beheerd. Hier zijn enkele belangrijke manieren waarop AI de pro-code ontwikkeling verandert:

- **Automatisering van codeerprocessen:** AI-gebaseerde tools zoals code-suggesties en automatische codegeneratie versnellen het ontwikkelingsproces door ontwikkelaars te helpen bij het schrijven van efficiëntere en foutvrije code. Dit vermindert de tijd en inspanning die nodig is voor handmatig coderen, waardoor projecten sneller kunnen worden afgerond.
- **Verbetering van debugging en testing:** AI-algoritmen kunnen patronen herkennen en voorspellingen doen over potentiële bugs en kwetsbaarheden in de code. Geavanceerde AI-tools kunnen automatisch testscripts genereren, bugs identificeren en zelfs suggesties geven voor oplossingen, wat de betrouwbaarheid en kwaliteit van de software verhoogt.
- **Optimalisatie van prestaties:** AI kan worden gebruikt om de prestaties van applicaties te monitoren en te optimaliseren. Door het analyseren van grote hoeveelheden gegevens kan AI knelpunten identificeren en aanbevelingen doen voor verbeteringen, zoals het optimaliseren van databasequery's of het verbeteren van de efficiëntie van algoritmen.
- **Intelligente documentatie en ondersteuning:** AI-gestuurde documentatietools kunnen automatisch technische documentatie genereren op basis van de geschreven code, wat tijd bespaart en de consistentie van documentatie verbetert. Daarnaast kunnen AI-chatbots en virtuele assistenten ontwikkelaars ondersteunen door directe antwoorden te geven op technische vragen en problemen.

De impact van AI op pro-code ontwikkeling is diepgaand en transformeert de manier waarop software wordt gemaakt. Door taken te automatiseren, de efficiëntie te verhogen en de kwaliteit van de code te verbeteren. Door het slim toepassen van AI in pro-code ontwikkeling, wordt het ontwikkelen van pro-code oplossingen steeds sneller.

5.4 | Pro-code builders

Tools zoals Visual Studio en Microsoft Frontpage legden in het verleden al de basis voor drag-and-drop interfaces binnen pro-code omgevingen, waarmee ontwikkelaars snel gebruikersinterfaces (UI) konden bouwen voor Microsoft programmeertalen.

Vandaag de dag zijn er nog meer geavanceerde drag-and-drop builders beschikbaar die pro-code als output genereren. Platforms zoals Webflow en FlutterFlow maken het mogelijk om visueel aantrekkelijke en functionele interfaces te ontwerpen met minimale handmatige codering, terwijl ze toch de flexibiliteit en kracht van pro-code behouden. Deze tools genereren efficiënte, onderhoudbare code die naadloos kan worden geïntegreerd in bestaande pro-code projecten. Daarnaast bieden oplossingen zoals Firebase en Hasura drop-in replacements voor API's, waardoor de ontwikkeling van back-end functionaliteiten sterk wordt versneld.

Bovendien zijn ontwerptools zoals Figma steeds beter in staat om front-end code te genereren, wat het ontwerpproces naadloos integreert met de ontwikkelfase.

Door deze tools slim in te zetten, kunnen ontwikkelaars het pro-code ontwikkelproces aanzienlijk versnellen. Dit vermindert het verschil in productiviteit tussen pro-code en low-code, terwijl de voordelen van maatwerk, controle en schaalbaarheid behouden blijven.

5.5 | Werven low-code talent

Low-code een relatief nieuwe technologie die snel aan populariteit wint. Hierdoor is de vraag naar gekwalificeerd low-code talent exponentieel gestegen, terwijl het aanbod nog achterblijft. Veel professionals hebben simpelweg nog niet de kans gehad om zich te specialiseren in low-code platforms.

Daarnaast vereist low-code ontwikkeling een unieke mix van vaardigheden. Het is niet alleen belangrijk om te begrijpen hoe de specifieke low-code tools werken, maar ook om over een sterke basis in softwareontwikkeling en bedrijfsprocessen te beschikken. Deze combinatie van technische en zakelijke kennis is zeldzaam en maakt het moeilijk om geschikte kandidaten te vinden.

In de praktijk zien we ook dat er maar een beperkte groep pro-code developers is, die zich om zou willen scholen naar low-code. De technologie is geen onderdeel van opleidingen, zoals informatica. De pro-code ontwikkelaars zien zich ook vaak als code-specialist; waar ze het gevoel hebben ‘hun ei’ minder kwijt te kunnen in een low-code platform.

