Programma Technologie

Yappa Knowledge Hub

Stagerapport van

Sarikaya Mustafa

Kandidaat voor de graad van Professionele bachelor in

ICT

Stagementor(en):

Bedrijfsmentor: Davy Dewit

Mentor UCLL: Mimi Willems

Academiejaar: 2025 - 2026

Voorwoord

Deze bachelorproef vormt het schriftelijke verslag van mijn stage bij Yappa. Tijdens deze stage kreeg ik de kans om een intern softwareproject van de eerste probleemverkenning tot een werkbaar Minimum Viable Product uit te bouwen. Het project kreeg de naam Yappa Knowledge Hub, of kortweg YapHub. Het doel was om kennis die bij Yappa verspreid raakt over Slack en andere bronnen beter te bewaren, te verwerken en opnieuw beschikbaar te maken voor de juiste medewerkers.

In de eerste plaats gaat mijn dank uit naar mijn bedrijfspromotor, de heer Davy Dewit, voor de intensieve begeleiding tijdens het project. Zijn code reviews waren inhoudelijk streng, maar daardoor bijzonder waardevol. Door de vele opmerkingen over architectuur, naamgeving, testbaarheid en onderhoudbaarheid leerde ik beter begrijpen wat professionele codekwaliteit in een bedrijfscontext betekent.

Daarnaast gaat mijn waardering uit naar de heer Wesley Lancel. Als hub lead binnen Flow speelde hij een belangrijke rol in het mogelijk maken van deze stage en in het opzetten van de technische context waarin ik dit project kon uitwerken. Zijn vertrouwen en tijd maakten het mogelijk om het project niet alleen als schoolopdracht, maar ook als een realistische interne toepassing voor Yappa te benaderen.

Mevrouw Laura Schrijvers, mijn projectmanager tijdens de stage, verdient eveneens een woord van dank. Zij volgde het project mee op, dacht mee wanneer ik nood had aan overleg en beantwoordde mijn vragen wanneer ik functionele of organisatorische keuzes moest aftoetsen. Die momenten van consultatie hielpen mij om mijn technische keuzes beter te verbinden met de noden van gebruikers en projectwerking.

Een hartelijke dank je wel richt ik tot mevrouw Mimi Willems voor de begeleiding vanuit UCLL en voor de opvolging van mijn stageproces. Ook de collega's van Yappa, in het bijzonder het Flow-team, wil ik bedanken voor de open werksfeer, de feedback tijdens overlegmomenten en de kans om mee te kijken naar de manier waarop een digitaal agentschap projecten organiseert. Ten slotte bedank ik mijn familie, medestudenten en vrienden voor hun steun tijdens deze stageperiode.

Abstract

Yappa is een full-service digitaal agentschap gevestigd te Bilzen dat hoogwaardige digitale oplossingen ontwikkelt voor externe klanten. Intern hecht Yappa veel waarde aan kennisdeling en innovatie binnen hun multidisciplinaire teams, georganiseerd in gespecialiseerde hubs. De stageopdracht bestond uit het ontwerpen en ontwikkelen van de “YapHub”: een centraal, AI-gedreven platform voor kennisbeheer.

Het doel van dit platform is om verspreide informatie uit diverse kanalen, zoals slackberichten en externe URL’s, te centraliseren en te categoriseren. Deze informatie wordt vervolgens via rolgerichte AI-samenvattingen en rapportages direct beschikbaar gemaakt binnen de specifieke teamomgevingen van de Yappanezen.

Het project volgt een incrementele aanpak (Proof of Concept gevolgd door een Minimum Viable Product) en introduceert een bewust architecturaal onderscheid: Slack fungeert als het primaire invoerkanaal, terwijl het “Orakel” (de centrale Notionworkspace van Yappa) dienstdoet als het consumptieplatform. De backend, gebouwd met Symfony 7 en Doctrine ORM (een database abstraction layer), stuurt de Notion API aan om kennisitems, gegenereerde AI-samenvattingen en periodieke rapportages automatisch als Notion-pagina’s te integreren.

De Slack-ingestiemodule is ontwikkeld met TypeScript en de Bolt SDK. OpenAI-modellen verzorgen hierbij de tekstuele adaptatie. Het resultaat is een robuuste en schaalbare infrastructuur die medewerkers behoedt voor informatie-overload door relevante kennis naadloos binnen hun vertrouwde Notion teampagina’s te presenteren.

Inhoudsopgave

Verklarende woordenlijst

Term	Verklaring
AI	Artificiele intelligentie. In dit project wordt AI gebruikt om kennisitems kort samen te vatten.
API	Application Programming Interface. Een technische koppeling waarmee systemen gegevens uitwisselen.
Bolt SDK	Softwarepakket van Slack om interactieve Slack-apps te bouwen.
CI/CD	Continuous Integration en Continuous Deployment. Automatische controles en uitrolstappen in het ontwikkelproces.
Digest	Een periodiek overzicht van meerdere kennisitems binnen een thematische lijst.
Doctrine ORM	Object-relational mapper voor PHP waarmee databasegegevens als objecten worden beheerd.
DTO	Data Transfer Object. Een object dat gegevens gestructureerd doorgeeft tussen lagen van de applicatie.
E2E-test	End-to-endtest. Een test die een volledige gebruikersflow controleert.
Ephemeral bericht	Slack-bericht dat alleen zichtbaar is voor de gebruiker die de actie uitvoert.
Knowledge item	Een opgeslagen stuk kennis, bijvoorbeeld een Slack-bericht, tekst of URL.
MVP	Minimum Viable Product. De kleinste bruikbare versie van een product die echte waarde levert.
Notion	Samenwerkingsplatform voor documentatie, databases en kennisbeheer.
POC	Proof of Concept. Een beperkte proefopstelling om technische en functionele aannames te valideren.
Slack App Home	Startscherm van een Slack-app waarin knoppen, lijsten en interactieve acties kunnen worden getoond.
Socket Mode	Slack-verbindingsmethode waarbij de app via een uitgaande WebSocket-verbinding events ontvangt.
SSRF	Server-Side Request Forgery. Een beveiligingsrisico waarbij een server ongewenste externe of interne URL’s opvraagt.
Vite Press	Static site generator waarmee de projectdocumentatie als navigeerbare website werd gepubliceerd
Yappanezen	Informele benaming voor de medewerkers van Yappa.

1 Inleiding

1.1 Situering van de stageopdracht

Deze bachelorproef beschrijft mijn stageproject bij Yappa, een digitaal agentschap waar verschillende teams dagelijks werken aan websites, applicaties, online marketing en digitale strategie. Binnen zo'n omgeving ontstaat voortdurend nieuwe kennis: technische artikels, oplossingen voor terugkerende problemen, inzichten uit klantenprojecten, updates over tools en interne afspraken.

Die kennis wordt vaak gedeeld op het moment dat ze nuttig is. Een developer deelt bijvoorbeeld een artikel in Slack, een marketeer stuurt een update over een platformwijziging door of een projectmanager merkt tijdens een overleg een belangrijk inzicht op. Het probleem is niet dat er te weinig kennis gedeeld wordt. Het probleem is dat die kennis verspreid staat over meerdere kanalen en daardoor moeilijk opnieuw te vinden is.

Mijn opdracht bestond erin om een systeem te ontwerpen en te bouwen dat die kloof verkleint. Het project kreeg de naam Yappa Knowledge Hub, intern ook YapHub. De kern van de oplossing is eenvoudig: medewerkers moeten kennis kunnen toevoegen op de plaats waar ze al werken, namelijk Slack, terwijl de opgeslagen kennis duurzaam en gestructureerd terechtkomt in Notion, de bestaande kennisomgeving van Yappa.

1.2 Probleemstelling

Yappa gebruikt Slack voor snelle communicatie en Notion voor documentatie en kennisopslag. Beide tools hebben een duidelijke meerwaarde, maar ze lossen samen niet automatisch het probleem van kennisbeheer op. Slack is sterk in directe samenwerking, maar minder geschikt als langetermijnarchief. Notion is sterk in structuur en documentatie, maar vraagt meer bewuste invoer van de gebruiker.

Daardoor ontstaat een praktische spanning. Waardevolle informatie wordt vaak wel gedeeld, maar niet consequent bewaard. Een link die vandaag nuttig is, verdwijnt morgen tussen andere berichten. Een medewerker die later naar hetzelfde artikel zoekt, weet vaak niet meer in welk kanaal het gedeeld werd of welke woorden toen gebruikt werden. Bovendien heeft niet elk team dezelfde informatiebehoefte. Een technische analyse kan voor een developer essentieel zijn, maar voor een projectmanager vooral samengevatte impactinformatie bevatten.

De centrale probleemstelling van dit project luidt daarom:

Hoe kan Yappa interne kennis die verspreid raakt via dagelijkse communicatie eenvoudig capteren, automatisch verwerken en opnieuw beschikbaar maken via een gestructureerde kennisomgeving?

1.3 Doelstelling en onderzoeksvragen

Het hoofddoel van de stageopdracht was het realiseren van een werkbaar intern kennisdeelplatform. Dat platform moest niet alleen aantonen dat de technische koppelingen mogelijk zijn, maar ook een bruikbare workflow aanbieden voor medewerkers.

De opdracht werd vertaald naar vier deelvragen. Die zijn bewust abstracter geformuleerd dan de uiteindelijke technische oplossing, omdat bij de start van de stage nog niet vastlag dat Slack, Notion en AI de definitieve bouwstenen zouden worden.

- Hoe kan interne kennis op een eenvoudige manier worden verzameld zonder de bestaande werkgewoontes van medewerkers te verstoren?
- Hoe kan verzamelde kennis gestructureerd worden bewaard zodat ze later opnieuw vindbaar en bruikbaar is?
- Hoe kan de verzamelde kennis verwerkt worden tot een korter, relevanter overzicht voor verschillende medewerkers of teams?
- Hoe kan het systeem later uitbreidbaar blijven naar andere invoer- en uitvoerkanalen, zoals e-mail, Gmail, RSS-feeds of andere interne tools?

Tijdens de POC- en MVP-fase kregen deze vragen een concreet antwoord: Slack werd het invoerkanaal, Notion werd de kennisomgeving, de Symfony-backend verzorgde verwerking en synchronisatie, en Slack DM's werden gebruikt om digests actief te verspreiden. Deze vragen vormen de rode draad doorheen dit verslag. In het besluit worden ze opnieuw opgenomen en beantwoord op basis van het gerealiseerde MVP.

