howest
university of applied sciences

Herontwikkeling van de Aphia Quality Control Toolbox

Tom Thomas

Bachelorproef Toegepaste Informatica

2026

VOORWOORD

Deze bachelorproef sluit mijn opleiding Toegepaste Informatica aan Howest af. Tijdens mijn stage bij het Vlaams Instituut voor de Zee (VLIZ) werkte ik mee aan de verdere ontwikkeling van de Aphia Quality Control Toolbox. Die stage gaf me de kans om mijn technische kennis in de praktijk te brengen en tegelijk nieuwe inzichten te krijgen binnen het domein van taxonomie, datakwaliteit en softwareontwikkeling.

Graag wil ik mijn stagebegeleider Ben Boydens bedanken voor de begeleiding, ondersteuning en feedback gedurende de stage. Ook wil ik Bart Vanhoorne bedanken voor het vertrouwen en de mogelijkheid om dit project binnen VLIZ uit te voeren.

Verder wil ik VLIZ bedanken voor de aangename werkomgeving en de kans om ervaring op te doen binnen een professionele ontwikkelomgeving, en mijn begeleiders en docenten van Howest voor hun ondersteuning gedurende mijn opleiding en bij het uitwerken van deze bachelorproef.

Mijn dank gaat ook uit naar de mensen in mijn omgeving, voor hun steun en aanmoediging.

Tom Thomas

Brugge, juni 2026

AI GEBRUIK BINNEN HET OLOD

Tijdens het uitvoeren van deze bachelorproef werd generatieve artificiële intelligentie gebruikt als ondersteunend hulpmiddel bij het schrijfproces. AI werd ingezet voor het herformuleren van teksten, het verbeteren van taalgebruik en het controleren van structuur.

Alle technische inhoud, zoals analyses, implementaties en benchmarkresultaten, is gebaseerd op eigen werkzaamheden tijdens de stage bij VLIZ. De inhoud van deze bachelorproef werd door de auteur gecontroleerd, aangepast en gevalideerd.

De inhoudelijke keuzes, analyses en implementaties zijn volledig eigen werk; AI ondersteunde enkel het schrijfproces.

ABSTRACT

Deze bachelorproef maakt deel uit van de opleiding Toegepaste Informatica aan Howest en werd uitgevoerd tijdens een stage bij het Vlaams Instituut voor de Zee (VLIZ). De opdracht richtte zich op de verdere ontwikkeling van de Aphia Quality Control Toolbox, een systeem dat kwaliteitsproblemen binnen de Aphia-taxonomiedatabank automatisch detecteert.

Bij aanvang van de stage bleek de bestaande toolbox verschillende uitdagingen te kennen. De architectuur was doorheen de jaren organisch gegroeid, waardoor verantwoordelijkheden niet altijd duidelijk gescheiden waren. Verschillende kwaliteitscontroles hadden hoge uitvoeringstijden, en de gebruikersinterface bood beperkte ondersteuning voor het efficiënt analyseren van resultaten.

Om die problemen aan te pakken werd de architectuur herwerkt naar een modulaire structuur. SQL-queries werden geoptimaliseerd via execution plans, tijdelijke tabellen, indexering en vroegtijdige filtering. Aanvullend werden meerdere cachingstrategieën ontwikkeld en geëvalueerd om de responstijden van performantiegevoelige kwaliteitscontroles verder te verbeteren. De gebruikersinterface werd vernieuwd met extra functionaliteit voor filtering, navigatie en resultaatanalyse, en nieuwe kwaliteitscontroles werden toegevoegd om een breder spectrum aan datakwaliteitsproblemen te dekken.

De verbeteringen leverden meetbare resultaten op. Sommige queries die oorspronkelijk meer dan twee minuten nodig hadden, werden teruggebracht tot ongeveer twintig seconden, terwijl andere controles van meerdere seconden naar minder dan één seconde gingen. De onderhoudbaarheid van de codebasis werd verbeterd en de toolbox kreeg nieuwe validatiemechanismen.

Die resultaten laten zien dat architecturale herwerking, gerichte query-optimalisatie, selectieve caching en functionele uitbreiding samen een effectieve aanpak zijn voor het verbeteren van kwaliteitscontroletoepassingen in een productieomgeving.

INHOUD

LIJST VAN FIGUREN

LIJST VAN TABELLEN

LIJST VAN AFKORTINGEN EN BEGRIPPEN

Deze woordenlijst bevat afkortingen, technische termen en domeinspecifieke begrippen die in deze bachelorproef worden gebruikt. De definities geven een beknopte toelichting bij concepten uit softwareontwikkeling, databankbeheer, taxonomie en datakwaliteitscontrole.

A

Accepted taxa
Taxa die in de databank als geldige of voorkeurstermen worden beschouwd.

Aphia
De databank en het platform binnen VLIZ waarin taxonomische informatie wordt beheerd. Aphia vormt de technische basis voor toepassingen zoals WoRMS.

B

Backendlogica
De code aan de serverkant die gegevens verwerkt en applicatieregels uitvoert.

Basionym
De oorspronkelijke wetenschappelijke naam waarop een latere naamcombinatie gebaseerd is.

Benchmarkmetingen
Metingen waarmee prestaties, zoals uitvoeringstijd, vergelijkbaar worden gemaakt.

Breadcrumbs
Navigatie-elementen die tonen waar de gebruiker zich bevindt en waarmee vorige stappen snel bereikbaar zijn.

Bronvermeldingen
Verwijzingen naar de bronnen waarop gegevens in de databank gebaseerd zijn.

C

Caching
Het tijdelijk bewaren van eerder berekende resultaten zodat dezelfde berekening niet telkens opnieuw nodig is.

CTE (Common Table Expression)
SQL-constructies waarmee een tijdelijk benoemd resultaat binnen een query kan worden gebruikt

D

Database-engine
Het onderdeel van de databanksoftware dat queries uitvoert en bepaalt hoe gegevens worden verwerkt.

Database-gebaseerde caching
Een cachingaanpak waarbij snapshots en metadata in databanktabellen worden opgeslagen.

Databasenormalisatie
Het opsplitsen van gegevens over meerdere tabellen om dubbele opslag en tegenstrijdige gegevens te beperken.

Datagedreven architectuur
Een architectuur waarin databankgegevens en dataverwerking centraal staan in de werking van de toepassing.

Datakwaliteit
De mate waarin gegevens correct, volledig, consistent en bruikbaar zijn.

Displaynaam
De naam zoals die aan gebruikers in de interface wordt getoond.

DMT (Data Management Team)
De groep gebruikers binnen VLIZ die wetenschappelijke gegevens beheert, valideert en corrigeert.

DOI
Afkorting voor Digital Object Identifier, een vaste identificatiecode voor publicaties of bronnen.

E

Editrights
Rechten die bepalen welke gegevens een gebruiker in Aphia mag aanpassen.

Execution plans
Overzichten waarin SQL Server toont hoe een query wordt uitgevoerd en waar de grootste kost zit.

F

File-based caching
Een cachingaanpak waarbij resultaten als bestanden worden opgeslagen.

Full-aanpak
Een cachingaanpak waarbij alle opgeslagen identifiers in een keer worden verwerkt.

G

Gebeurtenisgestuurde aanpak
Een aanpak waarbij acties automatisch worden uitgevoerd wanneer een gebeurtenis plaatsvindt, bijvoorbeeld wanneer een record wordt aangemaakt of gewijzigd.

Gebruikersinterface
Het deel van de toepassing waarmee gebruikers werken, zoals overzichtspagina's, filters en resultatenpagina's.

Genus
Een taxonomische rang boven soort. Meerdere verwante soorten kunnen tot hetzelfde genus behoren.

H

Habitatvlaggen
Aanduidingen die beschrijven in welk leefgebied een taxon voorkomt.

Hash
Een berekende waarde die gebruikt kan worden om te controleren of iets, zoals een querydefinitie, gewijzigd is.

Hyperlinks
Klikbare verwijzingen naar webpagina's of andere online bronnen.

I

Indexering
Het toevoegen van indexen aan tabellen of tussenresultaten zodat de databank sneller kan zoeken en koppelen.

J

Joins
SQL-bewerkingen waarmee gegevens uit meerdere tabellen worden samengebracht.

K

Kinderen
Onderliggende taxa in een taxonomische hierarchie.

L

Live uitvoering
Het rechtstreeks opnieuw uitvoeren van een controle tegen de databank, zonder gecachte resultaten te gebruiken.

M

Mariene organismen
Organismen die in of rond de zee leven.

Masonry-layout
Een visuele indeling waarbij blokken in kolommen worden geplaatst om schermruimte beter te benutten.

Metadata
Gegevens over andere gegevens, zoals het tijdstip waarop een snapshot werd opgebouwd.

N

Nomenclatorische code
Een formele set regels voor het geven en gebruiken van wetenschappelijke namen.

Nomenclatorische principes
Regels en uitgangspunten die bepalen hoe wetenschappelijke namen correct worden toegepast.

Nomenclatorische regels
Concrete regels voor de correcte vorm en toepassing van wetenschappelijke namen.

P

Paginering
Het opdelen van resultaten in pagina's zodat niet alles tegelijk getoond of verwerkt moet worden.

Parentrelaties
Relaties waarbij een taxon onder een hoger taxon valt.

Parenttaxa
Bovenliggende taxa in een taxonomische hierarchie.

Permissies
Toegangsrechten die bepalen wat een gebruiker wel of niet mag zien of wijzigen.

Persistent
Blijvend opgeslagen, zodat gegevens beschikbaar blijven na de uitvoering van een proces.

Productieomgeving
De live omgeving waarin de toepassing echt door gebruikers wordt gebruikt, in tegenstelling tot een lokale test- of ontwikkelomgeving.

Q

QC-toolbox
Verkorte naam voor de Quality Control Toolbox.

Quality Control Toolbox
De toolbox in Aphia die controles uitvoert om mogelijke datakwaliteitsproblemen automatisch op te sporen.

Query
Een opdracht aan de databank om gegevens op te halen, te combineren of te controleren.

R

Redundantie
Onnodige herhaling van dezelfde gegevens op meerdere plaatsen.

Relationele databank
Een databank waarin gegevens in tabellen worden bewaard en via relaties met elkaar verbonden zijn.