06.

OVERWEGINGEN

De keuze voor pro- of low-code is niet zwart-wit. De hieronder uitgezette overwegingen worden zelden direct beantwoord door een organisatie. Het geeft stof tot nadenken en geeft inzichten voor de keuze pro-/low-code. Daarnaast zijn low- en pro-code te combineren, waardoor de kracht van beide opties benut wordt.

In de praktijk zien we vier richtingen:

1. **No-code:** ideaal voor het bouwen van eenvoudige applicaties of automatiseringstaken, zonder programmeerkennis. Het wordt vaak gebruikt door niet-technische gebruikers of bedrijfsanalisten om eenvoudige workflows, formulieren of gegevensbeheer oplossingen te creëren.
2. **Low-code:** is geschikt voor projecten waarbij er behoefte is aan meer maatwerk en complexiteit dan wat met no-code bereikt kan worden. Met de visuele ontwikkelingstools kunnen ontwikkelaars snel applicaties bouwen door gebruik te maken van vooraf gebouwde componenten en een beperkte hoeveelheid handmatige

codering. Dit is handig wanneer je een aangepaste logica of integratie met externe systemen nodig hebt, maar niet vanaf nul wilt beginnen.

3. **Pro-code:** is de traditionele aanpak waarbij ontwikkelaars volledige controle hebben over de code en het ontwikkelingsproces. Dit wordt vaak gebruikt voor complexe projecten waarbij maatwerkfunctionaliteit, geavanceerde logica, prestatieoptimalisatie of diepe integratie met bestaande systemen vereist is. Pro-code biedt de grootste flexibiliteit, maar kan meer tijd en gespecialiseerde programmeerkennis vereisen.
4. **Hybride (pro- & low-code):** door het combineren van low- en pro-code kan een oplossing neergezet worden die de flexibileit van pro-code en de snelheid van het bouwen van low-code benut. Denk bijvoorbeeld aan een administratief portaal in low-code, met een consumenten gerichte 'gelikte' front-end in pro-code of een plannings portaal in low-code, met een pro-code planningsalgoritme. Nadeel hiervan zijn uiteraard de benodigde expertises (low- en pro-code) en mogelijke impact op de running-costs.

NOTITIE:

No-code valt niet te vergelijken met low- of pro-code. Wij beschouwen no-code alleen als een oplossing voor relatief kleine punt-oplossingen waar geen bedrijfskritische informatie in wordt verwerkt. Denk aan digitale formulieren voor bijvoorbeeld HR processen, of eenvoudige externe intake processen.

6.1 | Intellectuele eigendomsrechten (IE)

Bij low-code valt een deel van de oplossing onder het intellectuele eigendom van het low-code platform. De rechten en controle over de gebruikte basis bouwblokken blijven bij de low-code leverancier, wat bedrijven afhankelijk maakt van de low-code leverancier. Je moet dus mee in de strategie van het platform naar keuze. Dit kan een beperking zijn en de waarde van het bedrijf verminderen, omdat een deel van de technologie niet volledig eigendom is van de organisatie.

Pro-code biedt volledige eigendom van de intellectuele eigendomsrechten. Bedrijven hebben volledige controle over hun broncode en kunnen de software onafhankelijk beheren en aanpassen. Dit versterkt de waarde van een bedrijf aanzienlijk, omdat het de vrijheid biedt om de technologie te commercialiseren en aan te passen zonder beperkingen. Bovendien kan het als waardevol bedrijfsactiva dienen, beschermd door patenten, auteursrechten en handelsmerken. Volledig eigendom van IE maakt een bedrijf aantrekkelijker voor investeerders en potentiële kopers, als bewijs van innovatie en strategische capaciteiten.

Bijvoorbeeld bij SaaS leveranciers spelen intellectuele eigendomsrechten een grote rol. Zeker in een tijd waarin veel traditionele bedrijven veranderen van fysieke diensten naar virtuele diensten en daardoor veel meer digitaal afhankelijk worden. Banken zijn daar een schoolvoorbeeld van.

6.2 | Vendor lock-in

Bij het gebruik van low-code platforms is de doorontwikkeling alleen mogelijk binnen de beperkingen en mogelijkheden van het gekozen platform. Dit betekent dat er een sterke afhankelijkheid is van de leverancier voor toekomstige updates, nieuwe functionaliteiten en ondersteuning. Deze afhankelijkheid kan problematisch zijn als de leverancier de ondersteuning beëindigt, licentieprijzen verhoogt of technologische beperkingen oplegt. Dit kan leiden tot onverwachte kosten en beperkte flexibiliteit, waardoor het aanpassingsvermogen van de softwareoplossingen wordt beperkt.