1.4 Aanpak en afbakening

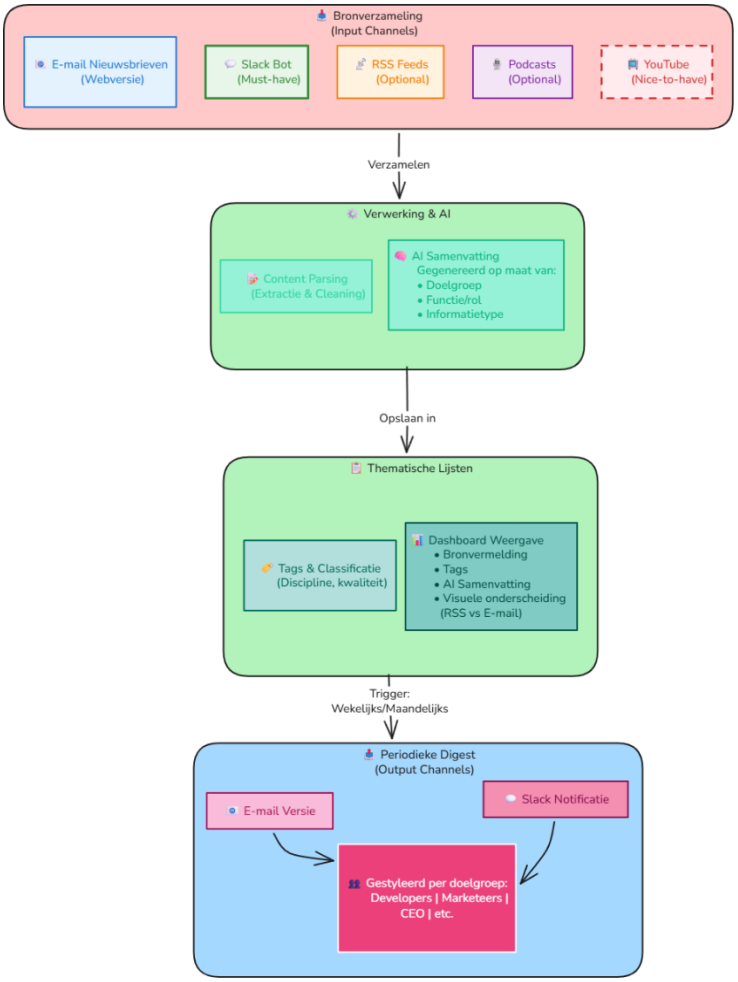
Het project werd iteratief aangepakt. In de eerste fase werd een Proof of Concept gebouwd. Een POC is geen afgewerkt product, maar een beperkte proefopstelling waarmee de belangrijkste aannames worden getest. In dit geval ging het niet alleen om de vraag of Slack, Symfony, Notion en AI technisch konden samenwerken, maar ook om de vraag welke vorm van kennisdeling voor Yappa-medewerkers werkelijk bruikbaar zou zijn.

Voor die POC-fase werkte ik eerst verschillende denkpistes uit. Ik maakte flowcharts, brainstormschema's en architectuurschetsen voor mogelijke oplossingen. Daarbij onderzocht ik onder meer of een dashboard in Twig zinvol was, of Craft CMS kon aansluiten bij de bestaande Yappa-expertise, of een monolithische C#-oplossing eenvoudiger zou zijn, en of een agentachtige aanpak met OpenClaw de intentie van gebruikers automatisch kon herkennen en kennis zelfstandig kon categoriseren. Uiteindelijk koos ik voor PHP en Symfony voor de backend, omdat die technologie het best aansloot bij de ervaring en standaarden binnen Yappa.

In totaal maakte ik tijdens de verkenning ongeveer vijftig flowcharts: onder meer voor Slack-flows, Symfony-flows, Notion-flows, dataflows, input-process-outputscenario's en OpenClaw-brainstorming. Die schema's hielpen mij om de mogelijkheden te ordenen en te vermijden dat ik te snel een technische richting koos. Een selectie daarvan is in dit verslag verwerkt als Mermaid-flowchart.

Na die validatie verschoof de focus naar een Minimum Viable Product. Een MVP is de kleinste versie van een product die echte gebruikerswaarde levert. Voor YapHub betekende dit dat de oplossing niet alle denkbare functies moest bevatten, maar wel de volledige kernflow moest ondersteunen: kennis toevoegen, inhoud verwerken, opslaan, samenvatten, bundelen en verspreiden.

Een POC had ook een tweede functie: de haalbaarheid van technologieën zichtbaar maken voordat er veel ontwikkeltijd in één richting werd geïnvesteerd. In de eerste twee weken onderzocht ik daarom meerdere mogelijke vormen. Een oplossing kon bijvoorbeeld bestaan uit een Slack-dashboard, een Twig-dashboard in Symfony, een Craft CMS-omgeving omdat Yappa daar veel ervaring mee heeft, of een meer monolithische applicatie in een andere stack. Door die opties eerst te schetsen in flowcharts en kleine technische proeven, kon ik sneller aftoetsen welke richting het best paste bij Yappa.



Figuur 1 POC-brainstorm volgens het input-process-outputmodel.

De uiteindelijke POC werd opgebouwd rond het model input, processing en output. Dat model maakte het mogelijk om ideeën te vergelijken zonder meteen vast te zitten aan één technologie. Input ging over de manieren waarop kennis kon binnenkomen. Processing ging over verrijking, extractie, opslag en samenvatting. Output ging over hoe kennis terug bij medewerkers terechtkomt. Pas na die voorbereiding werd Slack gekozen als praktische invoerlaag, Symfony als backend en Notion als kennisomgeving.

De scope werd bewust beperkt. Rolgebaseerde samenvattingen, geavanceerde zoekfilters, PDF-verwerking, een apart webdashboard, automatische verwerking van RSS-feeds, YouTube-video's, podcasts en uitgebreid promptbeheer werden onderzocht of overwogen, maar niet als gerealiseerde MVP-functionaliteit opgenomen. In de huidige MVP is er bijvoorbeeld één vaste prompt voor Nederlandstalige samenvattingen en geen PDF-extractie. Die keuze was belangrijk om het project binnen de stageperiode kwalitatief af te werken.

2 Bedrijfscontext en analyse

In dit hoofdstuk wordt eerst de organisatie voorgesteld waarin de stageopdracht plaatsvond. Daarna volgt een analyse van de bestaande kennisstroom, de betrokken stakeholders en de functionele en niet-functionele verwachtingen. Dit kader is belangrijk om de keuzes uit de volgende hoofdstukken te begrijpen, omdat zij in grote mate werden gestuurd door de manier waarop Yappa intern werkt.

2.1 Voorstelling van Yappa

Yappa is een full-service digitaal agentschap gevestigd in Bilzen. Het bedrijf ontwikkelt digitale oplossingen voor klanten en combineert daarbij verschillende disciplines: softwareontwikkeling, online marketing, branding, support en projectbegeleiding. Intern noemen medewerkers zichzelf Yappanezen. Die benaming past bij de informele en betrokken bedrijfscultuur die ik tijdens mijn stage heb ervaren.

De organisatie werkt met gespecialiseerde hubs. Hub Flow is het ontwikkelteam en vormde de directe context van mijn stage. Daarnaast zijn er onder meer teams voor online marketing, branding en support. Elk team gebruikt kennis op een andere manier. Developers zoeken vaak technische details, architecturale keuzes of codevoorbeelden. Marketeers hebben eerder nood aan platformupdates, trends en klantgerichte inzichten. Projectmanagers willen vooral snel begrijpen welke informatie impact heeft op planning, scope of klantencommunicatie.

Yappa heeft al een sterke cultuur van kennisdeling. Een voorbeeld daarvan is YappaWaDoeDa, een intern moment waarop medewerkers delen wat ze geleerd hebben. Ook in Slackkanalen wordt veel kennis uitgewisseld. Vooral het ontwikkelkanaal is belangrijk voor dit project: developers delen er nieuwe technische inzichten, links naar artikels, updates over frameworks en oplossingen die tijdens projecten ontstaan. Dat maakte het dev-kanaal een concrete en herkenbare toepassing voor YapHub. Die cultuur vormde een belangrijk uitgangspunt voor de stageopdracht: het project moest bestaande gewoontes ondersteunen in plaats van medewerkers naar een volledig nieuwe werkwijze te dwingen.

2.2 De bestaande kennisstroom

Voor de start van het project liep de interne kennisstroom vooral via twee platformen. Slack werd gebruikt voor dagelijkse communicatie, snelle vragen, links en korte updates. Notion werd gebruikt voor meer structurele documentatie. De centrale Notion-omgeving wordt binnen Yappa het Orakel genoemd.

Die combinatie is logisch, maar niet vanzelf voldoende. Slack verlaagt de drempel om iets te delen, maar verhoogt tegelijk het risico dat kennis vluchtig blijft. Notion biedt structuur, maar vraagt dat iemand bewust de tijd neemt om informatie te documenteren. In de praktijk gebeurt die tweede stap niet altijd, zeker niet wanneer het gaat om kleine inzichten of gedeelde links.

YapHub werd ontworpen als brug tussen beide werelden. Slack blijft het invoerkanaal omdat medewerkers daar al actief zijn. Notion wordt de plaats waar kennis duurzaam wordt bewaard en opnieuw geraadpleegd. De backend verbindt beide systemen en zorgt voor verwerking, synchronisatie en rapportering.

2.3 Stakeholders en gebruikersprofielen

De belangrijkste stakeholders zijn de medewerkers van Yappa die kennis toevoegen of raadplegen. Binnen dit project werden drie representatieve gebruikersprofielen gebruikt om de oplossing concreet te maken.

Een eerste profiel is de developer uit Hub Flow. Deze gebruiker deelt technische artikels, security-updates of frameworkdocumentatie in Slack. Voor deze gebruiker is het belangrijk dat kennis snel kan worden opgeslagen zonder de ontwikkelflow te onderbreken. De meerwaarde van YapHub ligt hier in snelle captatie, technische vindbaarheid en een samenvatting die helpt om de relevantie van een artikel in te schatten.

Een tweede profiel is de online marketeer. Deze gebruiker volgt wijzigingen in zoekmachines, advertentieplatformen en sociale media. De informatie is vaak extern en tijdsgevoelig. Voor dit profiel is het nuttig dat een URL automatisch wordt verwerkt en dat de essentie in het Nederlands beschikbaar wordt gemaakt, zonder dat elk lang artikel volledig moet worden gelezen.

Een derde profiel is de projectmanager of teamlead. Deze gebruiker heeft nood aan overzicht. Niet elk detail is relevant, maar het is wel belangrijk om te weten welke thema's leven binnen de organisatie en welke informatie impact kan hebben op projecten. Digests, of periodieke overzichtsrapporten, zijn voor deze doelgroep belangrijk omdat ze meerdere kennisitems bundelen tot een leesbaar overzicht.

2.4 Functionele en niet-functionele verwachtingen

Uit de analyse volgden functionele verwachtingen. Medewerkers moesten kennis kunnen toevoegen via een Slack Message Shortcut, een Global Shortcut of een actie in de Slack App Home. Kennisitems moesten aan thematische lijsten gekoppeld kunnen worden. Een thematische lijst is voor gebruikers een herkenbare verzameling rond een onderwerp, team of interessegebied waarop men zich kan abonneren. URL's moesten automatisch worden uitgelezen zodat de AI-samenvatting meer context kreeg dan alleen een titel. Verder moesten kennisitems, thematische lijsten, abonnementen en digests gesynchroniseerd worden met Notion.