Rendering
Het omzetten van gegevens en templates naar de uiteindelijke weergave in de interface.

Runtimeverbeteringen
Verbeteringen die ervoor zorgen dat een controle of query sneller uitgevoerd wordt.

S

Scope
De afbakening van welke gegevens of resultaten op dat moment worden meegenomen.

Snapshot
Een opgeslagen momentopname van resultaten of identifiers van een kwaliteitscontrole.

Subgenera
Taxonomische onderverdelingen binnen een genus.

Subspecies
Taxonomische onderverdelingen binnen een soort.

Synoniemen
Alternatieve wetenschappelijke namen die naar hetzelfde taxon verwijzen.

T

Taxa
Het meervoud van taxon.

Taxon
Een benoemde groep binnen de taxonomie. Dat kan bijvoorbeeld een soort, genus of hogere groep zijn.

Taxonomie
Het vakgebied dat organismen ordent, benoemt en in groepen plaatst.

Taxonomische databank
Een databank die informatie bewaart over organismen, wetenschappelijke namen en de relaties tussen die namen.

Tijdelijke tabellen
Tabellen die tijdelijk worden aangemaakt om tussenresultaten op te slaan tijdens queryverwerking.

Type specimens
Referentie-exemplaren waarop de beschrijving of naamgeving van een taxon gebaseerd is.

U

Unaccepted taxa
Taxa die niet als geldige voorkeursterm gelden, maar bijvoorbeeld als synoniem naar een accepted taxon verwijzen.

URL-syntaxis
De correcte structuur van een URL.

V

Validatieplatform
Een systeem dat controles uitvoert om gegevens te valideren en problemen zichtbaar te maken.

VLIZ
Het Vlaams Instituut voor de Zee. Dit instituut beheert mariene gegevens en ontwikkelt digitale systemen om die gegevens te bewaren en beschikbaar te maken.

W

Wetenschappelijke classificatie
De manier waarop organismen in taxonomische groepen worden geplaatst.

Windowed-aanpak
Een cachingaanpak waarbij resultaten in kleinere batches worden verwerkt.

WoRMS (World Register of Marine Species)
De volledige naam van WoRMS. Dit is een internationaal gebruikte databank die wordt gebruikt als referentiebron voor wetenschappelijke namen en taxonomische informatie over mariene soorten.

1 INLEIDING

1.1 CONTEXT VAN DE BACHELORPROEF

Deze bachelorproef maakt deel uit van de opleiding Toegepaste Informatica aan Howest. Als onderdeel van die opleiding liep de student stage bij het Vlaams Instituut voor de Zee (VLIZ), waar gedurende een volledig semester werd meegewerkt aan een softwareproject binnen de Aphia-databank. [1]

Aphia is de taxonomische databank achter verschillende mariene informatiesystemen, waaronder het World Register of Marine Species (WoRMS). [2] [3] Om de kwaliteit van de opgeslagen gegevens te bewaken beschikt Aphia over een Quality Control Toolbox (QC-toolbox): een verzameling kwaliteitscontroles die mogelijke inconsistenties, ontbrekende gegevens en andere datakwaliteitsproblemen detecteert.

De stage richtte zich op de verdere ontwikkeling van die QC-toolbox, met aandacht voor zowel technische verbeteringen als functionele uitbreidingen. Deze bachelorproef beschrijft de analyse, implementatie en resultaten van dat werk.

1.2 DOELSTELLING VAN DE BACHELORPROEF

Deze bachelorproef beschrijft de werkzaamheden die tijdens de stage werden uitgevoerd en evalueert de gerealiseerde verbeteringen aan de Aphia QC-toolbox.

Het project draaide niet om één specifiek technisch probleem, maar om een combinatie van uitdagingen op het vlak van onderhoudbaarheid, performantie, schaalbaarheid en gebruiksvriendelijkheid. De bestaande toolbox bevatte kwaliteitscontroles met uiteenlopende uitvoeringstijden en een architectuur die doorheen de jaren organisch was gegroeid.

Het doel was die bestaande omgeving verder te ontwikkelen binnen een professionele context, met oplossingen die performanter, onderhoudbaarder en uitbreidbaarder zijn voor toekomstige ontwikkeling.

1.3 CENTRALE PROBLEEMSTELLING

De centrale probleemstelling van deze bachelorproef luidt:

Hoe kunnen de onderhoudbaarheid, schaalbaarheid en performantie van de Aphia QC-toolbox verbeterd worden binnen een productieomgeving, zonder daarbij de functionaliteit en gebruiksvriendelijkheid voor eindgebruikers te verminderen?

Om die vraag te beantwoorden werden verschillende aspecten van de toolbox onderzocht: de bestaande architectuur werd geanalyseerd en gerichte verbeteringen doorgevoerd op het vlak van codeorganisatie, SQL-performantie, cachingstrategieën, gebruikersinterface en functionele uitbreiding.

De impact van die wijzigingen werd geëvalueerd aan de hand van performantiemetingen en een analyse van de gerealiseerde verbeteringen.

1.4 STRUCTUUR VAN HET DOCUMENT

Deze bachelorproef is opgebouwd uit een reeks hoofdstukken die elk een specifiek onderdeel van het project behandelen.

Hoofdstuk 2 introduceert VLIZ, Aphia en de rol van de QC-toolbox binnen de organisatie. Daarnaast worden de gebruikte technologieën en de rol van het Data Management Team besproken. Hoofdstuk 3 beschrijft de oorspronkelijke situatie van de toolbox en de voornaamste uitdagingen die tijdens de analysefase werden vastgesteld. Hoofdstuk 4 behandelt de databasestructuur en technische context van Aphia.

De hoofdstukken 5 tot en met 8 behandelen de technische en functionele verbeteringen: de architecturale herwerking, SQL-optimalisaties, cachingstrategieën, gebruikersinterfaceverbeteringen en functionele uitbreidingen.

Hoofdstuk 9 bespreekt de resultaten en impact van de uitgevoerde werkzaamheden. Hoofdstuk 10 bevat een persoonlijke reflectie op de stage. Hoofdstuk 11 sluit af met de belangrijkste conclusies en aanbevelingen voor toekomstige ontwikkelingen.

2 VLIZ EN APHIA CONTEXT

2.1 VLIZ

Het Vlaams Instituut voor de Zee (VLIZ) is een wetenschappelijk instituut gevestigd in Oostende dat zich bezighoudt met marien onderzoek en het beheer van mariene gegevens. Het ondersteunt nationale en internationale onderzoeksinitiatieven en beheert verschillende databanken en digitale platformen die wereldwijd worden gebruikt door onderzoekers, beleidsmakers en andere organisaties [1].

Figuur 2.1: VLIZ logo

Een centrale taak van het instituut is het verzamelen, beheren en beschikbaar stellen van wetenschappelijke gegevens. VLIZ beschikt daarvoor over een eigen IT-afdeling die de informatiesystemen ontwikkelt en onderhoudt die onderzoekers en databeheerders gebruiken voor het opslaan en raadplegen van wetenschappelijke informatie [1].

Tijdens mijn stage was ik onderdeel van die IT-afdeling. Mijn opdracht bestond erin een bestaande kwaliteitscontrolemodule binnen het Aphia-platform te analyseren en te herwerken. De nadruk lag op performantieverbeteringen en een herziene architectuur, met ook aandacht voor de gebruikservaring van de eindgebruikers.

2.2 APHIA AND WORMS

Aphia is een taxonomische databank die ontwikkeld en onderhouden wordt binnen VLIZ. Het platform bevat informatie over mariene organismen, hun wetenschappelijke classificatie en de relaties tussen taxa. Verschillende wetenschappelijke toepassingen binnen VLIZ zijn op Aphia gebouwd [2] [3].

Een van die toepassingen is het World Register of Marine Species (WoRMS), een internationaal gebruikte databank die een actuele lijst van mariene soorten bijhoudt. Onderzoekers, overheidsinstanties en internationale organisaties raadplegen WoRMS als referentie voor taxonomische informatie [3] [2].

Figuur 2.2: WoRMS logo

Datakwaliteit is binnen taxonomische databanken van groot belang, omdat fouten en inconsistenties zich kunnen verspreiden naar andere systemen die dezelfde gegevens gebruiken. Denk aan ontbrekende bronvermeldingen, ongeldige relaties tussen taxa of verouderde classificaties. Om dergelijke problemen op te sporen en te corrigeren, beschikt Aphia over een kwaliteitscontroletoolbox, ook wel de QC-toolbox genoemd.

2.3 HET DATA MANAGEMENT TEAM (DMT)

Binnen Aphia wordt een onderscheid gemaakt tussen technische ontwikkelaars en de gebruikers die verantwoordelijk zijn voor het beheer van de wetenschappelijke gegevens. Die laatste groep wordt binnen VLIZ aangeduid als het Data Management Team (DMT).

Het DMT beheert en valideert taxonomische informatie binnen de verschillende databanken die op Aphia steunen. In de praktijk gaat het om het toevoegen en aanpassen van taxa, het beheren van bronvermeldingen en het oplossen van datakwaliteitsproblemen. Daarmee zijn zij de voornaamste gebruikers van de kwaliteitscontroletoolbox.

Hoewel het DMT nauw samenwerkt met de IT-afdeling, zijn de teamleden voornamelijk domeinexperts en geen softwareontwikkelaars. Ze hebben diepgaande kennis van taxonomie en databeheer, terwijl de IT-afdeling de technische implementatie en ondersteuning op zich neemt. Die samenwerking is nodig om te zorgen dat de ontwikkelde functionaliteiten aansluiten bij wat de eindgebruikers effectief nodig hebben.

Tijdens mijn stage maakte ik gebruik van bestaande meldingen van het DMT, waarin zij mogelijke problemen of gewenste controles hadden aangegeven. Die meldingen waren een nuttige vertrekbasis voor het identificeren van verbeterpunten in de QC-toolbox.

2.4 DOEL VAN DE QC TOOLBOX

De Aphia-databank wordt voortdurend bijgewerkt door verschillende gebruikers, wat het bewaken van de gegevenskwaliteit noodzakelijk maakt. Zelfs kleine fouten kunnen op termijn leiden tot grotere problemen wanneer de gegevens worden hergebruikt door andere systemen of organisaties.