Pro-code biedt flexibiliteit in technologie en leverancier. Organisaties kunnen kiezen uit diverse programmeertalen, frameworks en tools die aansluiten bij hun specifieke behoeften. Deze onafhankelijkheid stelt hen in staat snel te reageren op marktveranderingen en technologische innovaties. Kritische noot hierbij is wel te blijven opletten bij de selectie van een pro-codetaal en framework; gevestigde talen als C# vormen met hun 24+ jarig bestaan, hun marktomvang en Microsoft als powerhouse weinig risico, maar voor de vele 'hippe, nieuwe frontend frameworks' kan dit heel anders lopen.

6.3 | Lifecycle

Een overweging is de verwachte levensduur van de software die ontwikkeld moet worden. Een korte levensduur spreekt voor een Low-Code aanpak gegeven de snelheid van ontwikkeling en dus het moment waarop de baten genomen kunnen worden. Software die langer gebruikt gaat worden zal uiteindelijk stabiliseren qua roadmap waarbij het kostenvoordeel qua werk afneemt en de licenties zwaarder gaan wegen.

Er dient een goede business case gemaakt te worden voor een juiste afweging, maar rond de 5 jaar is in veel gevallen een break-even punt als het gaat om Low-Code versus ontwikkeling met Pro-Code.

6.4 | Tijdkritische processen

Low-code biedt vaak minder controle en precisie. Dit kan problematisch zijn voor tijdkritische toepassingen zoals real-time data-analyse, alarmeringssystemen of beurs transacties, waar elke milliseconde telt. De beperkte controle kan leiden tot vertragingen en inconsistenties in de uitvoering van processen.

Pro-code biedt daarentegen volledige controle over de applicatie-architectuur. Ontwikkelaars kunnen optimalisaties en aanpassingen doorvoeren, wat de tijdigheid voor tijdkritische toepassingen kan garanderen. Pro-code biedt de robuustheid en precisie die nodig zijn voor nauwkeurige en betrouwbare procesuitvoering.

De regel is simpel, low-code is niet geschikt voor tijdkritische processen. Niet dat dit eenvoudig is met pro-code, maar in pro-code zijn er wel controlemechanisme en componenten beschikbaar voor een correcte en snelle tijdkritische realtime verwerking van gebeurtenissen.

6.5 | Multi-tenancy (SaaS)

Wanneer een organisatie overweegt hun software als dienst aan te bieden (ver-servicen, als SaaS), is ondersteuning voor multi-tenant architectuur cruciaal. Low-code platforms kunnen hierbij uitdagingen bieden vanwege de beperkte flexibiliteit van multi-tenant structuren. Bijvoorbeeld de gescheiden dataopslag per tenant, ondersteuning van verschillen in organisaties structuren van tenants of verschil in branding zijn niet out-of-the-box beschikbaar in low-code.

Pro-code ontwikkeling biedt volledige vrijheid en nauwkeurige controle bij het ontwerpen en implementeren van multi-tenancy structuren. Ontwikkelaars kunnen een maatwerk architectuur ontwikkelen die voldoet aan de specifieke vereisten van klanten, inclusief data-isolatie en beveiligingsmaatregelen.

6.6 | Budget

Een van de belangrijkste overwegingen bij het kiezen tussen low-code en pro-code ontwikkeling is de balans tussen initiële ontwikkelkosten en doorlopende operationele kosten. De productiviteitswinst is factor 2 tot 5, voor oplossingen die binnen de mogelijkheden van het low-code platform liggen. Hierbij dient opgemerkt te worden dat het dan vooral de netto-ontwikkeling effort betreft, want een deugdelijk ontwikkeltraject zowel pro- als low-Code kent ook een architectuur/analyse/UX component, coördinatie en investeringen in QA.

Echter, deze kostenvoordelen gaan gepaard met hogere doorlopende operationele kosten vanwege het licentiemodel dat door de low-code platforms wordt gehanteerd.