Naast functionele eisen waren ook niet-functionele verwachtingen belangrijk. De oplossing moest onderhoudbaar zijn, omdat ze na de stage verder begrepen en eventueel uitgebreid moet kunnen worden door Yappa. Ook de gebruikerservaring speelde een rol. Rechtstreekse Notion API-aanroepen konden merkbare wachttijd veroorzaken. Daarom werd lokale opslag toegevoegd, zodat Slack snel feedback kan geven en Notion daarna als duurzame kennisomgeving wordt bijgewerkt. Ten slotte moest de oplossing veilig omgaan met externe URL's, API-sleutels en integraties.

3 Van Proof of Concept naar Minimum Viable Product

Dit hoofdstuk beschrijft hoe het project in twee fases werd aangepakt. Eerst werd een Proof of Concept gebouwd om de haalbaarheid van het idee en de gekozen technologieën te valideren. Daarna werd dat prototype stapsgewijs uitgewerkt tot een Minimum Viable Product met een volledige werkflow. Op het einde van het hoofdstuk volgt een overzicht van de scopekeuzes die tijdens de stage werden gemaakt.

3.1 Waarom eerst een Proof of Concept

De eerste weken van de stage stonden in het teken van verkenning en validatie. De oorspronkelijke opdracht was nog breed geformuleerd: er moest een manier komen om kennis te delen, te bewaren en opnieuw beschikbaar te maken. Op dat moment stond nog niet vast dat Slack de invoerlaag zou worden of dat Notion de beste plaats was om kennis te raadplegen. Daarom was een Proof of Concept noodzakelijk.

Een POC toont niet alleen of een idee functioneel nuttig is, maar ook of een technologie haalbaar is. In deze fase kon ik bijvoorbeeld onderzoeken of een Slack-dashboard voldoende mogelijkheden bood, of een Twig- of Craft CMS-dashboard beter bij Yappa paste, of een aparte monolithische applicatie zinvol was, en hoe zulke keuzes de gebruikerservaring zouden beïnvloeden. Zo konden concepten besproken worden voor er veel ontwikkeltijd in een definitieve richting ging.

Daarnaast gebruikte ik de POC om feedback te verzamelen. Sommige medewerkers wilden vooral personalisatie per groep, bijvoorbeeld voor developers, marketeers of HR. Anderen wilden kunnen inschatten hoe belangrijk een thematische lijst voor hun eigen werk was, zodat ze eerst het meest relevante nieuws konden lezen. Die feedback maakte duidelijk dat het project niet alleen een opslagplaats mocht worden, maar een systeem dat kennis helpt beoordelen, filteren en opnieuw activeren.

De POC had dus drie doelen: technische haalbaarheid testen, de gebruikersnoden beter begrijpen en de scope afbakenen. Pas daarna werd de oplossing concreter: Slack bleek geschikt als plaats waar kennis binnenkomt, Notion bleek logisch als bestaande kennisomgeving, en Symfony paste bij de technische standaarden van Yappa.

Die fase had ook een belangrijke leerwaarde. Een POC mag snel gebouwd worden, maar de code mag niet zomaar doorgroeien tot productiekwaliteit zonder herwerking. Tijdens de code reviews werd duidelijk waar de initiële oplossing te veel technische schuld bevatte: te grote pull requests, te veel hardcoded waarden, te weinig afbakening tussen verantwoordelijkheden en te complexe abstracties op plaatsen waar een eenvoudiger ontwerp beter paste.

3.2 Evolutie naar een MVP

Na de POC werd het project stapsgewijs omgevormd tot een Minimum Viable Product. Het verschil tussen beide fases is belangrijk. De POC bewees dat de kernflow mogelijk was. Het MVP moest die flow betrouwbaar, onderhoudbaar en bruikbaar maken voor de praktijk.

De MVP bevatte uiteindelijk een volledige keten. Een medewerker kan kennis toevoegen via Slack. Wanneer een URL aanwezig is, haalt de backend de relevante pagina-inhoud op. Het kennisitem wordt opgeslagen in SQLite en gesynchroniseerd naar Notion. Thematische lijsten organiseren de kennis voor gebruikers. Gebruikers kunnen zich abonneren op zulke lijsten. Op basis van kennisitems binnen een periode kan de applicatie een digest genereren. Daarbij worden AI-samenvattingen gemaakt of hergebruikt. Het resultaat wordt opgeslagen in Notion en kan via Slack Direct Message naar abonnees worden gestuurd.

Deze evolutie vroeg niet alleen nieuwe functies, maar ook herstructurering. De backend werd opgesplitst in controllers, orchestrators, domeinservices, repositories en integratieservices. De Slack-code werd modulairer opgebouwd, zodat interactieve acties, schermen en modals beter onderhoudbaar werden. De documentatie werd parallel bijgewerkt, zodat het project overdraagbaar bleef.

Tijdens deze fase werkte ik met verschillende repositorycontexten. Mijn eigen monorepo bevatte documentatie, tests, Slack-code en Symfony-code samen, zodat ik het volledige project kon opvolgen en documenteren. Aan Yappa-zijde bestonden vooral aparte Slack- en Symfony-repositories. De documentatie en testopzet hield ik daarom deels afzonderlijk bij en ik opende hiervoor ook een PR, zodat Yappa die onderdelen later kon overnemen indien gewenst.

3.3 Scopekeuzes tijdens de stage

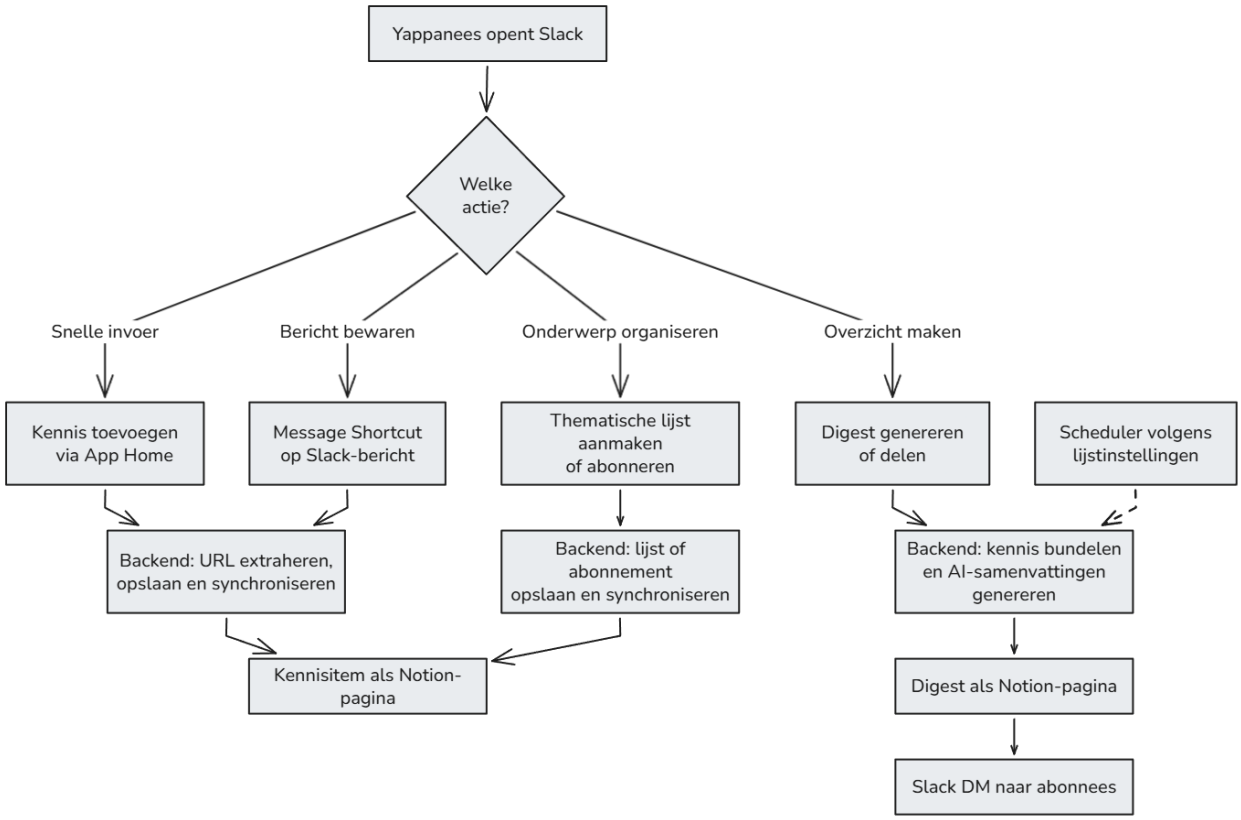
Een belangrijk onderdeel van professioneel ontwikkelen is niet alleen bepalen wat gebouwd wordt, maar ook bepalen wat bewust niet gebouwd wordt. Tijdens de stage zijn verschillende functies besproken die nuttig kunnen zijn, maar niet in de MVP zijn opgenomen.

Voorbeelden daarvan zijn meerdere AI-samenvattingen per doelgroep, een volledig promptbeheersysteem, een aparte webinterface dashboard, geavanceerde zoekfilters, PDF- en YouTube-verwerking en een volledige gebruikersauthenticatie. Die functies kunnen later waardevol zijn, maar ze zouden tijdens de stage te veel complexiteit toegevoegd hebben.

De gekozen MVP-scope focuste daarom op de waardevolste basis: kennis snel capteren, inhoud verrijken, bewaren in Notion, samenvatten met AI en gericht verspreiden via digests. Daardoor bleef het project haalbaar en kon er meer aandacht gaan naar codekwaliteit, tests en documentatie.
Naast de broncode werd ook een VitePress-documentatiesite opgebouwd en gepubliceerd op Cloudflare Pages: knowledgehub-aks.pages.dev. Die website bundelt de projectdocumentatie, MVP-presentatie, architectuuroverzichten, workflowdocumenten, testdocumentatie, weekly reports en deze scriptie. Voor de lezer van dit verslag is dat belangrijk, omdat niet elke technische tabel, demo-opname of uitgebreide flowchart volledig in de hoofdtekst past. De website fungeert daarom als raadpleegbare documentatiebasis bij het verslag.