De QC-toolbox bevat een verzameling controles die potentiële problemen in de databank opsporen, zoals ontbrekende bronvermeldingen, ontbrekende type specimens, inconsistente relaties tussen taxa of verouderde gegevens.

De resultaten worden aan de gebruikers gepresenteerd, zodat zij de betrokken records kunnen onderzoeken en waar nodig corrigeren. Zo helpt de QC-toolbox het DMT bij het op peil houden van de datakwaliteit in Aphia.

De basis van de QC-toolbox was al aanwezig vóór de start van mijn stage, maar het systeem was nog niet klaar voor gebruik door de beoogde eindgebruikers. Mijn stage richtte zich daarom op de verdere uitwerking, optimalisatie en voorbereiding op productiegebruik van de toolbox.

2.5 EDITRIGHTS EN PERMISSIES

Niet alle gebruikers binnen Aphia hebben dezelfde bevoegdheden. Afhankelijk van hun expertise en verantwoordelijkheden krijgen gebruikers toegang tot specifieke delen van de databank. Dit systeem van toegangsrechten wordt binnen Aphia aangeduid als editrights.

Met editrights kunnen gebruikers enkel gegevens aanpassen waarvoor zij verantwoordelijk zijn. Een specialist die verantwoordelijk is voor een bepaalde taxonomische groep krijgt bijvoorbeeld enkel toegang tot de taxa binnen zijn of haar bevoegdheidsdomein. Zo blijft het beheer van de databank overzichtelijk en wordt het risico op ongewenste wijzigingen beperkt.

Die rechtenstructuur heeft ook een technische impact op de QC-toolbox. Bij het uitvoeren van kwaliteitscontroles wordt rekening gehouden met de editrights van de ingelogde gebruiker. Gebruikers kunnen ervoor kiezen om enkel de resultaten te zien die betrekking hebben op hun eigen domein, of om alle resultaten van een controle te raadplegen. Ook wanneer alle resultaten zichtbaar zijn, blijven de bewerkingsrechten beperkt tot de taxa waarvoor de gebruiker bevoegd is.

Dit maakt de SQL-queries en cachingmechanismen die tijdens het project werden ontwikkeld complexer. Afhankelijk van de gekozen weergave en de toegekende editrights kunnen verschillende gebruikers immers een verschillend resultaat ontvangen voor dezelfde kwaliteitscontrole.

2.6 GEBRUIKTE TECHNOLOGIEËN

Technologie	Toepassing binnen het project
PHP	Implementatie van de backendlogica van Aphia en de QC-toolbox
SQL Server	Opslag van gegevens en uitvoering van de geoptimaliseerde queries
Twig	Rendering van de gebruikersinterface
GitLab	Versiebeheer en samenwerking binnen het ontwikkelteam
Jira	Opvolging van taken en bugs en projectwerkzaamheden
WAMP	Lokale omgeving voor ontwikkeling en testen

Tabel 2.1: Gebruikte technologieën

3 PROBLEEMDEFINITIE EN INITIËLE SITUATIE

3.1 BESTAANDE QC-ARCHITECTUUR

Bij de start van de stage beschikte Aphia al over een kwaliteitscontroletoolbox met verschillende controles voor het opsporen van datakwaliteitsproblemen. De implementatie was echter grotendeels gecentraliseerd in een beperkte set bestanden, waardoor de architectuur doorheen de jaren steeds omvangrijker was geworden.

De kern van die architectuur was een centrale bibliotheek met een groot aantal kwaliteitscontroles. Naarmate nieuwe controles werden toegevoegd, groeide die structuur uit tot een geheel dat steeds moeilijker te overzien was. Functionaliteiten met uiteenlopende verantwoordelijkheden kwamen dicht bij elkaar te liggen, waardoor wijzigingen makkelijker onbedoelde gevolgen konden hebben voor andere onderdelen van het systeem.

Hoewel de architectuur functioneel bleef, was ze een uitdaging voor verdere uitbreiding en onderhoud. Het toevoegen of aanpassen van controles vereiste een grondige kennis van de bestaande structuur, wat de nood aan een meer modulaire aanpak deed groeien.

3.2 BESTAANDE QC-WORKFLOW

De QC-toolbox was ontworpen om potentiële problemen in de databank automatisch op te sporen aan de hand van vooraf gedefinieerde controles. Elke controle richtte zich op een specifiek type datakwaliteitsprobleem, zoals ontbrekende gegevens of inconsistente relaties tussen taxa.

Wanneer een controle werd uitgevoerd, werden de relevante records opgehaald en beschikbaar gemaakt aan de gebruiker. Databeheerders konden de betrokken gegevens verder onderzoeken en indien nodig corrigeren. De toolbox hield daarbij rekening met de editrights van de ingelogde gebruiker, zodat iedereen enkel toegang kreeg tot relevante informatie.

De basisfunctionaliteit was aanwezig, maar er waren beperkingen op het vlak van performantie, onderhoudbaarheid en gebruikservaring. Die beperkingen werden tijdens de stage onderzocht en vormden de aanleiding voor de herwerking van het systeem.

3.3 PERFORMANTIEPROBLEMEN

De QC-toolbox gebruikte een reeks SQL-queries om problemen in de databank te detecteren. Naarmate de databank groeide en de controles complexer werden, nam de uitvoeringstijd van bepaalde queries toe. Sommige controles moesten grote hoeveelheden gegevens verwerken of maakten gebruik van complexe joins tussen tabellen.

Daarnaast werden bepaalde berekeningen herhaald uitgevoerd bij het genereren van resultaten. Dat leverde functioneel correcte uitkomsten op, maar vertraagde het systeem merkbaar. Voor gebruikers betekende dit dat sommige controles langer nodig hadden dan verwacht.

Het analyseren en optimaliseren van deze queries werd daarom een van de centrale aandachtspunten van het stageproject.

3.4 ONDERHOUDBAARHEIDSPROBLEMEN

De meeste logica van de QC-toolbox was ondergebracht in een centrale bibliotheek die doorheen de tijd sterk was gegroeid. Verschillende verantwoordelijkheden liepen door elkaar, waardoor het steeds moeilijker werd om te overzien wat welk onderdeel deed.

Dat maakte het aanpassen van bestaande controles of het toevoegen van nieuwe omslachtig. Ontwikkelaars moesten zich eerst door veel code werken voordat een wijziging kon worden doorgevoerd, en de impact van aanpassingen was niet altijd goed in te schatten.

Ook het debuggen was lastig door de beperkte scheiding tussen de verschillende onderdelen. Logisch van elkaar te onderscheiden functionaliteiten waren vaak sterk met elkaar verweven, wat de complexiteit deed toenemen naarmate het systeem verder groeide.

Een herwerking van de architectuur was nodig om de verdere ontwikkeling van de toolbox mogelijk te maken.

3.5 UI- EN USABILITYPROBLEMEN

Naast de technische beperkingen waren er ook aandachtspunten op het vlak van gebruikservaring. De toolbox bevatte veel informatie, maar de presentatie ervan liet op sommige punten te wensen over.

Op de overzichtspagina was het niet altijd eenvoudig om snel een beeld te krijgen van de resultaten van de verschillende controles. Bepaalde visuele elementen die de interpretatie konden vergemakkelijken, ontbraken, waardoor gebruikers soms extra stappen moesten zetten om de nodige context te vinden.

De resultaatpagina's hadden ook ruimte voor verbetering. Bij sommige controles was het nuttig om naast het betrokken taxon ook aanvullende informatie te tonen, zoals een suggestie voor een correcte naam wanneer een naamprobleem werd gedetecteerd. In andere gevallen, zoals bij controles die nagaan of een attribuut correct wordt doorgegeven van een oudertaxon naar zijn kinderen, was een gegroepeerde weergave per oudertaxon overzichtelijker dan een vlakke lijst van afzonderlijke resultaten. Bovendien verschilden de beschikbare filters van pagina tot pagina, wat de consistentie van de gebruikservaring niet ten goede kwam.

Deze aandachtspunten stonden een werkende toolbox niet in de weg, maar boden wel concrete aanknopingspunten om de efficiëntie voor de eindgebruikers te verhogen.

3.6 DOELSTELLINGEN VAN HET STAGEPROJECT

Uit de analyse van de bestaande situatie kwamen drie doelstellingen naar voren. De eerste was de onderhoudbaarheid verbeteren door de architectuur te herwerken en een duidelijkere scheiding van verantwoordelijkheden aan te brengen.

Het tweede aandachtspunt was performantie. De focus lag op het analyseren en optimaliseren van SQL-queries, zodat resultaten sneller gegenereerd konden worden en het systeem beter schaalbaar werd.

Tot slot was er de gebruikservaring. Aanpassingen aan de interface en de presentatie van resultaten moesten de toolbox overzichtelijker en efficiënter maken voor de eindgebruikers. Samen moesten deze verbeteringen de QC-toolbox omvormen tot een onderhoudbaar, performant en bruikbaar hulpmiddel voor het bewaken van de datakwaliteit binnen Aphia.

4 DATABASESTRUCTUUR EN TECHNISCHE CONTEXT

4.1 RELATIONELE STRUCTUUR VAN APHIA

Aphia is opgebouwd als een relationele databank voor het opslaan van taxonomische gegevens over mariene organismen. Naast informatie over individuele taxa bevat de databank ook de relaties tussen taxa en verschillende ondersteunende gegevensbronnen.

Taxa staan centraal in die structuur. Een taxon kan elke rang in de taxonomische hiërarchie voorstellen, van rijk tot variëteit, en is vaak verbonden met andere taxa via hiërarchische relaties. Daarnaast kunnen taxa gekoppeld zijn aan bronvermeldingen, type specimens, synoniemen en andere taxonomische informatie.

Het opslaan van die informatie in afzonderlijke maar onderling verbonden tabellen beperkt duplicatie en houdt de databank flexibel uitbreidbaar. Tegelijkertijd maakt deze structuur het opvragen van informatie complex, omdat dit vaak meerdere relaties en tabellen omvat.