Als vuistregel kan gesteld worden dat Low-Code financieel niet interessant is voor een (eenvoudige) single-app strategie. Daarvoor zijn de licentiekosten veelal te hoog. De ROI voor low-code is sneller/eenvoudiger te bereiken als er meerdere oplossingen mee gebouwd worden. Pas dan biedt de productiviteitswinst voordelen.

6.7 | Time-to-market

In low-code platforms kan veel sneller worden ontwikkeld en geïmplementeerd, dan met pro-code. Dit versnelt niet alleen het ontwikkelproces, maar stelt organisaties ook in staat om snel prototypes te bouwen, feedback van gebruikers te verzamelen en iteratief verbeteringen door te voeren. Voor bedrijven die snel op marktveranderingen moeten reageren of binnen strakke deadlines nieuwe producten en diensten willen lanceren, biedt low-code een efficiënte en flexibele oplossing om concurrentievoordeel te behalen.

Een alternatieve strategie die (als het gaat om keuze voor een bepaalde architectuur) zou kunnen worden overwogen, is Low-code als innovatieplatform te zien. Als de innovatie succesvol blijkt te zijn en de specificaties (MVP+) grotendeels helder zijn, de oplossing alsnog door Pro-code te vervangen.

6.8 | User base

Veel low-code platformen hanteren een prijsmodel waarbij de kosten stijgen naarmate meer gebruikers toegang krijgen tot de applicatie. Soms wordt hierbij nog onderscheid gemaakt

tussen interne en externe gebruikers. Dit kan betekenen dat de licentiekosten snel toenemen naarmate de gebruikersbasis groeit. Aan de andere kant heeft het aantal actieve gebruikers ook bij een pro-code invloed op de infrastructuurkosten, zoals servercapaciteit en netwerkverkeer.

Pro-code is aantrekkelijk voor applicaties die een groot aantal gebruikers moeten ondersteunen, omdat de kosten niet lineair toenemen met de groei van de gebruikersbasis. In plaats daarvan kunnen organisaties hun infrastructuur schaalbaar maken om aan de vraag te voldoen, wat vaak kostenefficiënter kan zijn op lange termijn. Deze flexibiliteit kan aanzienlijke besparingen opleveren en biedt een duidelijk voordeel voor bedrijven met een groot of groeiend aantal gebruikers.

Het advies is dus bij het maken van keuzes rekening te houden met een potentiële user base na enkele jaren en niet alleen te focussen op bijvoorbeeld de eerste (MVP) releases.

6.9 | Gebruikerservaring (UX)

Dankzij standaard componenten en templates in het low-code platform wordt een consistente en functionele gebruikerservaring geborgd. Dit beperkt echter wel de maatwerk mogelijkheden en creativiteit. Zo resulteert low-code in minder unieke user interfaces, die minder aansluiting hebben bij specifieke branding en gebruikers verwachtingen. Grootste beperkingen zitten bijvoorbeeld bij apps in het creëren van een native feel met soepele animaties en overgangen van pagina's. De meeste low-code platforms resulteren in meer website-achtige interacties met gelimiteerde mogelijkheden voor functional feedback voor de gebruiker. Daarnaast zijn de design systemen bij low-code oplossingen vaak niet dynamisch inzetbaar genoeg om aan alle moderne toegankelijkheids eisen (WCAG 2.1+) te voldoen.

Pro-code ontwikkeling biedt daarentegen volledige vrijheid om elk aspect van de interface aan te passen, van geavanceerde animaties tot op maat gemaakte lay-outs, waardoor een unieke en sterk merkgerichte gebruikerservaring kan worden gecreëerd. Er zijn legio aan off-the-shelf component libraries die gebaseerd zijn op de hoogste accessibility eisen. Dit vormt een eenvoudige basis voor een solide en modern design systeem. Dit verhoogt de gebruikers betrokkenheid en tevredenheid door een onderscheidende, goed aansluitende en esthetisch aantrekkelijke gebruikerservaring te bieden.

6.10 | Veranderlijke bedrijfsprocessen

In een dynamische omgeving waar bedrijfsprocessen voortdurend evolueren of beïnvloed worden van buitenaf, biedt low-code aanzienlijk meer flexibiliteit. Low-code platforms zijn ontworpen om snelle aanpassingen en iteraties mogelijk te maken. Hierdoor kunnen organisaties snel inspelen op veranderende marktvoorwaarden, klantbehoeften of interne procesverbeteringen zonder langdurige ontwikkelcycli.