4 Functionele realisatie

Dit hoofdstuk beschrijft de gerealiseerde werking vanuit het perspectief van de gebruiker. De technische uitwerking volgt in hoofdstuk 5; hier ligt de nadruk op wat een medewerker daadwerkelijk doet en ziet. De volledige flow start bij het toevoegen van kennis in Slack en eindigt bij een raadpleegbaar kennisitem in Notion of een digest die actief naar abonnees wordt verspreid via Slack. Onderstaand schema bundelt de belangrijkste gebruikersacties, de verwerking in de backend en de uitvoer richting Notion en Slack:



Figuur 2: High-level functionele flow van YapHub van gebruikersactie tot Notion-pagina en Slack-distributie

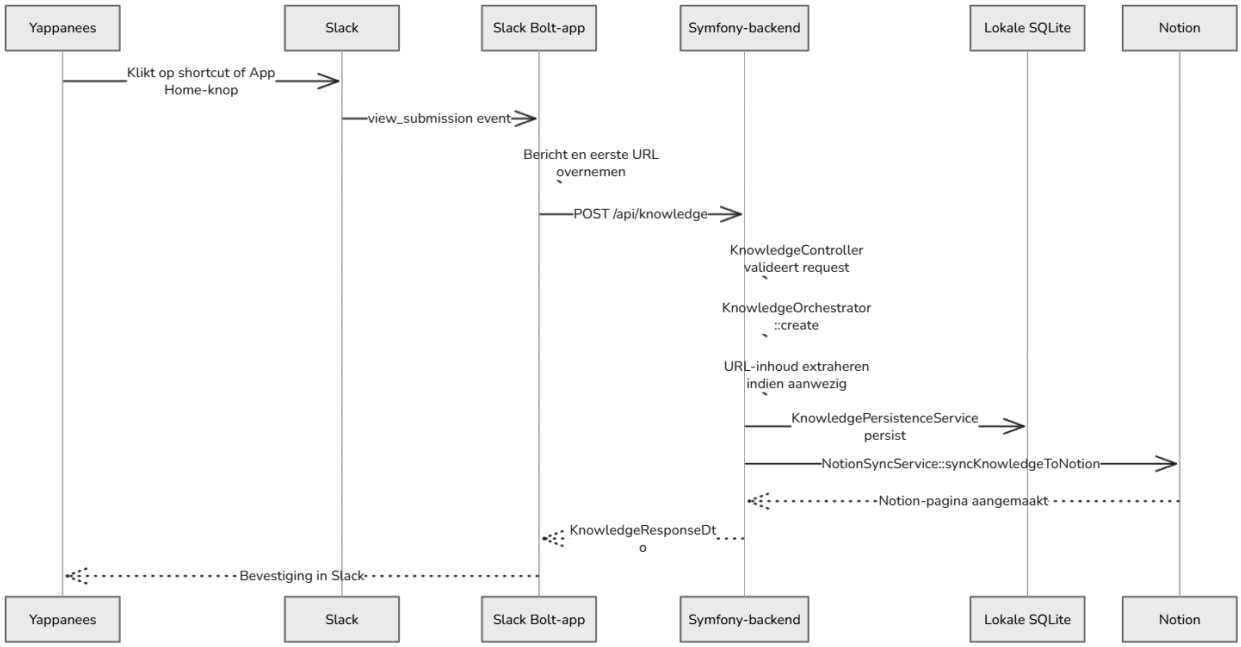
4.1 Kennis toevoegen via Slack

Slack is het primaire invoerkanaal van YapHub. Die keuze is bewust gemaakt omdat medewerkers Slack al dagelijks gebruiken. Een nieuw systeem heeft meer kans op adoptie wanneer het aansluit bij bestaande gewoontes.

De applicatie ondersteunt meerdere manieren om kennis toe te voegen. Via een Message Shortcut kan een gebruiker een bestaand Slack-bericht bewaren als kennisitem. Via een Global Shortcut kan een gebruiker vanop eender welke plaats in Slack snel een link of tekst toevoegen. Daarnaast biedt de Slack App Home knoppen om acties te starten vanuit een centraal dashboard.

Bij het toevoegen van kennis opent Slack een modal. Daarin kan de gebruiker informatie aanvullen, zoals titel, inhoud, URL, tags en thematische lijst. Het systeem probeert zoveel mogelijk context over te nemen uit het oorspronkelijke bericht of uit de ingevoerde URL. Daardoor blijft de invoer eenvoudig en wordt de kans groter dat medewerkers de functie effectief gebruiken.

Onderstaand sequentiediagram toont de feitelijke interactie tussen Slack, de backend en Notion bij het toevoegen van een kennisitem:



Figuur 3: Sequentiediagram van het toevoegen van een kennisitem via Slack, met de echte controller- en servicenamen uit de backend.

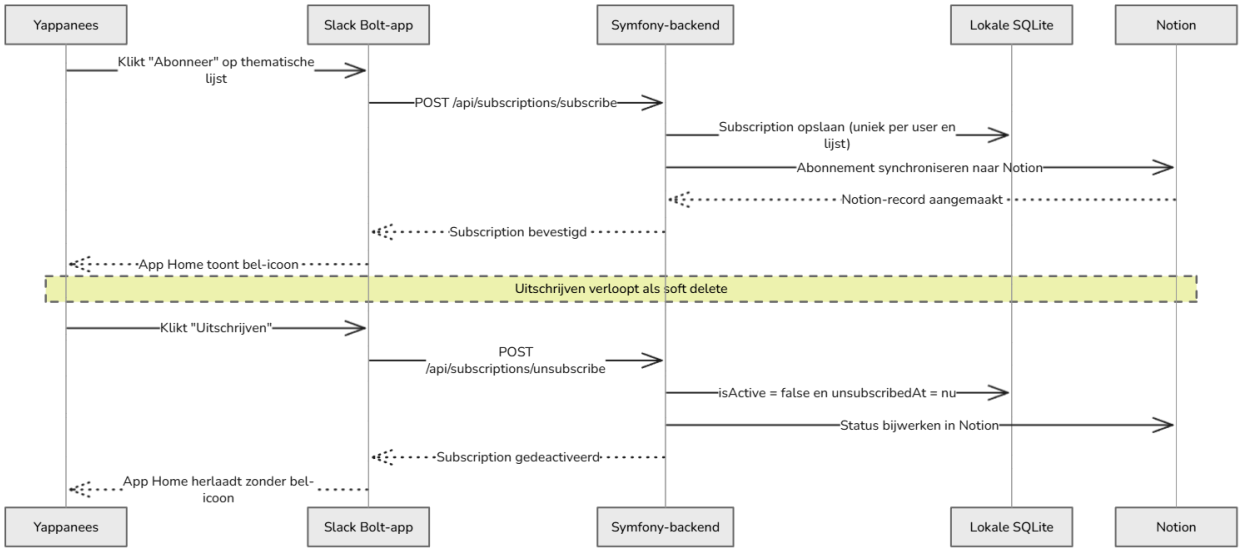
4.2 Thematische lijsten en abonnementen

Kennis heeft pas waarde wanneer ze geordend kan worden. Daarom werkt YapHub voor gebruikers met thematische lijsten. Een thematische lijst groepeert kennis rond een onderwerp, team of interessegebied, bijvoorbeeld development, marketing of projectmanagement. Elke lijst kan een naam, beschrijving, icoon en digestinstellingen hebben. In de code wordt dit model intern `Category` genoemd, maar in de tekst van dit verslag wordt de gebruikersterm thematische lijst gebruikt.

Naast thematische lijsten ondersteunt het MVP ook abonnementen. Een abonnement betekent dat een gebruiker aangeeft interesse te hebben in een bepaalde lijst. Wanneer later een digest voor die lijst wordt verspreid, kan het systeem automatisch bepalen welke gebruikers het overzicht moeten ontvangen.

Deze aanpak voorkomt dat iedereen alle informatie ontvangt. Kennisdeling wordt daardoor gerichter. Een medewerker kiest zelf welke lijsten relevant zijn. Dat is belangrijk omdat het oorspronkelijke probleem niet alleen kennisverlies was, maar ook information overload.

Onderstaand sequentiediagram toont hoe een gebruiker zich abonneert of uitschrijft op een thematische lijst en hoe deze actie wordt opgeslagen en gesynchroniseerd:



Figuur 4: Sequentiediagram van het abonneren en uitschrijven op een thematische lijst

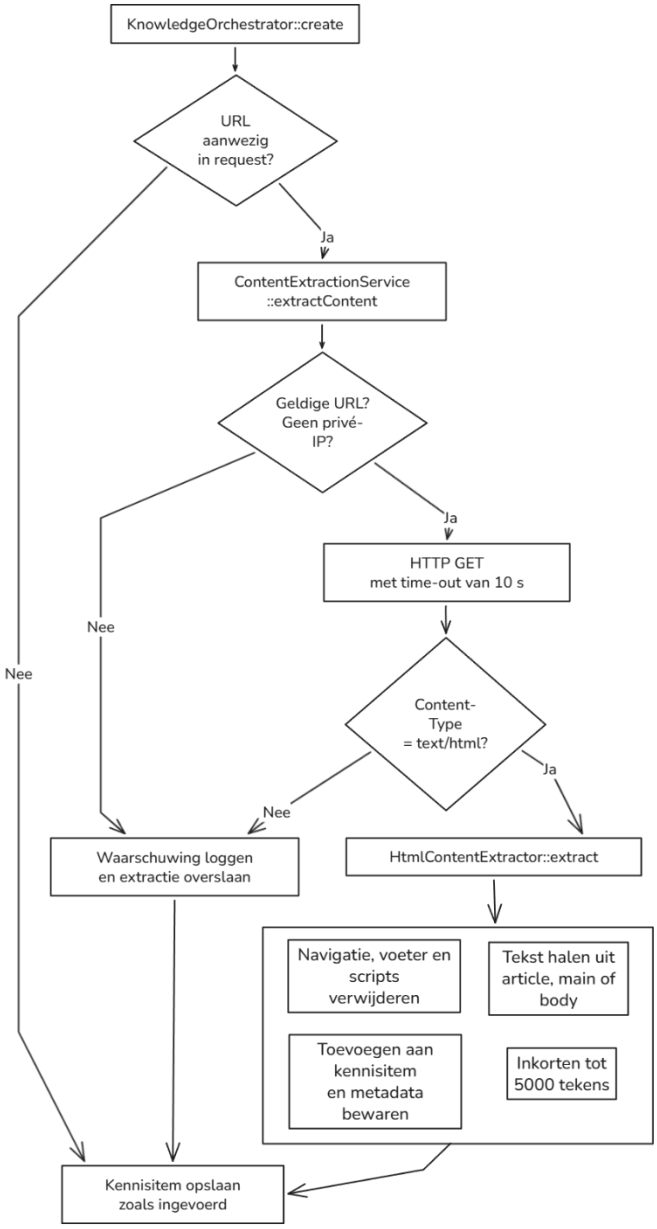
4.3 URL-inhoudsextractie

Veel kennisitems bestaan uit externe links. Alleen de URL of titel bewaren is vaak onvoldoende, want de inhoud van een webpagina kan langer en informatiever zijn dan de korte beschrijving in Slack. Daarom bevat de backend een URL-inhoudsextractie.

Wanneer een gebruiker een kennisitem met URL toevoegt, probeert de backend de HTML-pagina op te halen en de hoofdinhoud te extraheren. Navigatie, advertenties en voeters worden zoveel mogelijk weggefilterd. De relevante tekst wordt toegevoegd aan het kennisitem en later gebruikt als input voor de AI-samenvatting.

Deze functie verhoogt de kwaliteit van de samenvattingen. De AI krijgt niet alleen een titel of korte beschrijving, maar meer inhoudelijke context. Tegelijk moest deze functie veilig worden uitgewerkt. Externe URL's kunnen risico's vormen, bijvoorbeeld Server-Side Request Forgery. Daarom werd aandacht besteed aan URL-validatie, time-outs en foutafhandeling.