4.2 DATABASENORMALISATIE

De structuur van Aphia is sterk genormaliseerd. Gegevens worden opgesplitst over verschillende tabellen om redundantie te beperken en de consistentie te verbeteren. Wanneer informatie slechts op één plaats wordt opgeslagen, verkleint het risico op tegenstrijdige of verouderde gegevens.

Normalisatie heeft voordelen op het vlak van datakwaliteit en onderhoudbaarheid, maar verhoogt ook de complexiteit van SQL-queries. Om alle benodigde informatie samen te brengen, moeten gegevens uit meerdere tabellen worden gecombineerd met behulp van joins.

Voor de kwaliteitscontroles binnen Aphia betekende dit dat bepaalde queries een groot aantal tabellen moesten raadplegen om een volledig resultaat te kunnen samenstellen. De impact hiervan op de performantie wordt verder besproken in hoofdstuk 6.

4.3 QC-RELEVANTE DATASTRUCTUREN

Naast de algemene taxonomische gegevens bevat Aphia ook datastructuren die specifiek door de QC-toolbox worden gebruikt. Deze ondersteunen het definiëren, uitvoeren en opslaan van kwaliteitscontroles.

Tabel	Doel
qc_queries	Bevat de definities van de beschikbare kwaliteitscontroles en de bijhorende configuratie.
qc_queries_scope	Koppelt kwaliteitscontroles aan specifieke rijken, zodat controles die gebonden zijn aan een bepaalde nomenclatorische code enkel worden toegepast op de relevante taxa.
qc_cache	Slaat de resultaten van geselecteerde kwaliteitscontroles tijdelijk op om herhaalde berekeningen te vermijden.
qc_cache_meta	Bevat metadata over de opbouw en geldigheid van gecachte resultaten.

Tabel 4.1: Kwaliteitscontrole tabellen

Deze tabellen lagen aan de basis van de cachingstrategie die tijdens het project werd uitgewerkt. De werking ervan wordt verder besproken in hoofdstuk 7.

4.4 COMPLEXITEIT VAN QUERIES

De kwaliteitscontroles binnen Aphia gebruikten SQL-queries om potentiële problemen op te sporen. Hoewel de onderliggende controles vaak een duidelijk doel hadden, vereiste de uitvoering ervan regelmatig het combineren van gegevens uit verschillende tabellen.

Door de genormaliseerde structuur van de databank moesten queries informatie ophalen uit meerdere gerelateerde tabellen. Relaties tussen taxa, bronvermeldingen, classificaties en andere taxonomische gegevens werden gecombineerd om een volledig resultaat samen te stellen, waardoor bepaalde controles een aanzienlijk aantal joins bevatten.

Naast de databankstructuur verhoogde ook het gebruik van editrights de complexiteit. Afhankelijk van de rechten van de gebruiker moesten resultaten worden gefilterd zodat enkel relevante gegevens zichtbaar waren. Het volstond dus niet om enkel de datakwaliteitsproblemen te detecteren; ook de toegangsrechten van de gebruiker moesten worden meegenomen.

Sommige controles maakten verder gebruik van aanvullende filters of berekeningen voor een correcte presentatie van resultaten. De combinatie van grote hoeveelheden gegevens, complexe relaties en gebruikersspecifieke filtering belastte de databank aanzienlijk, wat het optimaliseren van deze queries tot een van de voornaamste technische uitdagingen van het project maakte.

Deze combinatie van factoren verklaart waarom verschillende kwaliteitscontroles een aanzienlijke uitvoeringstijd konden hebben en waarom een grondige analyse van de onderliggende queries noodzakelijk was.

5 ARCHITECTURALE HERWERKING

5.1 ANALYSE VAN DE OORSPRONKELIJKE STRUCTUUR

Bij aanvang van de stage was de QC-toolbox opgebouwd rond een sterk gecentraliseerde architectuur. Het grootste deel van de functionaliteit zat in één centrale bibliotheek, *QcLib.php*, die doorheen de jaren organisch was gegroeid naarmate nieuwe kwaliteitscontroles werden toegevoegd. Die bibliotheek bevatte de implementatie van individuele kwaliteitscontroles én ondersteunende functionaliteiten zoals filtering, paginering, permissiebeheer en diverse helperfuncties.

De bibliotheek telde meer dan zeshonderd regels code met tientallen functies met uiteenlopende verantwoordelijkheden. Wijzigingen doorvoeren zonder onbedoelde neveneffecten was daardoor moeilijk, en ook het terugvinden van de juiste functionaliteit vergde steeds meer tijd.

Naast *QcLib.php* was de algemene structuur van de QC-module relatief plat. De scheiding tussen dataverwerking, businesslogica en presentatie was niet altijd duidelijk, waardoor de codebase minder schaalbaar werd naarmate de toolbox verder groeide.

Die situatie was de aanleiding voor een architecturale herwerking. Het doel was de leesbaarheid te verbeteren en tegelijk een structuur te creëren die toekomstige uitbreidingen en optimalisaties beter aankon.

5.2 DOELSTELLINGEN VAN DE REFACTOR

Op basis van de analyse van de bestaande structuur werden vier doelstellingen voor de refactor geformuleerd.

De eerste was onderhoudbaarheid. Door verantwoordelijkheden duidelijker te scheiden en functionaliteit modulair te organiseren, moest het eenvoudiger worden om code te begrijpen, aan te passen en uit te breiden [4].

Schaalbaarheid was het tweede aandachtspunt. Tijdens de stage werden naast optimalisaties ook nieuwe kwaliteitscontroles ontwikkeld. De architectuur moest flexibel genoeg zijn om die uitbreidingen op te vangen zonder de complexiteit opnieuw te laten oplopen.

Een derde doelstelling was een betere scheiding tussen dataverwerking, businesslogica en presentatie [5]. Die ontkoppeling maakt het mogelijk om wijzigingen aan één onderdeel door te voeren zonder andere onderdelen te beïnvloeden, wat ook het testen en debuggen vereenvoudigt.

Tot slot moest de nieuwe architectuur een goede basis vormen voor de performantie- en cachingverbeteringen die in de volgende hoofdstukken aan bod komen.

5.3 MODULAIRE QUERYSTRUCTUUR

Om de onderhoudbaarheid te verbeteren, werd de gecentraliseerde structuur van `QcLib.php` vervangen door een modulaire opbouw. Kwaliteitscontroles die voorheen vanuit één bibliotheek werden beheerd, werden opgesplitst over gespecialiseerde componenten op basis van hun functionele verantwoordelijkheid.

Die opsplitsing maakte het eenvoudiger om gerelateerde functionaliteit terug te vinden en wijzigingen door te voeren zonder andere onderdelen te raken. Ook de projectstructuur werd overzichtelijker, waardoor nieuwe ontwikkelaars sneller inzicht krijgen in de werking van de toolbox.

Aspect	Oorspronkelijke architectuur	Nieuwe architectuur
Structuur	Sterk gecentraliseerd	Modulair opgebouwd
Hoofdcomponent	`QcLib.php`	Gespecialiseerde componenten
Verantwoordelijkheden	Vermengd	Duidelijk gescheiden
Onderhoudbaarheid	Beperkt	Verbeterd
Uitbreidbaarheid	Moeilijker	Eenvoudiger
Overzichtelijkheid	Lagere leesbaarheid	Betere structuur

Tabel 5.1: Vergelijking tussen de oorspronkelijke en de gerefactorde architectuur

Hetzelfde principe werd toegepast op ondersteunende functionaliteit. De nieuwe structuur voorziet afzonderlijke componenten voor onder meer gegevensopvraging, presentatie, exportfunctionaliteit, caching en ondersteunende infrastructuur. Hierdoor namen de afhankelijkheden tussen onderdelen af en werden verantwoordelijkheden duidelijker gescheiden.

Die modulaire opbouw was de basis voor de verdere optimalisaties die tijdens de stage werden uitgevoerd. Een gestructureerde codebase maakt het eenvoudiger om bestaande controles uit te breiden of nieuwe toe te voegen zonder de complexiteit te laten oplopen.

5.4 VERBETERDE VERANTWOORDELIJKHEDEN EN FLOW

Na de modulaire herstructurering werd ook de interne dataflow van de applicatie verduidelijkt. Componenten voor het ophalen en verwerken van gegevens werden losgekoppeld van de presentatielaag, waardoor wijzigingen aan de interface mogelijk zijn zonder de onderliggende logica aan te raken, en omgekeerd.

Component	Verantwoordelijkheid
Querycomponenten	Uitvoeren van kwaliteitscontroles
Cachingcomponenten	Beheer van snapshots en cachevalidatie
Presentatielaag	Weergave van resultaten
Exportfunctionaliteit	Exporteren van resultaten
Ondersteunende infrastructuur	Filtering, rechtenbeheer en hulpfuncties

Tabel 5.2: Voorbeeld van de verantwoordelijkheidsverdeling binnen de nieuwe architectuur

De flow verloopt nu in duidelijke stappen: gegevens worden opgehaald en verwerkt door gespecialiseerde componenten, waarna de resultaten worden voorbereid voor presentatie en via de gebruikersinterface weergegeven. Die gelaagdheid maakt het eenvoudiger om problemen te lokaliseren tijdens het debuggen of verder ontwikkelen.

Die expliciete afbakening van verantwoordelijkheden maakte de codebase overzichtelijker en legde tegelijk de basis voor de optimalisaties die in de volgende hoofdstukken aan bod komen.

5.5 VOORDELEN VAN DE NIEUWE ARCHITECTUUR

De refactor had direct zichtbare gevolgen voor de dagelijkse werking met de codebase.

Wijzigingen aan een specifiek onderdeel vereisen niet langer een analyse van de volledige bibliotheek. Doordat functionaliteit is gegroepeerd per verantwoordelijkheid, is de impact van een aanpassing beter in te schatten en neemt de kans op onbedoelde neveneffecten af.

De leesbaarheid verbeterde mee. Wie de werking van een specifieke kwaliteitscontrole wil begrijpen, hoeft niet meer door de volledige *QcLib.php* te navigeren. Dat maakt ook kennisoverdracht binnen het team eenvoudiger.

Nieuwe controles en optimalisaties liggen binnen de nieuwe structuur makkelijker binnen bereik. Er is een duidelijk aanknopingspunt voor uitbreidingen, zonder dat fundamentele wijzigingen aan de architectuur nodig zijn.