Hoewel aanpassingen in pro-code meer tijd in beslag kunnen nemen en uitgevoerd moeten worden door specialistische ontwikkelaars, zijn wijzigingen beter te controleren in het ontwikkel- en testproces. Binnen pro-code bestaat al jaren tooling om bijvoorbeeld versiebeheer op broncode uit te voeren (bijvoorbeeld Git). Doordat deze tooling onderdeel is

van het ontwikkelproces, zijn developers in staat om code te reviewen, af te splitsen en changes vast te leggen (om ze later te herleiden). Hiermee zorgen pro-code developers ervoor dat aanpassingen correct en betrouwbaar zijn. Voor organisaties die prioriteit geven aan robuustheid en betrouwbaarheid, biedt pro-code de zekerheid dat elke wijziging grondig is gevalideerd voordat deze in productie wordt genomen.

6.11 | Complexiteit

Low-code platforms zijn uitstekend geschikt voor het snel en efficiënt bouwen van applicaties met een relatief eenvoudige tot beperkte mate van complexiteit in logica. Ze bieden visuele ontwikkeltools en herbruikbare componenten die het mogelijk maken om veelvoorkomende bedrijfsprocessen en workflows eenvoudig te modelleren. Echter, wanneer applicaties complexere logica vereisen, zoals geavanceerde algoritmen, gedetailleerde gegevensverwerking of gespecialiseerde integraties, kunnen de beperkingen van low-code platforms naar voren komen. In dergelijke gevallen moeten ontwikkelaars vaak terugvallen op maatwerkcode of zelfs geheel uitwijken naar pro-code oplossingen om de benodigde functionaliteit te realiseren. Dit kan leiden tot hybride scenario's waar zowel low-code als pro-code wordt ingezet, maar het kan ook de ontwikkelings- en onderhouds complexiteit verhogen.

Er is geen heldere definitie van ‘complexiteit’ van software. Vaak wordt dit pas geconstateerd als de symptomen zich aandienen. Denk aan het terugvallen van de productiviteit van de ontwikkelaar omdat het overzicht kwijt raakt. Of performance problemen door de omvang van software. Of - wellicht de belangrijkste - de noodzaak logica buiten Low-code in aparte services te positioneren omdat de ‘grafische structuren’ tekortschieten om bijvoorbeeld complexe acties op dataverzamelingen door te voeren. Kortom het inschatten van dergelijke aspecten vooraf is belangrijk alvorens de technologie keuze wordt gemaakt.

6.12 | Compliance en traceability

Low-code platforms kunnen beperkingen opwerpen op het gebied van compliance en traceability. Hoewel ze ingebouwde beveiligings- en compliance-functies bevatten, voldoen deze mogelijk niet aan de specifieke eisen van bepaalde industrieën zoals gezondheidszorg, financiën of overheid. Beperkingen in data governance, infrastructuur controle en audit trails kunnen het moeilijk maken om aan strikte regelgeving te voldoen. Bijvoorbeeld het ontsluiten van trace logs of het aansluiten van een Security Operations Center is niet in alle gevallen mogelijk.

Pro-code biedt daarentegen volledige controle over de architectuur en implementatie van compliance- en traceability-maatregelen. Ontwikkelaars kunnen aangepaste oplossingen creëren die voldoen aan specifieke eisen en regelgeving zoals GDPR en NEN7510. Dit omvat gedetailleerde data governance policies, uitgebreide logging en monitoring, en robuuste beveiligingsmaatregelen. Hierdoor kunnen organisaties die strikte naleving van regelgeving en volledige traceerbaarheid vereisen, effectief aan deze behoeften voldoen.

07.

CONCLUSIE

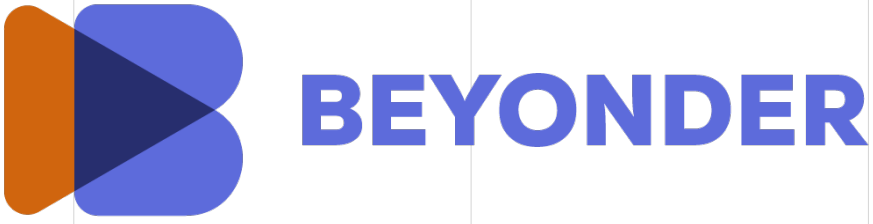
De keuze tussen low-code en high-code is niet altijd evident en ligt genuanceerd. Om een goede overweging te maken, spelen er veel aspecten mee, die tegen elkaar afgewogen moeten worden. De toekomst maakt de overwegingen ook gecompliceerder, de productiviteit van pro-code neemt toe, de beperkingen van low-code nemen af. Daarnaast worden we geconfronteerd met tegengestelde overwegingen; tussen compliance, snelheid, levensduur van de oplossing en complexiteit.