Het beslissingsschema aan de rechterkant toont welke controles de backend uitvoert voor een URL effectief wordt opgehaald en welke filters de extractie veilig en bruikbaar houden:



Figuur 5: Beslissingsschema voor URL-inhoudsextractie, inclusief SSRF-controle, content-type-check en truncatie

4.4 Notion als kennisomgeving

Notion is binnen Yappa al een vertrouwde omgeving voor documentatie. Daarom werd Notion niet vervangen door YapHub, maar net gebruikt als duurzame kennislaag. De applicatie synchroniseert kennisitems, thematische lijsten, digests en abonnementen met Notion-databases.

Het voordeel van deze keuze is dat medewerkers kennis kunnen blijven raadplegen in een omgeving die ze al kennen. Een opgeslagen kennisitem wordt een Notion-pagina met eigenschappen zoals titel, bron-URL, tags, status en relaties naar thematische lijsten. Digests worden eveneens als Notion-pagina's bewaard, zodat ze later opnieuw geraadpleegd kunnen worden.

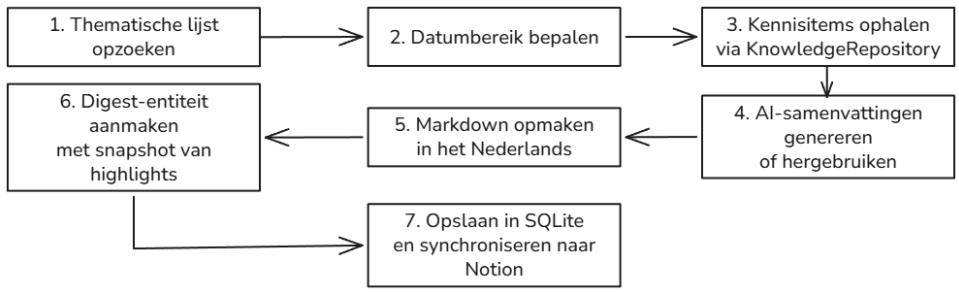
Notion fungeert in dit project dus niet alleen als externe integratie, maar als onderdeel van de productvisie. Slack is de snelle ingang, de backend verwerkt en bewaakt de flow, en Notion is het archief en de leesomgeving.

4.5 AI-samenvattingen en digests

AI wordt in YapHub gebruikt om informatie sneller bruikbaar te maken. Het systeem genereert korte Nederlandstalige samenvattingen van kennisitems. In de MVP gebeurt dat via een vaste prompt en een AI-provider die kan werken met OpenAI of OpenRouter.

De samenvattingen worden lazy gegenereerd. Dat betekent dat de AI niet onmiddellijk wordt aangeroepen wanneer een kennisitem wordt opgeslagen. Die keuze houdt de Slack-respons snel. De samenvatting wordt pas gemaakt wanneer ze nodig is, bijvoorbeeld tijdens het genereren van een digest. Als er al een samenvatting bestaat, kan die opnieuw gebruikt worden.

Een digest is een periodiek rapport over kennisitems binnen een thematische lijst en tijdsperiode. Het systeem verzamelt de relevante items, genereert of hergebruikt samenvattingen, formatteert het resultaat als Nederlandstalige tekst en bewaart het rapport. Daardoor hoeft een medewerker niet elk kennisitem afzonderlijk te openen om de grote lijnen te begrijpen.
Onderstaande pipeline vat de zeven stappen samen die de backend doorloopt om een digest aan te maken:



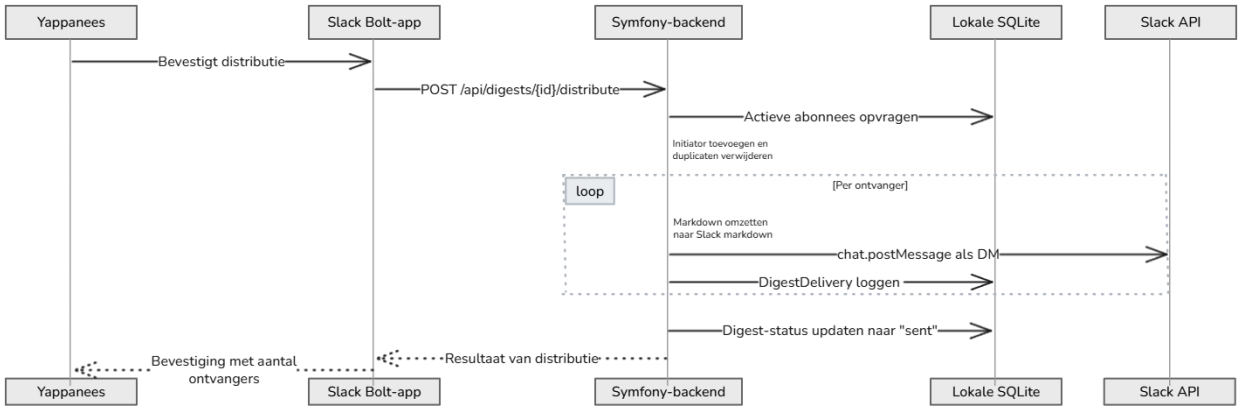
Figuur 6: Pipeline van zeven stappen voor digestgeneratie, van lijstresolutie tot Notion-synchronisatie.

4.6 Distributie en planning

Naast het genereren van digests ondersteunt het MVP ook distributie. Wanneer een digest klaar is, kan het systeem bepalen welke gebruikers op de thematische lijst geabonneerd zijn. Die gebruikers ontvangen het rapport via Slack Direct Message. Per ontvanger wordt de leveringsstatus bijgehouden, zodat mislukte leveringen later opnieuw geprobeerd kunnen worden.

Daarnaast bevat het systeem digestinstellingen per thematische lijst. Een lijst kan bijvoorbeeld een wekelijkse, tweewekelijkse, maandelijkse of driemaandelijkse digest hebben. Een scheduler kan controleren welke lijsten aan de beurt zijn en vervolgens de digestgeneratie en distributie starten.

Deze functies maken van YapHub meer dan een opslagplaats. Het systeem brengt kennis actief terug naar medewerkers op het moment dat die relevant is.
Onderstaand sequentiediagram toont hoe de backend de abonnees verzamelt, de digest per ontvanger via Slack DM verstuurt en de leveringsstatus per gebruiker bijhoudt:



Figuur 7: Sequentiediagram van de distributie van een digest naar abonnees, met DigestDelivery-logging per ontvanger

5 Technische realisatie

In dit hoofdstuk wordt de technische uitwerking van YapHub besproken. Eerst volgt een overzicht van de algemene architectuur. Daarna worden de Symfony-backend, de Slack-applicatie in TypeScript, de Notion-synchronisatie, de AI-integratie en de kwaliteitsbewaking afzonderlijk toegelicht. Per onderdeel wordt verwezen naar concrete klassen en bestanden uit de codebase, en wordt een flowchart toegevoegd waar dat de werking verduidelijkt. De flowcharts zijn bewust compact gehouden zodat ze leesbaar blijven op een A4-pagina; de volledige klasse- en bestandsstructuur is terug te vinden in de bijhorende repository en in de technische documentatie van het project.

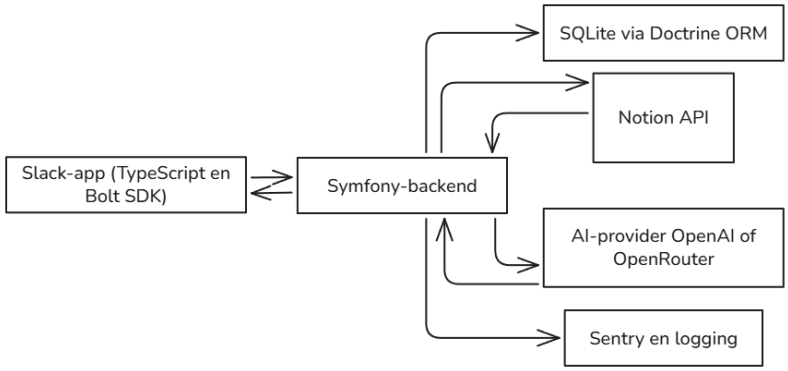
5.1 Algemene architectuur

De technische architectuur van YapHub is gelaagd opgebouwd. Die keuze werd gemaakt om verantwoordelijkheden duidelijk te scheiden en de code onderhoudbaar te houden. Een Slack-interactie mag bijvoorbeeld niet rechtstreeks alle businesslogica bevatten. Een controller mag input ontvangen, maar moet complexe workflows delegeren.

De hoofdlijn van de architectuur is als volgt:

- Slack vormt de gebruikersinterface.
- De Symfony-backend ontvangt API-aanvragen en coordineert workflows.
- Doctrine ORM bewaart lokale data in SQLite.
- Notion bewaart en presenteert kennis op lange termijn.
- De AI-provider genereert samenvattingen wanneer dat nodig is.

In de documentatie werd deze opbouw verder uitgewerkt als een gelaagde architectuur met duidelijke verantwoordelijkheden voor presentatie, orchestration, domeinlogica, persistence, externe integraties, logging en AI-infrastructuur. Het doel daarvan was niet om complexiteit toe te voegen, maar om te voorkomen dat alle logica in controllers of Slack-handlers terechtkomt.



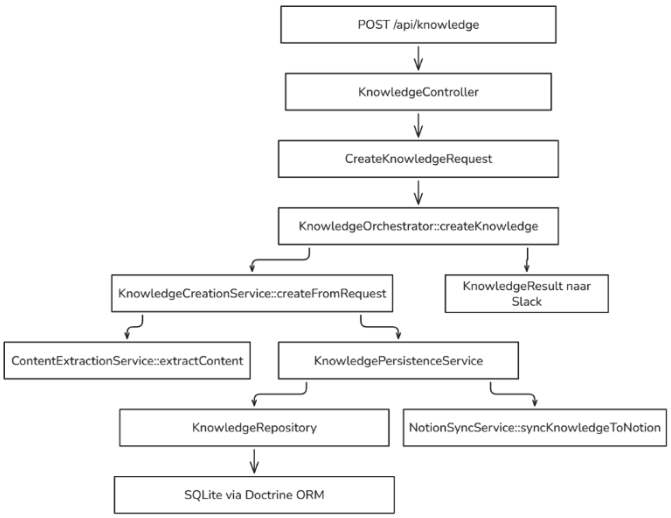
Figuur 8: Technische hoofdarchitectuur van YapHub.

5.2 Symfony-backend

De backend is gebouwd met Symfony 7 en PHP. Symfony werd gekozen omdat het aansluit bij de technische omgeving van Yappa en een volwassen basis biedt voor dependency injection, routing, configuratie, console commands en testing.

De backend bevat meerdere domeinen. Het domein Knowledge beheert kennisitems. `Category` is de interne codenaam voor thematische lijsten. Digest beheert rapporten en de bijbehorende generatieflow. Subscription beheert abonnementen. Notion-services verzorgen de synchronisatie met Notion. AI-services abstraheren de communicatie met externe AI-providers.