Ook het debuggen van problemen werd eenvoudiger. Door de duidelijkere scheiding tussen componenten kan sneller worden vastgesteld waar een fout ontstaat en welke onderdelen hierbij betrokken zijn.

5.6 MOGELIJKHEDEN VOOR TOEKOMSTIGE UITBREIDING

Naast het oplossen van bestaande problemen schept de nieuwe architectuur ook ruimte voor toekomstige uitbreidingen. Door verantwoordelijkheden duidelijk af te bakenen en functionaliteit modulair op te bouwen, heeft de QC-toolbox een flexibeler fundament voor verdere ontwikkeling.

Nieuwe kwaliteitscontroles kunnen eenvoudiger worden geïntegreerd omdat ontwikkelaars zich kunnen focussen op de implementatie van de controle zelf, zonder bestaande functionaliteit ingrijpend te moeten aanpassen. Dat verlaagt de ontwikkelkost en vermindert de kans op regressies.

De betere scheiding tussen componenten biedt ook ruimte voor gerichte performantieverbeteringen. Optimalisaties kunnen worden toegepast op specifieke onderdelen zonder de volledige workflow te beïnvloeden. Denk aan bijkomende cachingmechanismen, verdere query-optimalisaties of geautomatiseerde validatieprocessen.

De toolbox past daarmee beter bij de huidige behoeften van VLIZ en heeft structureel ruimte om mee te groeien naarmate het aantal kwaliteitscontroles toeneemt.

6 SQL-OPTIMALISATIE

6.1 ANALYSE VAN BESTAANDE QUERIES

De QC-toolbox bevat een verzameling kwaliteitscontroles die elk specifieke inconsistenties of ontbrekende gegevens in de Aphia-database detecteren. Een deel van die controles liep snel, maar andere queries bleken aanzienlijk meer verwerkingstijd te vereisen.

Uit de analyse bleek dat verschillende kwaliteitscontroles grote hoeveelheden data moesten verwerken via complexe joins over meerdere tabellen. Controles die afhankelijk waren van taxonomische relaties, brongegevens of hiërarchische structuren hadden daarbij de grootste impact op de uitvoeringstijd.

Benchmarkmetingen maakten de omvang van het probleem concreet: sommige queries hadden meerdere seconden nodig om resultaten te genereren, in uitzonderlijke gevallen zelfs meer dan twee minuten. Dat had gevolgen voor zowel de gebruikerservaring als de schaalbaarheid van de toolbox wanneer meerdere controles gelijktijdig werden uitgevoerd.

Die vaststellingen waren de aanleiding voor een systematische analyse van de performantie van de bestaande queries en de ontwikkeling van gerichte optimalisaties.

6.2 GEBRUIK VAN EXECUTION PLANS

Om de oorzaken van de performantieproblemen te achterhalen, werden de execution plans geanalyseerd die door Microsoft SQL Server worden gegenereerd. Die plans geven inzicht in hoe de database-engine een query uitvoert en wijzen kostelijke operaties aan [6] [7].

Bij de zwaarste kwaliteitscontroles kwamen verschillende bottlenecks naar voren. In meerdere gevallen werden grote hoeveelheden data verwerkt voordat relevante records werden gefilterd. Complexe joins, uitgebreide scans en herhaalde berekeningen kwamen regelmatig terug als oorzaken van hoge uitvoeringstijden.

Die bevindingen waren de basis voor de verdere optimalisaties. In plaats van uitsluitend individuele SQL-statements te optimaliseren, werd gekeken naar de volledige queryverwerking en de hoeveelheid data die bij elke stap werd meegenomen.

Dat leverde een algemene optimalisatiestrategie op waarbij het beperken van de verwerkte dataset zo vroeg mogelijk in de query centraal stond.

6.3 OPTIMALISATIESTRATEGIE

Het centrale probleem dat de optimalisatiestrategie moest oplossen was de volgorde van bewerkingen in bestaande queries. Complexe joins werden vaak uitgevoerd op grote, ongefilterde datasets, waarna pas later in de query relevante records werden geselecteerd. De database-engine verwerkte zo aanzienlijke hoeveelheden data die uiteindelijk niet bijdroegen aan het eindresultaat.

De strategie was gericht op het omdraaien van die volgorde: relevante records zo vroeg mogelijk beperken, zodat latere stappen op een kleinere dataset konden worden uitgevoerd. Dat verlaagde de belasting van joins, sorteringen en andere kostelijke operaties [6].

Aanvullend werd gezocht naar mogelijkheden om herhaalde berekeningen te vermijden en tussentijdse resultaten te hergebruiken, waardoor de totale hoeveelheid werk die de database moest uitvoeren verder afnam.

6.4 TEMPORARY TABLES EN INDEXERING

De optimalisatiestrategie werd toegepast op verschillende kwaliteitscontroles, maar kwam het duidelijkst tot uiting in de controle *authorityAndSourceNotMatching*. Die query combineerde taxonomische relaties, brongegevens en meerdere validatiestappen binnen één grote querystructuur, wat complexe execution plans opleverde waarbij dezelfde tussenresultaten meermaals verwerkt konden worden.

De voornaamste aanpassing was het opsplitsen van dergelijke queries in afzonderlijke verwerkingsstappen met behulp van tijdelijke tabellen. Waar de oorspronkelijke implementatie voornamelijk gebruikmaakte van Common Table Expressions (CTE's), werden de zwaarste delen gematerialiseerd in tijdelijke tabellen. Tussentijdse resultaten werden zo expliciet opgeslagen en konden hergebruikt worden zonder dat SQL Server dezelfde berekeningen opnieuw moest uitvoeren.

Die aanpak maakte het mogelijk om grote datasets stapsgewijs te beperken tot een relevante kandidaatset voordat verdere verwerking plaatsvond. De duidelijke scheiding tussen verwerkingsfasen hielp SQL Server ook om eenvoudigere uitvoeringsplannen te genereren. Daarnaast konden gerichte indexen worden toegevoegd op tussentijdse resultaten die in latere stappen opnieuw gebruikt werden.

Indexering werd ook breder toegepast. Door indexen toe te voegen op veelgebruikte zoek- en joinkolommen konden vervolgoperaties efficiënter worden uitgevoerd, wat vooral relevant was voor controles met meerdere validatiestappen op dezelfde tussentijdse datasets [8].

Daarnaast werd de volgorde van bewerkingen herzien: filters zo vroeg mogelijk, dure operaties pas nadat datasets voldoende waren verkleind. Verschillende queries werden ook herschreven om herhaalde berekeningen te vermijden.

De combinatie van tijdelijke tabellen, indexering, vroegtijdige filtering en herschreven querylogica was de basis van de performantieverbeteringen die in de volgende sectie worden besproken.

6.5 RUNTIMEVERBETERINGEN

De optimalisaties hadden een duidelijke impact. Vooral de zwaarste queries gingen er sterk op vooruit.

Kwaliteitscontrole	Oorspronkelijk	Geoptimaliseerd
authorityAndSourceNotMatching	~108 s	~20 s
DisplaynameOutOfSync	>120 s	~20 s
CompareValues	~12,5 s	<1 s
DeletedParent	~12 s	<1 s

Tabel 6.1: Voorbeelden van runtimeverbeteringen

De resultaten in Tabel 6.1 tonen aan dat vooral de zwaarste kwaliteitscontroles sterk profiteerden van de uitgevoerde optimalisaties. Een opvallend voorbeeld is de controle voor taxa waarvan de displaynaam niet synchroon loopt met de onderliggende taxonomische gegevens.

Andere controles vertoonden vergelijkbare verbeteringen. De query voor het vergelijken van waarden daalde van ongeveer 12,6 seconden naar minder dan één seconde, en de controle voor taxa met een verwijderde parentrelatie van bijna twaalf seconden naar minder dan één seconde.

Deze resultaten bevestigen dat vroegtijdige filtering, tijdelijke datasets, indexering en herschreven querylogica samen een effectieve aanpak vormen voor performantieoptimalisatie. Bovendien maakten deze verbeteringen het mogelijk om caching gerichter toe te passen op de kwaliteitscontroles die na optimalisatie nog steeds een hoge uitvoeringstijd hadden.

6.6 TRADE-OFFS EN BEPERKINGEN

De optimalisaties leverden aanzienlijke winst op, maar brachten ook bijkomende complexiteit met zich mee. Verschillende queries werden uitgebreider door het gebruik van tijdelijke tabellen, bijkomende filters en tussentijdse verwerkingsstappen.

Hierdoor ontstaat een afweging tussen performantie en leesbaarheid. Een sterk geoptimaliseerde query is niet noodzakelijk eenvoudiger te begrijpen of te onderhouden dan een rechtlijnigere implementatie. Tijdens de optimalisaties werd daarom gezocht naar een evenwicht tussen beide.

Performantiemetingen blijven bovendien afhankelijk van de omvang en samenstelling van de dataset. Naarmate de Aphia-database verder groeit, kunnen nieuwe bottlenecks ontstaan die bijkomende optimalisaties vereisen.

De behaalde resultaten tonen aan dat de gekozen aanpak werkte, maar verdere tuning zal nodig blijven naarmate de databank evolueert.

7 CACHINGSTRATEGIEËN

7.1 MOTIVATIE VOOR CACHING

Ondanks de query-optimalisaties uit het vorige hoofdstuk bleven verschillende kwaliteitscontroles zware database-operaties uitvoeren. Vooral controles die grote hoeveelheden data moesten verwerken of complexe taxonomische relaties analyseerden, vereisten nog steeds meerdere seconden verwerkingstijd.

Veel kwaliteitscontroles veranderen bovendien niet voortdurend en worden regelmatig door verschillende gebruikers geraadpleegd. Dat maakt het mogelijk om eerder berekende resultaten tijdelijk op te slaan en te hergebruiken, zodat dezelfde controle niet telkens opnieuw volledig uitgevoerd hoeft te worden wanneer de onderliggende gegevens ongewijzigd zijn [9].

Caching vermindert zo de belasting van de database en geeft gebruikers sneller toegang tot resultaten, zonder dat de kwaliteitscontrole opnieuw berekend moet worden.