Doordat low-code steeds meer mainstream zal worden, evolueren bestaande low-code platformen en komen er nieuwe, misschien wel specialistischere, platformen bij. De technologie zal zich blijven verbeteren met meer geavanceerde functionaliteiten, betere integraties en verhoogde gebruiksvriendelijkheid, waardoor zowel technische als niet-technische gebruikers in staat zullen zijn om snel applicaties te ontwikkelen en aan te passen. Bovendien zal de focus op governance en beveiliging toenemen om te voldoen aan de groeiende eisen van compliance en data privacy.

High-code ontwikkeling zal zich blijven richten op complexe, maatwerkoplossingen die specifieke functionaliteiten en schaalbaarheid vereisen, wat moeilijk te realiseren is met low-code platforms. Terwijl low-code een groter deel van routine- en minder complexe applicatie ontwikkeling zal overnemen, blijft high-code essentieel voor kritieke systemen, geavanceerde integraties en toepassingen die hoge prestaties en nauwkeurige controle vereisen. Daarnaast zal de ontwikkelsnelheid met de toepassing van Gen AI in het ontwikkelproces, pro-code ontwikkeling versnellen. Ook zullen high-code ontwikkelaars steeds meer samenwerken met low-code platforms om hybride oplossingen te creëren, waarbij de snelheid en flexibiliteit van low-code worden gecombineerd met de robuustheid en diepgang van high-code.
Whyellow zet in op zowel low- als pro-code. We kunnen deze expertises voor iedere use-case een juiste afweging maken tussen low- en pro-code. Wij denken graag met je mee. Kom een kopje koffie drinken met een van onze experts.

BIJLAGE: KEUZEMATRIX

	OVERWEGINGEN VOOR LOW-CODE	OVERWEGING VOOR PRO-CODE
Belang van intellectueel eigendom	Deel van de technologische oplossing valt onder het IE van het low-code platform.	Volledig eigendom van de IE rechten.
Onafhankelijkheid van technologie/leverancier	Doorontwikkeling alleen mogelijk gebruik makende van het low-code platform.	Flexibiliteit in technologie en leverancier.
Levensduur oplossing	Langere levensduur maakt licentiemodel ongunstig.	Kortere levensduur maakt ontwikkelkosten buiten proportie.
Time-to-market	Low-code heeft een snelle doorloop, is sneller te realiseren.	Ontwikkel doorlooptijd van pro-code is vaak 2-3 keer de tijd van low-code.
Budget	Lage kosten voor ontwikkeling, hogere running costs door licentiemodel.	Hoge kosten voor ontwikkeling, lagere running costs.
Aantal actieve gebruikers	Aantal actieve gebruikers hebben direct invloed op licentie kosten.	Aantal actieve gebruikers heeft enkel invloed op infrastructuur kosten.
Streven om te 'ver-servicen' (als dienst buiten organisatie)	Multi-tenant complexiteit is moeilijker onder te brengen in low-code.	Volledige vrijheid in multi-tenancy structuren en complexiteit.
Veranderlijkheid bedrijfsprocessen	Veranderingen zijn snel door te voeren, eventueel door business zelf.	Veranderingen moeten altijd via het ontwikkelteam, maar daardoor wel altijd gevalideerd, door een controleproces.
Gebruikerservaring (UX)	Beperkingen door componentenset beschikbaar in platform.	Volledige vrijheid en daardoor altijd passend bij de branding van het bedrijf.
Complexiteit van logica	Complexiteit kan maken dat er deels uitgeweken moet worden naar pro-code.	Geschikt voor complexe algoritmische toepassingen.
Compliance en traceability	Beperkingen in maatregelen op basis van platform, zoals data governance, eigen infrastructuur en voldoen aan regulering.	Vrijheid in inrichting, waardoor maatregelen beter te borgen zijn.
Technisch tijdkritische processen	Controle over tijdigheid en volgordelijkheid van afhandeling ontbreekt. Daardoor minder geschikt voor oplossingen met tijdsafhankelijkheid.	Ondanks de additionele complexiteit in architectuur en algoritmiek, is controle te verkrijgen over de tijdigheid binnen een pro-code oplossing.


BEYONDER