Een belangrijk patroon in de backend is het orchestrator pattern. Een orchestrator coordineert een volledige workflow. Bij het aanmaken van een kennisitem kan zo'n workflow bestaan uit validatie, inhoudsextractie, opslag, synchronisatie naar Notion en foutafhandeling. Door die stappen in een orchestrator te plaatsen, blijven controllers dun en blijft de businesslogica testbaar.



Figuur 9: Backend-flow van request tot resultaat.

Voor databasebeheer wordt Doctrine ORM gebruikt. Entiteiten zoals `Knowledge`, `Category`, `Digest`, `DigestDelivery` en `Subscription` worden als PHP-objecten gemodelleerd en door Doctrine naar SQLite gemapt. Voor tijdstempels wordt Gedmo Timestampable gebruikt, zodat `createdAt` en `updatedAt` niet telkens handmatig bijgewerkt moeten worden.

Codefragmenten kunnen in een technische bachelorproef nuttig zijn, maar alleen wanneer ze een ontwerpkeuze verduidelijken. Grote hoeveelheden code zouden de leesbaarheid verminderen en horen eerder thuis in de repository of in een bijlage. Daarom gebruikt dit verslag vooral flowcharts en noemt het enkel de belangrijkste klassen en methodes in de tekst.

Een kort codefragment kan wel helpen om de contracten tussen lagen zichtbaar te maken. Onderstaande methodesignatures tonen hoe controllers en services niet met losse arrays werken, maar met expliciete request- en resultobjecten:

```php
public function createKnowledge(CreateKnowledgeRequest $request): KnowledgeResult;
public function generateDigest(GenerateDigestRequest $request): DigestResult;
public function syncKnowledgeToNotion(Knowledge $knowledge, bool $isUpdate = false): void;
```

Door dit patroon te volgen is in elke methode meteen duidelijk wat er binnenkomt en wat er teruggegeven wordt. De controllers zelf blijven beperkt tot validatie via Symfony's `MapRequestPayload` en het delegeren naar de orchestrator. Op die manier wordt de businesslogica niet in de HTTP-laag geschreven en kunnen de orchestrators los van Symfony getest worden.

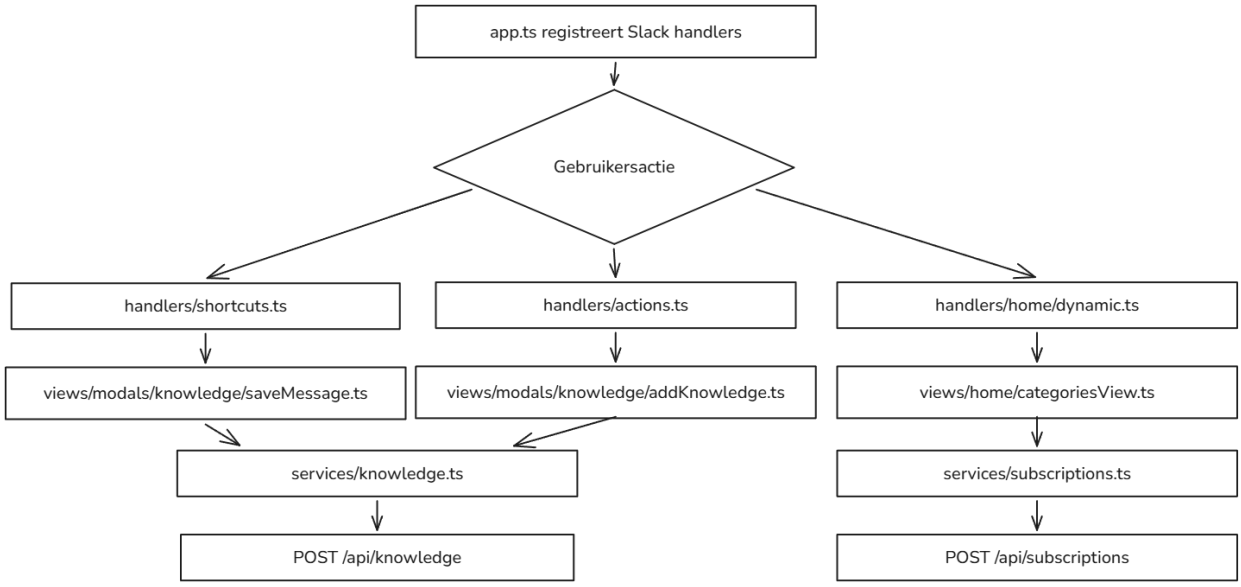
5.3 Slack-applicatie in TypeScript

De Slack-applicatie is gebouwd in TypeScript met de Bolt SDK. TypeScript helpt om de code betrouwbaarder te maken doordat types al tijdens ontwikkeling fouten kunnen tonen. De Bolt SDK vereenvoudigt het bouwen van Slack-commando's, shortcuts, modals en interactieve acties.

De applicatie gebruikt Socket Mode. Bij Socket Mode maakt de app zelf een uitgaande WebSocket-verbinding met Slack. Daardoor is er voor lokale ontwikkeling geen publieke webhook-URL nodig. Dat maakt testen en ontwikkelen eenvoudiger en veiliger.

De Slack App Home fungeert als dashboard. Gebruikers kunnen er kennis toevoegen, thematische lijsten bekijken, acties starten en abonnementen beheren. Omdat Slack-interacties stateless zijn, moest de applicatie zorgvuldig omgaan met context. Acties bevatten daarom identifiers waarmee de handler weet welke lijst, digest of flow bedoeld wordt. De registratie van events, shortcuts, actions en views gebeurt vanuit `slack/src/app.ts`, waarna gespecialiseerde handler- en viewbestanden de concrete UI-flow afhandelen.

Concreet routeert `app.ts` de Slack-events naar `handlers/shortcuts.ts` voor message- en global shortcuts, naar `handlers/actions.ts` voor interactieve knoppen en menu's, en naar `handlers/home/dynamic.ts` voor de App Home. Wanneer een modal wordt verzonden, komt die binnen via `handlers/submissions.ts`, die op basis van het `callback_id` doorroutet naar de juiste handler, bijvoorbeeld naar `handlers/submissions/digestHandler.ts` voor het genereren van een digest. De views zelf zijn opgesplitst in herbruikbare blokken onder `views/modals/` en `views/home/`, zodat dezelfde knoppen en lijsten op meerdere plaatsen consistent worden opgebouwd.



Figuur 10: Ingestieflow via Slack, met de echte handler- en viewbestanden uit slack/src.

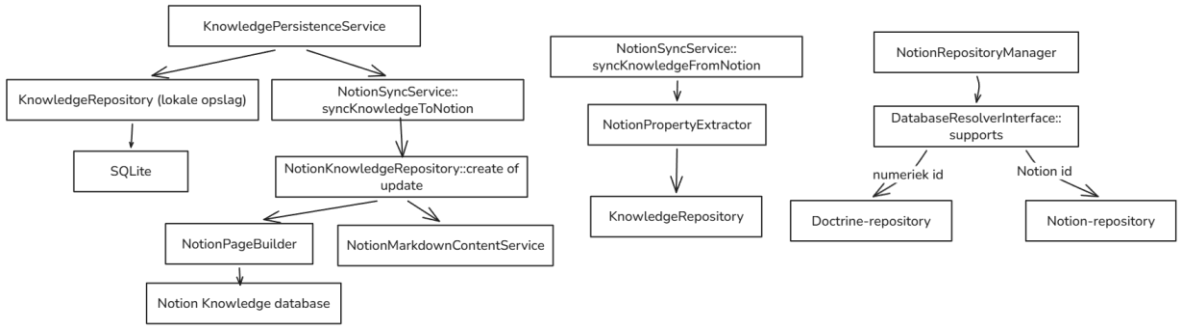
Tijdens de MVP-fase werd de Slack-code opgesplitst in kleinere modules. Handlers, views, actions en submissions kregen duidelijkere verantwoordelijkheden. Die modularisering was nodig omdat de initiële Slack-code uit de POC-fase te monolithisch werd zodra er meer interactieve flows bijkwamen.

5.4 Synchronisatie met Notion

De Notion-integratie is een van de centrale onderdelen van YapHub. De applicatie werkt met vier Notion-databases: Knowledge, Categories, Digests en Subscriptions. In de gebruikersinterface wordt de Categories-database voorgesteld als thematische lijsten. Elk domein heeft eigen mappinglogica, omdat Notion werkt met specifieke propertytypes zoals title, rich text, relation, multi-select, date en checkbox.

De synchronisatie combineert lokale opslag met Notion-opslag. SQLite wordt gebruikt voor snelle lokale verwerking en als operationele database voor de backend. Notion wordt gebruikt als leesbare, collaboratieve kennisomgeving. Bij het aanmaken of aanpassen van gegevens werkt de applicatie de relevante Notion-pagina bij en bewaart ze de Notion-identifier lokaal als referentie.

Concreet verloopt de uitvoer via `NotionSyncService`. Voor een nieuw kennisitem roept de service `NotionKnowledgeRepository::create` aan, waarbij `NotionPageBuilder` de Notion-pagina opbouwt met de juiste eigenschappen. Daarna gebruikt `NotionMarkdownContentService` de inhoud om de body van de pagina als Notion-blocks samen te stellen. De omgekeerde richting verloopt via `syncKnowledgeFromNotion`, die gewijzigde pagina's ophaalt en `NotionPropertyExtractor` gebruikt om de inhoud terug te mappen naar de lokale entiteit.



Figuur 11: Bidirectionele synchronisatie tussen lokale opslag en Notion.

Een technische uitdaging was dat Notion niet gewoon platte tekst opslaat. Notion-pagina's bestaan uit blocks. Daarom werd functionaliteit voorzien om Markdown-achtige inhoud om te zetten naar Notion-blocks. Daardoor kunnen kennisitems en digests niet alleen als properties worden bewaard, maar ook als leesbare pagina-inhoud.

Daarnaast moest rekening gehouden worden met foutafhandeling, rate limits en paginering. De Notion API werkt met limieten en cursor-gebaseerde resultaten. De integratielaag moet dus niet alleen data versturen, maar ook betrouwbaar omgaan met gedeeltelijke fouten en vervolgpagina's. Voor dat laatste werd in de Notion-services een `NotionPaginationTrait` voorzien zodat de cursorlogica niet op meerdere plaatsen werd herhaald.

De `NotionRepositoryManager` speelt hier een coordinerende rol. Hij verzamelt alle implementaties van `DatabaseResolverInterface` (zowel de Doctrine-repositories als de Notion-repositories) en kiest op basis van het meegegeven identifier-type de juiste implementatie. Een numerieke identifier verwijst naar lokale data, een Notion-UUID verwijst naar Notion. Daardoor blijft de hogere businesslogica onafhankelijk van de concrete opslagplaats.