Om die redenen werd caching onderzocht als aanvullende techniek bovenop de reeds uitgevoerde query-optimalisaties. Tijdens de stage werden verschillende cachingstrategieën ontwikkeld en geëvalueerd om te bepalen voor welke kwaliteitscontroles caching een meerwaarde kon bieden. Daarbij werd niet alleen gekeken naar performantieverbeteringen, maar ook naar aspecten zoals actualiteit van gegevens, onderhoudbaarheid en operationele complexiteit.

7.2 BESTANDGEBASEERDE CACHING

De eerste cachingimplementatie sloeg resultaten op als bestanden op het bestandssysteem. Voor elke kwaliteitscontrole werd een lijst van gevonden resultaten bewaard, zodat die niet telkens opnieuw via een volledige query-uitvoering moest worden opgebouwd.

Tijdens de ontwikkeling werden twee benaderingen uitgewerkt. Bij de full-aanpak werden alle opgeslagen identifiers in één keer verwerkt. Bij de windowed-aanpak gebeurde dit in kleinere batches, waardoor de geheugenbelasting beter beheersbaar bleef bij grote datasets.

Om bij te houden of een snapshot nog geldig was, werd per snapshot metadata opgeslagen. Die metadata bevatte onder andere het tijdstip waarop de cache werd opgebouwd en een hash van de querydefinitie waarop de snapshot gebaseerd was. Was een snapshot ouder dan de toegelaten geldigheidsperiode, of was de onderliggende query gewijzigd, dan werd automatisch teruggevallen op een live uitvoering van de kwaliteitscontrole.

De windowed-aanpak beperkte het geheugenverbruik, maar introduceerde nieuwe uitdagingen. Grote snapshots moesten in meerdere batches worden verwerkt, wat een aanzienlijk aantal afzonderlijke database-opvragingen veroorzaakte. Dat verhoogde de belasting van de databank en maakte het performantiegedrag minder voorspelbaar voor omvangrijke kwaliteitscontroles.

Benchmarkresultaten toonden aan dat caching voor bepaalde kwaliteitscontroles aanzienlijke snelheidswinsten opleverde, terwijl andere controles slechts beperkt voordeel uit deze aanpak haalden. Die combinatie van operationele complexiteit, schaalbaarheidsvragen en uiteenlopende benchmarkresultaten was uiteindelijk de aanleiding om een alternatieve, database-gebaseerde cachingstrategie te onderzoeken.

Kwaliteitscontrole	Live query (ms)	Full cache (ms)	Windowed cache (ms)
authorityAndSourceNotMatching	108499	64772	10213
getAcceptedSpeciesWithSameBasionym	1352	1108	1774
compareValues	378	1186	2451
getMissingTypeSubspecies	2437	2423	872
getTaxaWithoutSpecimen	3736	n.v.t. (OOM)	279

Tabel 7.1: Evaluatie van de bestand-gebaseerde cachingstrategieën voor de optimalisatie van de live uitvoering

De benchmarkresultaten tonen aan dat de effectiviteit van file-based caching sterk afhankelijk is van de aard van de kwaliteitscontrole. Voor bepaalde controles leverde caching een aanzienlijke performantieverbetering op, terwijl andere controles weinig voordeel haalden uit deze aanpak. De windowed-strategie bleek bovendien beter geschikt voor omvangrijke datasets doordat geheugenproblemen vermeden konden worden.

Om een eerlijke vergelijking mogelijk te maken, werden de benchmarkmetingen uitgevoerd met identieke filters en vergelijkbare datasets. Hierdoor kunnen de gemeten verschillen toegeschreven worden aan de cachingstrategie zelf en niet aan verschillen in de omvang van de verwerkte gegevens.

7.3 DATABASEGEBASEERDE CACHING

Op basis van de ervaringen met de file-based aanpak werd een alternatieve cachingstrategie ontwikkeld waarbij snapshots rechtstreeks in de databank worden opgeslagen. In plaats van resultaten als afzonderlijke bestanden te bewaren, worden gecachte resultaten centraal beheerd via specifieke databasetabellen.

Voor elke kwaliteitscontrole wordt een snapshot opgebouwd die de relevante identifiers bevat. Metadata van die snapshot, waaronder het opbouwtijdstip en een hash van de querydefinitie, wordt bijgehouden in een afzonderlijke tabel.

Bij het opvragen van een kwaliteitscontrole wordt eerst gecontroleerd of een geldige snapshot beschikbaar is. Is de cache ouder dan de toegelaten geldigheidsperiode, of is de querydefinitie gewijzigd, dan wordt automatisch teruggevallen op een live uitvoering. Zo worden verouderde resultaten vermeden wanneer de onderliggende logica of gegevens zijn aangepast.

Om de presentatie verder te optimaliseren, werd ook de sorteervolgorde in de cache opgenomen. Tijdens het opbouwen van een snapshot krijgt elk record een volgnummer toegewezen dat overeenkomt met de uiteindelijke alfabetische ordening. Gecachte resultaten kunnen zo efficiënt worden opgehaald en gepagineerd zonder dat bij elke aanvraag opnieuw complexe sorteerstappen nodig zijn.

Door snapshots centraal in de databank op te slaan werd het beheer van gecachte resultaten vereenvoudigd, nam de afhankelijkheid van het bestandssysteem af en sluit de oplossing beter aan bij de bestaande datagedreven architectuur van de QC-toolbox.

7.4 VERGELIJKING VAN BEIDE BENADERINGEN

Beide cachingstrategieën moesten de uitvoeringstijd van kwaliteitscontroles verminderen door eerder berekende resultaten te hergebruiken. De praktische uitwerking en de bijbehorende eigenschappen verschilden echter aanzienlijk.

De file-based aanpak was een nuttige eerste stap in het onderzoek naar caching binnen de QC-toolbox. Door resultaten lokaal op te slaan konden verschillende kwaliteitscontroles sneller worden weergegeven zonder telkens opnieuw de volledige query uit te voeren.

Toch kwamen bij gebruik van deze aanpak ook beperkingen naar voren. Vooral bij grote datasets leidde de verwerking van snapshots tot uitdagingen op het vlak van geheugenbeheer en databasebelasting. De windowed-aanpak beperkte weliswaar het geheugenverbruik, maar vereiste een groot aantal afzonderlijke database-opvragingen om resultaten in batches te verwerken.

De database-gebaseerde aanpak loste een deel van deze problemen op door snapshots centraal binnen de databank op te slaan. Resultaten konden efficiënter worden opgehaald en gepagineerd, en de integratie met bestaande databasefunctionaliteit werd eenvoudiger omdat zowel gegevens als cache zich binnen dezelfde omgeving bevinden.

Daarnaast werd ook het beheer van metadata, zoals opbouwtijdstippen, queryversies en sorteervolgordes, centraal en uniform georganiseerd. De oplossing werd daarmee schaalbaarder en sloot beter aan bij de verdere ontwikkeling van de QC-toolbox.

Aspect	**Bestand-gebaseerde caching**	**Database-gebaseerde caching**
Opslaglocatie	Bestanden op het bestandssysteem	Tabellen binnen de databank
Beheer van metadata	Verspreid over bestanden	Centraal via *qc_cache_meta*
Schaalbaarheid	Beperkt bij grote datasets	Beter geschikt voor grote datasets
Geheugenverbruik	Lager met de windowed-aanpak	Centraal beheerd binnen de databank
Databasebelasting	Meer afzonderlijke database-opvragingen	Minder database round-trips
Paginering en sortering	Dynamisch tijdens verwerking	Vooraf berekend en opgeslagen
Onderhoudbaarheid	Hogere operationele complexiteit	Eenvoudiger centraal beheer
Integratie met bestaande architectuur	Beperkt	Sterk geïntegreerd

Tabel 7.2: Vergelijking van de eigenschappen van de bestand-gebaseerde en database-gebaseerde cachingstrategieën

Beide benaderingen bleken bruikbaar, maar de database-gebaseerde implementatie bood uiteindelijk de beste balans tussen performantie, onderhoudbaarheid en schaalbaarheid.

7.5 RESULTATEN

De benchmarkresultaten tonen aan dat caching vooral een meerwaarde biedt voor kwaliteitscontroles met een hoge uitvoeringstijd. Voor verschillende controles werden aanzienlijke performantieverbeteringen gerealiseerd doordat complexe berekeningen niet bij elke aanvraag opnieuw uitgevoerd hoefden te worden.

Kwaliteitscontrole	Live (ms)	Cache (ms)	Verschil (%)
authorityAndSourceNotMatching	18910	1601	-91,5%
getTaxaWithDisplaynameOutOfSync	20200	671	-96,7%
getTaxaWithoutTypeOfLocality	5305	723	-86,4%
getSynonymsWithDifferentParents	2517	396	-84,3%
getAcceptedSpeciesWithSameBasionym	455	10	-97,8%
getAttributesWithUnknownGazetteer	581	962	+65,6%

Tabel 7.3: Representatieve benchmarkresultaten van database-gebaseerde caching ten opzichte van de geoptimaliseerde live uitvoering

De effectiviteit van caching bleek echter sterk afhankelijk van de aard van de onderliggende query. Voor bepaalde kwaliteitscontroles waren de uitvoeringstijden na de query-optimalisaties al voldoende laag, waardoor de meerwaarde van caching beperkt bleef. In dergelijke gevallen werd gekozen voor een rechtstreekse uitvoering tegen de databank, zodat gebruikers steeds over de meest actuele resultaten beschikken.

De uiteindelijke implementatie combineert daarom beide benaderingen. Zwaardere kwaliteitscontroles gebruiken caching om de responstijd te verbeteren en de belasting van de databank te verminderen, terwijl lichtere controles live uitgevoerd blijven worden wanneer de voordelen van actuele gegevens zwaarder doorwegen dan de performantievoordelen van caching.

7.6 MOGELIJKE TOEKOMSTIGE VERBETERINGEN

Hoewel de cachingstrategieën aanzienlijke performantieverbeteringen opleverden, blijven er aandachtspunten. Beide aanpakken vereisen een periodieke vernieuwing van de cache om resultaten actueel te houden. Dat creëert steeds een afweging tussen performantie en de snelheid waarmee wijzigingen in de onderliggende gegevens zichtbaar worden.

Bepaalde kwaliteitscontroles blijven bovendien relatief duur om op te bouwen. Gecachte resultaten zijn snel op te halen, maar de initiële opbouw van een snapshot vraagt nog steeds rekentijd. Voor controles met complexe taxonomische relaties of grote datasets kan dat de cache-vernieuwing merkbaar vertragen.

Een ander aandachtspunt is de opslag van snapshots zelf. Naarmate het aantal kwaliteitscontroles en de omvang van de datasets toenemen, groeit ook de hoeveelheid opgeslagen cachegegevens. Binnen de huidige omgeving is dit geen probleem, maar op langere termijn kunnen bijkomende optimalisaties nodig zijn op het vlak van opslagbeheer en retentiebeleid.

Voor toekomstige ontwikkelingen zijn verschillende verbeteringen mogelijk. Een voor de hand liggende verbetering is het selectief vernieuwen van caches op basis van wijzigingen in de onderliggende gegevens,

in plaats van periodiek alle snapshots opnieuw op te bouwen. Verdere optimalisaties van individuele kwaliteitscontroles kunnen ook de tijd die nodig is om snapshots op te bouwen verder terugdringen.

Hoewel verdere optimalisaties mogelijk blijven, tonen de resultaten aan dat caching een effectieve techniek is om de performantie van kwaliteitscontroles te verbeteren.

8 FUNCTIONELE EN UI-VERBETERINGEN

8.1 ANALYSE VAN DE OORSPRONKELIJKE GEBRUIKERSINTERFACE

De oorspronkelijke gebruikersinterface van de QC-toolbox was functioneel, maar weinig gestructureerd. Naarmate er meer kwaliteitscontroles werden toegevoegd, groeiden de overzichtspagina's uit tot lange verticale lijsten die weinig houvast boden. Een specifieke controle terugvinden vergde meer scrollen dan nodig, en de beschikbare schermruimte werd daarbij niet optimaal benut.

Ook de resultatenpagina's boden weinig context. Resultaten verschenen als eenvoudige lijsten van records, zonder duidelijke toelichting over waarom een bepaald record erin voorkwam. De interface gaf daarbij weinig inzicht in de aard van het probleem of de toegepaste filters, waardoor gebruikers bijkomende stappen moesten zetten om dat te achterhalen.

Dat was de aanleiding om verschillende onderdelen van de interface te herzien, met focus op overzichtelijkheid en begrijpelijkheid.

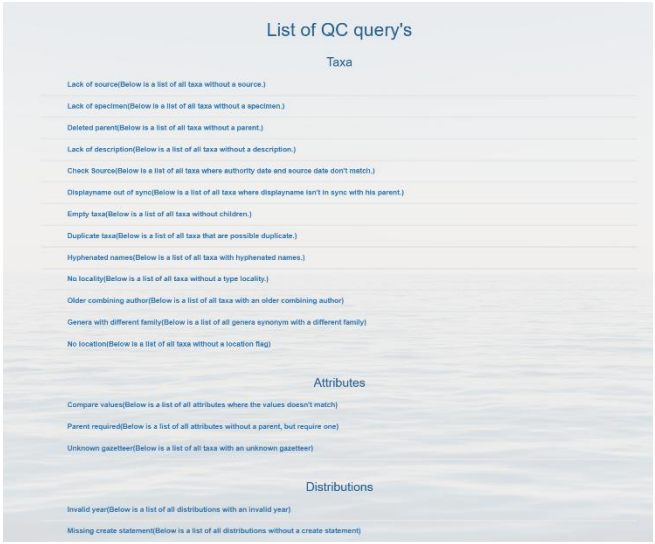
8.2 DOELSTELLINGEN VAN DE HERWERKING

De herwerking richtte zich op een betere gebruikerservaring zonder de bestaande functionaliteit te wijzigen. De nadruk lag op een duidelijkere presentatie van zowel de overzichtspagina's als de resultaten zelf.

Op de overzichtspagina's moest meer structuur komen. Door kwaliteitscontroles logisch te groeperen en de schermruimte beter te benutten kon het eenvoudiger worden om een specifieke controle terug te vinden.

Voor de resultatenpagina's was het doel gebruikers sneller inzicht te geven in waarom een record in een controle voorkwam, zonder daarvoor meerdere pagina's te moeten doorlopen. Daarnaast moest relevante contextinformatie beter zichtbaar worden: actieve filters, gebruikersspecifieke functionaliteiten en de huidige zoekscope werden daarvoor prominenter weergegeven.

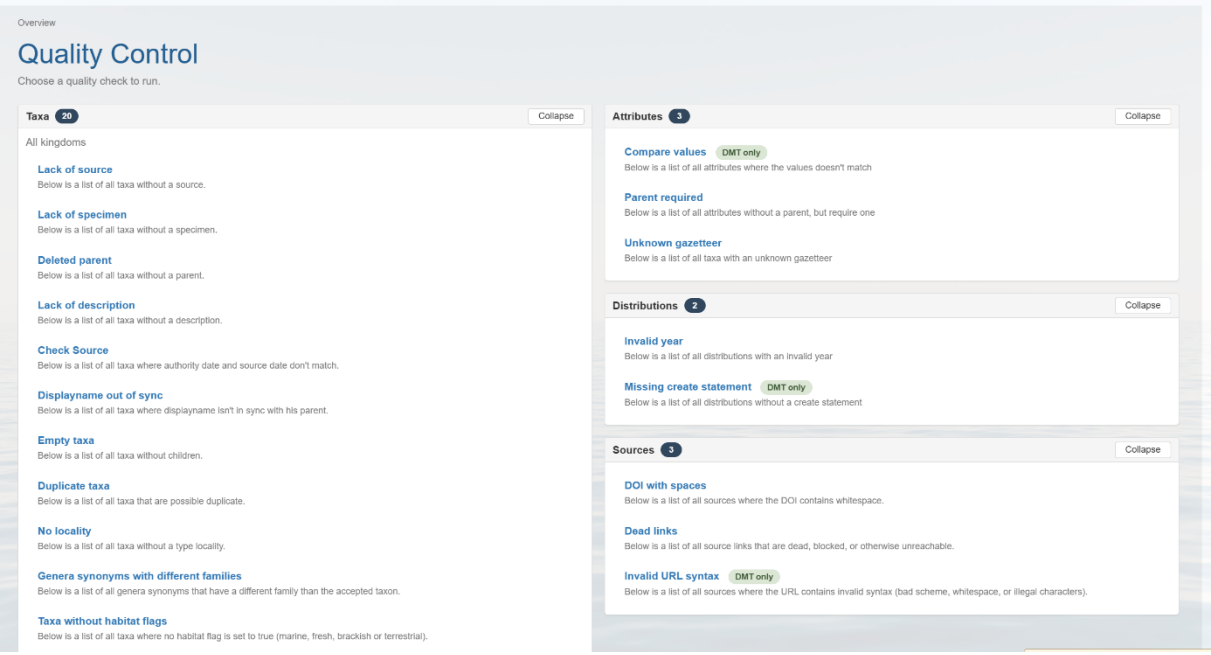
8.3 VERBETERINGEN AAN DE OVERZICHTSPAGINA



Figuur 8.1: Oorspronkelijke overzichtspagina van de QC-toolbox.

De meest zichtbare verandering aan de interface was de herwerking van de overzichtspagina. Alle kwaliteitscontroles stonden oorspronkelijk in één lange lijst. Naarmate het aantal beschikbare controles toenam, werd het moeilijker om snel de gewenste controle terug te vinden.

Om dat te verbeteren werden de controles logisch gegroepeerd per domein: taxonomische controles, attribuutcontroles, distributiecontroles en broncontroles worden elk afzonderlijk weergegeven. Zo kunnen gebruikers zich sneller richten op het deel van de toolbox dat voor hun werk relevant is.



Figuur 8.2: Vernieuwde overzichtspagina met gegroepeerde en inklapbare secties.

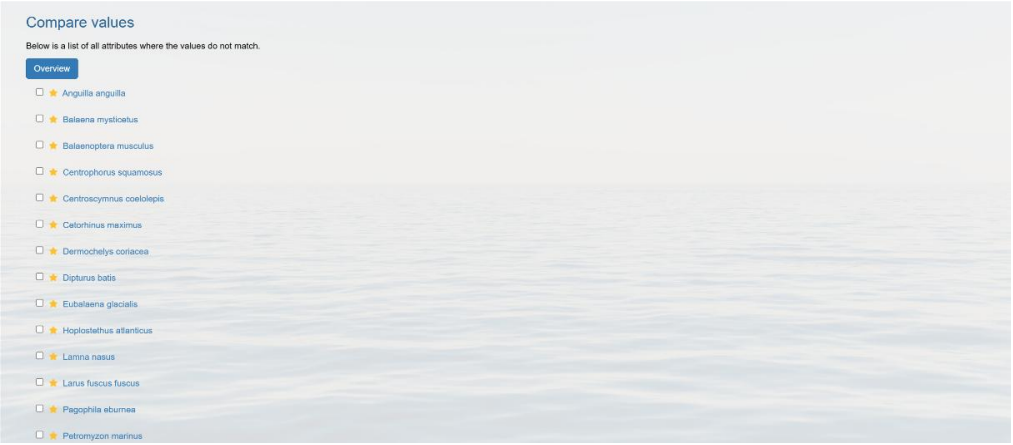
De indeling werd ook aangepast naar een masonry-layout, zodat de beschikbare horizontale schermruimte beter benut wordt. In plaats van één verticale lijst worden de categorieën naast elkaar weergegeven, wat het geheel compacter maakt.

Daarnaast zijn de secties inklapbaar. Gebruikers kunnen categorieën openen of sluiten naargelang hun behoeften, waardoor de hoeveelheid zichtbare informatie beter beheersbaar blijft. Dat is vooral handig voor wie slechts met een beperkt aantal controles werkt.

Kwaliteitscontroles die uitsluitend beschikbaar zijn voor DMT-gebruikers kregen een duidelijke visuele markering, zodat meteen zichtbaar is welke controles enkel voor die groep toegankelijk zijn, zonder dat dit ten koste gaat van de overzichtelijkheid van de pagina.