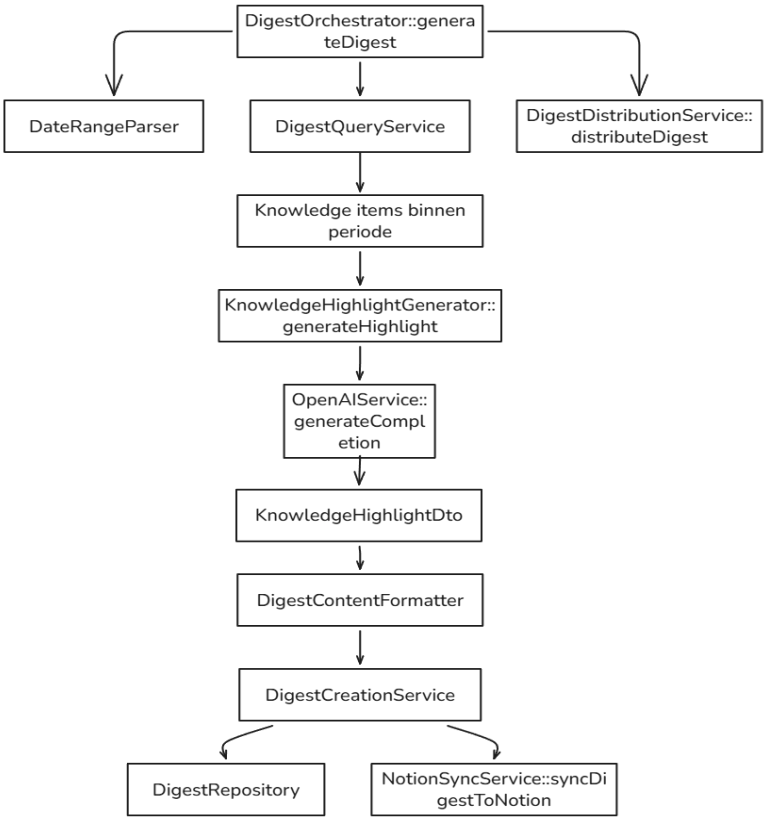
5.5 AI-integratie

De AI-integratie is bewust via een interface opgebouwd. De businesslogica is daardoor niet rechtstreeks afhankelijk van een specifieke provider. De `AiProviderInterface` definieert een methode `generateCompletion`, en `OpenAIService` voorziet de concrete implementatie. Op basis van configuratie kan er gewerkt worden met OpenAI of met OpenRouter. Voor de daadwerkelijke aanroep wordt gebruikgemaakt van het Symfony AI Bundle.

De prompt in de MVP vraagt om een korte Nederlandstalige samenvatting van twee tot drie zinnen. De template staat als constante `HIGHLIGHT_SUMMARY_PROMPT` in de digestconstanten. De temperatuur is laag ingesteld, zodat de output zo consistent en feitelijk mogelijk blijft. De samenvatting wordt opgeslagen in het `highlight`-veld van het kennisitem en als snapshot opgenomen in een digest. Dat snapshotprincipe is belangrijk omdat een digest later hetzelfde resultaat moet blijven tonen, ook als een kennisitem ondertussen gewijzigd wordt.

Wanneer de AI-aanroep faalt, valt de logica terug op een eenvoudige inkorting van de oorspronkelijke inhoud. Op die manier blijft de digest bruikbaar, zelfs als de externe AI-provider tijdelijk niet beschikbaar is. De fout wordt wel gelogd, zodat het probleem zichtbaar blijft in de observability-laag.

AI wordt hier niet gebruikt om de menselijke beoordeling te vervangen. Het systeem helpt om sneller een eerste inzicht te krijgen. De inhoud blijft afkomstig uit de toegevoegde kennisitems en de waarde van het systeem hangt af van de kwaliteit van de ingevoerde bronnen en de controle door medewerkers.



Figuur 12: AI- en digestgeneratieflow.

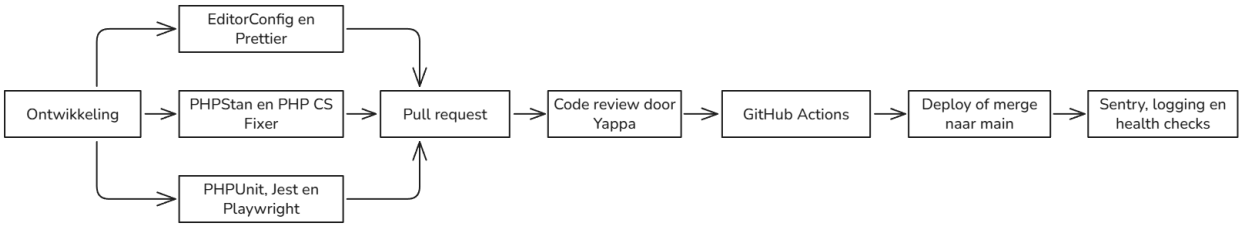
5.6 Teststrategie, CI en observability

Om de betrouwbaarheid van YapHub te verhogen, werd een teststrategie uitgewerkt met meerdere lagen. PHPUnit-tests controleren backendlogica, entiteiten en services. Voor de Slack-applicatie werd Jest gebruikt voor losse logica. Playwright-tests controleren volledige flows, bijvoorbeeld van Slack-interactie tot backendverwerking en Notion-synchronisatie. Voor externe systemen kunnen mockinstellingen gebruikt worden via `NOTION_MOCK` en `OPENAI_MOCK`, zodat tests niet afhankelijk zijn van echte Notion- of AI-aanroepen.

Het is belangrijk om hierbij te nuanceren dat deze testlaag vooral vanuit mijn eigen projectopzet en pull request kwam. Yappa gebruikt in de dagelijkse Symfony-praktijk vooral statische analyse en code style-controles zoals PHPStan en PHP CS Fixer. Voor TypeScript werd gewerkt met Prettier en EditorConfig om formatting consistent te houden. Die tooling sluit goed aan bij code reviews, omdat reviewers dan minder tijd verliezen aan stijlverschillen en meer aandacht kunnen geven aan ontwerp en onderhoudbaarheid.

In de repository draaien deze controles automatisch via GitHub Actions. De workflow `php-checks.yml` voert PHP CS Fixer in dry-run, PHPStan en PHPUnit uit voor de backend. De workflow `ts-checks.yml` doet hetzelfde voor de Slack-code: Prettier, typecheck en Jest. Bij een merge naar `main` zorgt een aparte deployment-workflow voor het uitrollen van de Slack-bot en de Symfony-backend op de Yappa-infrastructuur.

Voor observability werden logging, health checks en Sentry toegevoegd. Logging gebeurt via Monolog en gebruikt gestructureerde context-DTO's, zodat fouten terug te brengen zijn tot de juiste service of workflow. Health checks geven een eenvoudige manier om te controleren of backend en Slack-bot bereikbaar zijn. Sentry werd geïntegreerd via de `SentryBundle`, zodat runtimefouten in een productieomgeving sneller gedetecteerd en onderzocht kunnen worden.



Figuur 13: Kwaliteitsflow van ontwikkeling tot monitoring.

6 Projectaanpak, kwaliteit en overdraagbaarheid

Naast de technische realisatie zijn ook de manier van werken en de overdraagbaarheid belangrijk in deze stage. In dit hoofdstuk wordt eerst toegelicht hoe iteratief werken en code reviews een centrale rol speelden in de evolutie van het project. Daarna wordt besproken hoe de documentatie werd opgebouwd om YapHub na de stage begrijpelijk en bruikbaar te houden.

6.1 Iteratief werken en code reviews

Het project werd niet in een keer volledig ontworpen en daarna gebouwd. De aanpak was iteratief. Eerst werd een POC opgezet, daarna werden functies uitgebreid, vereenvoudigd, opgesplitst en opnieuw beoordeeld. Die manier van werken sloot aan bij de praktijk binnen Yappa, waar pull requests en code reviews een belangrijk onderdeel zijn van het ontwikkelproces.

De code reviews van de heer Davy Dewit hadden een grote invloed op mijn leerproces. In een afzonderlijk code-reviewrapport werden ruim 230 opmerkingen over 12 pull requests geanalyseerd en gegroepeerd in feedbackcategorieen. De grootste terugkerende thema's waren vereenvoudiging, het verwijderen van ongebruikte code, DTO's en builders, magic strings vervangen door enums of constants, code style, method size, dependency injection en het correct plaatsen van businesslogica.

Vooral in de eerste weken waren er veel opmerkingen, omdat ik toen nog moest wennen aan de kwaliteitsstandaarden van Yappa. Die feedback ging niet alleen over kleine codefouten, maar vooral over structuur, onderhoudbaarheid, naamgeving, verantwoordelijkheden en het vermijden van onnodige complexiteit. De evolutie doorheen de pull requests toont dat code review niet alleen een controlemechanisme was, maar ook een belangrijk leermiddel.

Een concreet voorbeeld was het gebruik van bestaande Symfony- en Doctrine-mogelijkheden in plaats van zelf boilerplate te schrijven. Zo leerde ik onder meer waarom Gedmo Timestampable nuttig is voor automatische tijdstempels en waarom bestaande bibliotheken vaak betrouwbaarder zijn dan eigen ad-hocoplossingen. Ook leerde ik pull requests kleiner en gerichter te maken, zodat review haalbaar blijft. Het volledige code-reviewrapport kan als bijlage worden toegevoegd, omdat het concreet aantoont hoe mijn codekwaliteit tijdens de stage evolueerde.

6.2 Documentatie en overdraagbaarheid

Omdat YapHub na de stage verder begrepen moet kunnen worden, was documentatie een belangrijk onderdeel van het project. De documentatie werd opgebouwd met VitePress en bevat uitleg over projectscope, architectuur, setup, Notion-integratie, Slack-flows, teststrategie en MVP-demo's. De gepubliceerde website is beschikbaar op Cloudflare Pages via knowledgehub-aks.pages.dev en vormt een digitale aanvulling op deze bachelorproef.

De VitePress-site heeft daarbij een andere functie dan de scriptie. Dit verslag legt de opdracht, keuzes, realisatie en reflectie samenhangend uit. De documentatiesite is praktischer opgezet: ze bevat een documentatiekaart, MVP-demo's met flowcharts, architectuurdocumenten, setupinformatie, backenddocumentatie, testdocumentatie en wekelijkse voortgangsrapporten. Daardoor kan een begeleider, reviewer of toekomstige developer snel doorklikken naar meer detail zonder dat de hoofdtekst van de scriptie onleesbaar lang wordt.

Tijdens de stage werd de documentatie ook opgeschoond. Dat was nodig omdat oudere documenten soms nog geplande functies beschreven alsof ze al definitief waren. Voor een overdraagbaar project is dat gevaarlijk. Documentatie moet niet alleen uitgebreid zijn, maar vooral actueel en bruikbaar.

De combinatie van code, tests en documentatie maakt het project beter overdraagbaar. Een volgende developer kan niet alleen zien wat er gebouwd is, maar ook waarom bepaalde keuzes gemaakt zijn en welke functies bewust buiten scope vielen. De VitePress-site ondersteunt die overdraagbaarheid omdat ze de losse Markdown-documenten omzet naar een navigeerbare website met duidelijke hoofdrubrieken zoals MVP, project, architectuur, workflow, techniek, roadmap en rapporten.

7 Besluit en reflectie

In dit slothoofdstuk wordt de probleemstelling uit hoofdstuk 1 opnieuw opgenomen en beantwoord op basis van het gerealiseerde MVP. Daarna volgt een evaluatie van wat dit project concreet oplevert voor Yappa. In de persoonlijke reflectie kijk ik terug op mijn leerproces, en in de aanbevelingen worden mogelijke vervolgstappen geformuleerd voor wanneer Yappa het systeem verder wil inzetten.

7.1 Antwoord op de probleemstelling

De probleemstelling van deze bachelorproef was hoe Yappa interne kennis uit dagelijkse communicatie eenvoudig kan capteren, automatisch kan verwerken en opnieuw beschikbaar kan maken via een gestructureerde kennisomgeving.

Het gerealiseerde MVP geeft hier een concreet antwoord op. Slack wordt gebruikt omdat medewerkers daar al dagelijks werken. Medewerkers kunnen kennis toevoegen via shortcuts, modals en de App Home. De Symfony-backend verwerkt de informatie, haalt waar mogelijk URL-inhoud op, bewaart de data lokaal en synchroniseert naar Notion. AI-samenvattingen en digests zorgen ervoor dat kennis niet alleen opgeslagen wordt, maar ook sneller opnieuw bruikbaar wordt. Abonnementen en distributie via Slack Direct Messages maken het mogelijk om kennis gericht bij de juiste medewerkers te brengen.

Daarmee is de oorspronkelijke kloof tussen vluchtige Slack-communicatie en structurele Notion-documentatie kleiner geworden. Het systeem vervangt de menselijke kennisdeling niet, maar ondersteunt ze met automatisering en structuur.

7.2 Resultaat voor Yappa

Voor Yappa levert YapHub een werkend fundament voor intern kennisbeheer. Het bedrijf beschikt over een MVP waarin kennisitems, thematische lijsten, digests en abonnementen doorheen Slack, Symfony en Notion met elkaar verbonden zijn. De oplossing biedt waarde omdat ze aansluit bij bestaande tools en gewoontes.

Het belangrijkste voordeel is dat kennis minder snel verloren gaat. Wat vroeger in Slack kon verdwijnen, kan nu worden opgeslagen als Notion-pagina. Daarnaast verlaagt de AI-samenvatting de drempel om gedeelde informatie te begrijpen. Digests maken het mogelijk om op vaste momenten een overzicht te krijgen zonder alle losse items manueel te verzamelen.

Ook technisch heeft Yappa voordeel. Het project bevat een gelaagde Symfony-backend, een modulaire TypeScript Slack-app, testdekking, documentatie en monitoringaanzetten. Daardoor is het geen losse demo gebleven, maar een basis waarop verder gebouwd kan worden.

7.3 Persoonlijke reflectie

Tijdens deze stage heb ik geleerd hoe het is om een project van start tot einde op te zetten in een professionele omgeving. In de eerste weken lag de nadruk op verkennen, vragen stellen en een POC bouwen. Daarna verschoof de focus naar het verfijnen van die oplossing tot een MVP dat waarde kon tonen aan Yappa.

Technisch heb ik veel geleerd over Symfony, Doctrine, Slack-integraties, Notion API's, AI-integratie, testing en CI/CD. Vooral de code reviews waren bepalend. De vele opmerkingen leerden mij niet alleen hoe iets werkt, maar ook waarom een bepaalde aanpak beter onderhoudbaar is. Ik leerde bijvoorbeeld kritischer kijken naar abstracties, pull request-grootte, testbaarheid en het gebruik van bestaande frameworkfunctionaliteit.

Daarnaast heb ik inzicht gekregen in projectwerking en bedrijfsvoering. Door mee te volgen in meetings, wrap-ups en teammomenten zag ik hoe ontwikkeling samenhangt met planning, budget, billability, gegenereerde opbrengst en samenwerking tussen sales, marketing en development. Begrippen zoals billability en GDU kregen daardoor een concrete betekenis binnen een bedrijfscontext.

Ook de werksfeer bij Yappa had een positieve impact. De combinatie van informele communicatie, duidelijke feedback en professionele verwachtingen zorgde ervoor dat ik mij kon ontwikkelen als softwareontwikkelaar en als teamlid.

7.4 Aanbevelingen

Hoewel het MVP een bruikbaar resultaat oplevert, zijn er nog logische uitbreidingen mogelijk. Een eerste aanbeveling is om het systeem intern vrij te laten gebruiken binnen Yappa. Omdat het project bedoeld is als interne toepassing, kunnen medewerkers het meteen in hun eigen werkcontext uitproberen. Hun feedback kan daarna gebruikt worden om de UI en UX te verbeteren, vooral rond de Slack App Home, modals, thematische lijsten en Notion-weergaven.

Een tweede aanbeveling is om AI-samenvattingen verder te personaliseren. De huidige MVP gebruikt een algemene Nederlandstalige samenvatting. In een volgende fase kunnen prompts per doelgroep of thematische lijst worden ingevoerd, zodat developers, marketeers en projectmanagers elk een aangepast perspectief krijgen.

Een derde aanbeveling is om zoekfunctionaliteit en kwaliteitscontrole uit te breiden. Naarmate de kennisbank groeit, worden filtering, archivering en duplicaatdetectie belangrijker. Ook koppelingen met Jira/GitHub of andere projecttools kunnen waardevol zijn om kennisitems te verbinden met concrete tickets of klantprojecten.

Ten slotte is het belangrijk om de kosten en het gebruik van AI te blijven monitoren. AI-samenvattingen leveren waarde, maar hun kost hangt af van volume, modelkeuze en promptlengte.

8 Literatuurlijst

Deze literatuurlijst is een werkversie voor de Markdown-versie van het verslag. In de Word-versie kan ze automatisch worden aangemaakt via het tabblad **Verwijzingen**, met dezelfde bronnen in een consistente referentiestijl.

- Doctrine Project. (2026). *Doctrine ORM Documentation*. https://www.doctrine-project.org/projects/doctrine-orm/en/current/
- GitHub Docs. (2026). *GitHub Actions Documentation*. https://docs.github.com/actions
- Notion Developers. (2026). *Notion API Reference*. https://developers.notion.com/reference/intro
- OpenAI. (2026). *OpenAI API Documentation*. https://platform.openai.com/docs/
- OWASP Foundation. (2026). *Server-Side Request Forgery Prevention Cheat Sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
- Playwright. (2026). *Playwright Test Documentation*. https://playwright.dev/docs/intro
- Slack. (2026). *Bolt for JavaScript Documentation.* https://slack.dev/bolt-js/
- Symfony. (2026). *Symfony Documentation*. https://symfony.com/doc/current/
- Yappa. (2026). *Interne projectdocumentatie Yappa Knowledge Hub*. Niet-publieke documentatie binnen de projectrepository.
- Dewit, D. (2026). *Code-reviewopmerkingen op de Symfony-pull requests van het stageproject*. Intern code-reviewrapport, opgenomen als mogelijke bijlage.
- Sarikaya, M. (2026). *YapHub-documentatiesite*. https://knowledgehub-aks.pages.dev/

9 Bijlagenlijst

10 Bijlage A: logboek

Week 1: opstart en fundament

In de eerste week lag de nadruk op kennismaking met Yappa, het team en de gebruikte tools. De projectstructuur werd opgezet, de eerste documentatieomgeving werd voorbereid en de Slack-app werd aangemaakt. Daarnaast werden de eerste user stories en het Product Requirements Document opgesteld. Deze week vormde de basis voor de verdere POC.

Week 2: POC Slack en Notion

In de tweede week werd gewerkt aan de eerste koppeling tussen Slack en Notion. Het doel was om te testen of kennis vanuit Slack naar een Notion-database kon worden gebracht. Daarnaast maakte ik verder kennis met de agile werkwijze, het Jira-bord, sprints en dagelijkse overlegmomenten.

Week 3: repositorystructuur en servermigratie

In week drie werd de backlog verder georganiseerd en groeide het project naar een multi-repository structuur. Er werd gewerkt aan CI/CD-pipelines en aan de migratie naar Yappa-infrastructuur. Deze fase maakte duidelijk dat het project niet alleen uit code bestond, maar ook uit hosting, deployment en ontwikkelproces.

Week 4: Notion SDK en AI-samenvattingen

In week vier werd de Notion-integratie verbeterd door over te stappen naar een officiele SDK. Ook werd de backendinfrastructuur voor AI-samenvattingen voorbereid. De Slack-integratie werd verder uitgebreid met modals en interacties.

Week 5: Slack App Home en MVP-structuur

In week vijf verschoof de interface van losse modals naar een meer centrale Slack App Home. Dat verbeterde de gebruikerservaring omdat acties op een herkenbare plaats samenkwamen. De backlog werd ook opgesplitst in kernflows voor input, processing en output.

Week 6: refactoring en testen

In week zes werd veel aandacht besteed aan refactoring. Hardcoded waarden werden vervangen door constants, de Slack-code werd modularer en end-to-endtests met Playwright werden toegevoegd. Ook het abonnementensysteem begon vorm te krijgen.

Week 7: architecturale afronding

In week zeven werden het orchestrator pattern en Result DTO’s verder toegepast. Technische schuld uit eerdere fases werd aangepakt en de documentatieomgeving kreeg een grotere update. Bugs in Slack App Home-navigatie en UI-state werden opgelost.

Week 8: scopeherziening en kwaliteit

In week acht werd de scope opnieuw aangescherpt. Grote pull requests werden opgesplitst in kleinere, beter reviewbare delen. Pre-commit hooks en lokale kwaliteitscontroles werden verder geintegreerd, zodat formattering en statische analyse vroeger in het proces fouten konden vinden.

Week 9: vereenvoudiging en URL-verwerking

Na een korte onderbreking werd in week negen sterk gewerkt aan code review-feedback. Een grote POC-pull request werd verkleind en vereenvoudigd. Daarnaast werd URL-inhoudsextractie uitgewerkt, zodat de AI later betere input zou krijgen voor samenvattingen.

Week 10: distributie en abonnementen

In week tien werd de end-to-end POC-flow afgerond en werden abonnementen en digestdistributie verder gebouwd. De backend kreeg een `Subscription`-domein en een distributielaag die digests via Slack Direct Messages naar abonnees kan sturen.

Week 11: scheduler, UX en documentatie

In week elf werd de scheduler voor automatische digests uitgewerkt. Daarnaast werd de Slack-gebruikerservaring verbeterd, onder meer door modals beter op elkaar te laten aansluiten. De documentatiesite werd aangevuld met demo's en uitleg over de MVP-functionaliteit.

Week 12: stabilisatie, monitoring en finale documentatie

In week twaalf verschoof de focus naar stabilisatie. Er werd gewerkt aan Sentry voor foutopvolging, extra PHPUnit- en Playwright-tests, documentatieopschoning en finale MVP-demo's. Deze week markeerde de overgang van featureontwikkeling naar oplevering en overdraagbaarheid.