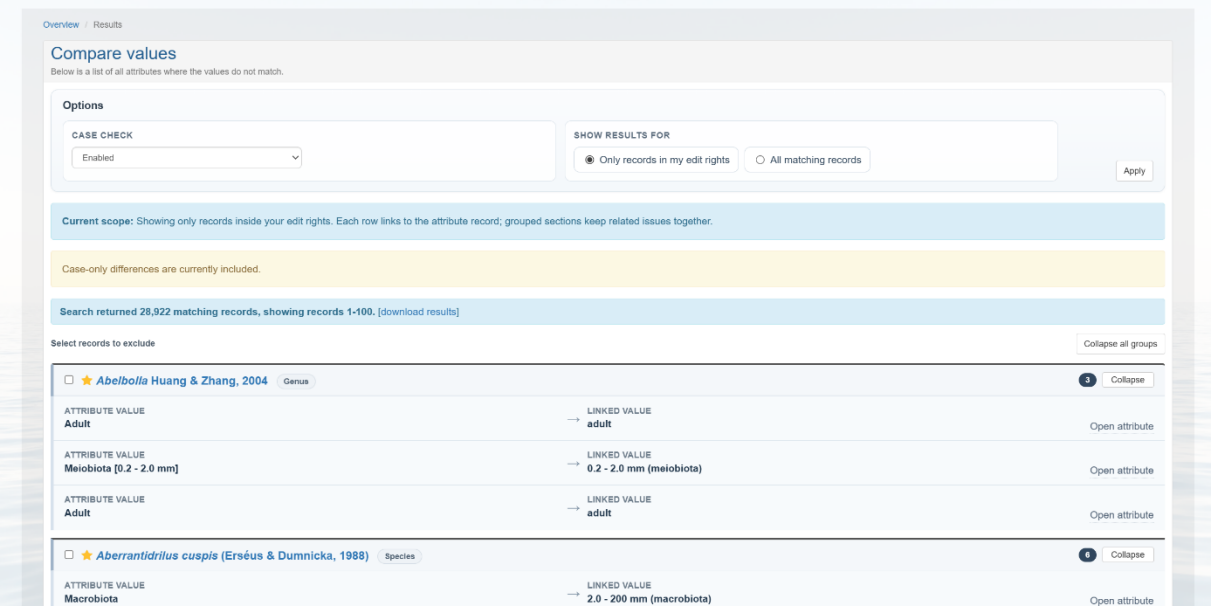
8.4 VERBETERINGEN AAN DE RESULTATENWEERGAVE

De herwerking ging verder dan de overzichtspagina. Ook de resultatenpagina's werden aangepast om gebruikers sneller inzicht te geven in de aard van de gevonden kwaliteitsproblemen.



Figuur 8.3: Oorspronkelijke resultatenweergave van de kwaliteitscontrole Compare Values, waarbij resultaten als een eenvoudige lijst van records worden weergegeven.

In de oorspronkelijke implementatie werden resultaten voornamelijk weergegeven als eenvoudige lijsten van records. Gebruikers konden wel zien welke records door een kwaliteitscontrole werden gedetecteerd, maar moesten vaak bijkomende pagina's openen om te achterhalen waarom een bepaald record in de resultaten voorkwam.



Figuur 8.4: Vernieuwde resultatenweergave van de kwaliteitscontrole Compare Values met bijkomende contextinformatie, filtering en gegroepeerde resultaten.

Bij de vernieuwde weergave wordt extra context rechtstreeks op de pagina getoond. Afhankelijk van de kwaliteitscontrole zijn de onderliggende verschillen of validatieproblemen meteen zichtbaar, waardoor gebruikers sneller kunnen beoordelen welke actie vereist is.

Daarnaast werden verschillende ondersteunende elementen toegevoegd. Gebruikers krijgen informatie over de actieve zoekscope, beschikbare filters en het aantal gevonden resultaten, zodat duidelijker wordt welke gegevens worden weergegeven en welke beperkingen of filters momenteel actief zijn.

Ook de navigatie werd verbeterd. Via breadcrumbs kunnen gebruikers eenvoudig terugkeren naar de overzichtspagina of eerdere stappen in de workflow, zonder telkens opnieuw het volledige pad te moeten doorlopen.

8.5 CONTEXTUELE INFORMATIE EN FILTERING

Tijdens de herwerking werd niet alleen aandacht besteed aan de visuele presentatie van resultaten, maar ook aan de context waarin die resultaten worden weergegeven. Gebruikers moeten kunnen begrijpen welke gegevens op een pagina zichtbaar zijn en welke voorwaarden daarbij van toepassing zijn.

Daarvoor werden verschillende contextuele elementen toegevoegd aan de resultatenpagina's. Zo wordt aangegeven welke zoekscope actief is en of resultaten beperkt worden tot records binnen de bewerkingsrechten van de gebruiker, wat meer transparantie biedt over de herkomst en selectie van de weergegeven gegevens.

Daarnaast werden filters geïntegreerd waarmee gebruikers de resultaten verder kunnen verfijnen. Afhankelijk van de kwaliteitscontrole kunnen specifieke opties worden geselecteerd, waardoor relevante records sneller te identificeren zijn zonder grote hoeveelheden resultaten handmatig te doorzoeken.

Het totaal aantal gevonden resultaten wordt ook zichtbaar weergegeven. Dat geeft gebruikers meteen inzicht in de omvang van een kwaliteitsprobleem en helpt inschatten welke inspanning nodig is om de gevonden records te verwerken.

Aanvullende meldingen informeren gebruikers bovendien over actieve filters, geselecteerde instellingen of andere relevante omstandigheden die invloed hebben op de weergegeven resultaten. Dat vermindert het risico op misinterpretaties en maakt de werking van de kwaliteitscontroles beter begrijpbaar.

8.6 NIEUWE KWALITEITSCONTROLES EN FUNCTIONELE UITBREIDING

Naast de technische en visuele verbeteringen werd de functionaliteit van de QC-toolbox tijdens de stage ook inhoudelijk uitgebreid. De oorspronkelijke versie bevatte achttien kwaliteitscontroles, voornamelijk gericht op taxonomische, attribuut- en distributiegerelateerde controles. Tijdens de verdere ontwikkeling kwamen daar bijkomende controles bij om nieuwe datakwaliteitsproblemen te detecteren en bestaande validaties te verbreden.

Het doel was een breder spectrum aan inconsistenties in de Aphia-databank automatisch te kunnen identificeren: niet alleen ontbrekende gegevens, maar ook inconsistenties tussen gerelateerde records, taxonomische afwijkingen en problemen binnen bronverwijzingen [10].

Domein	Kwaliteitscontrole	Doel
Taxa	Accepted taxa with unaccepted parent	Opsporen van inconsistente parentrelaties
Taxa	Duplicate taxa by display name	Detecteren van mogelijke duplicaten op basis van displaynaam
Taxa	Missing nominal subgenus	Detecteren van ontbrekende nominale subgenera
Taxa	Missing nominal subspecies	Detecteren van ontbrekende nominale subspecies
Taxa	Multiple accepted taxa share the same basionym	Detecteren van geaccepteerde taxa met hetzelfde basioniem
Taxa	Parent missing habitat flags present in child	Opsporen van ontbrekende habitatvlaggen in parenttaxa
Taxa	Unaccepted taxa with identical suggested accepted name	Detecteren van verdachte synonymrelaties
Sources	Dead links	Detecteren van onbereikbare hyperlinks
Sources	DOI with spaces	Opsporen van DOI's met ongeldige spaties
Sources	Invalid URL syntax	Detecteren van foutieve URL-structuren

Tabel 8.1: Overzicht van de nieuw geïmplementeerde kwaliteitscontroles.

Verschillende nieuwe controles richtten zich op taxonomische kwaliteit. Voorbeelden zijn controles voor ontbrekende nominale subgenera of subspecies, geaccepteerde taxa met een niet-geaccepteerde parentrelatie en mogelijke duplicaten op basis van displaynamen. Ook controles voor habitatvlaggen en taxonomische relaties tussen synoniemen en geaccepteerde taxa werden toegevoegd.

Buiten het taxonomische domein werden uitbreidingen doorgevoerd voor brongegevens: controles voor DOI's met ongeldige spaties, foutieve URL-syntaxis en onbereikbare hyperlinks maken het mogelijk om kwaliteitsproblemen op te sporen die voordien niet automatisch gedetecteerd werden [11].

Door deze uitbreidingen evolueerde de toolbox van een verzameling basiscontroles naar een uitgebreider validatieplatform dat een groter deel van de databankkwaliteit automatisch bewaakt.

8.7 EVALUATIE VAN DE WIJZIGINGEN

De functionele en gebruikersgerichte verbeteringen waren gericht op een betere bruikbaarheid van de QC-toolbox, zonder de kernfunctionaliteit van de databank te raken. De aandacht ging daarbij naar twee vlakken: de presentatie van informatie en de uitbreiding van de beschikbare kwaliteitscontroles.

Aspect	Oorspronkelijke interface	Vernieuwde interface
Structuur	Lange lijst	Gegroepeerde categorieën
Organisatie	Eén overzicht	Inklapbare secties
Schermgebruik	Voornamelijk verticaal	Masonry-indeling
Contextinformatie	Beperkt	Scope- en statusmeldingen
Resultaatweergave	Lijst van records	Gevisualiseerde verschillen
Navigatie	Meer doorklikken nodig	Meer informatie direct zichtbaar

Tabel 8.2: Vergelijking tussen de oorspronkelijke en vernieuwde gebruikersinterface.

De herwerkte overzichtspagina biedt een duidelijkere structuur door kwaliteitscontroles logisch te groeperen en gebruik te maken van inklapbare secties. Gebruikers kunnen sneller relevante controles terugvinden en de interface blijft overzichtelijk, ook wanneer nieuwe controles worden toegevoegd.

Ook de resultatenpagina's werden inhoudelijk versterkt. Bijkomende contextinformatie, filtering en een duidelijkere presentatie van kwaliteitsproblemen maken sneller zichtbaar waarom een record geselecteerd werd, waardoor minder navigatiestappen nodig zijn om de oorzaak van een probleem te achterhalen.

Naast de gebruikersinterface werd ook de functionele dekking uitgebreid door nieuwe kwaliteitscontroles. Meer soorten datakwaliteitsproblemen kunnen daardoor automatisch worden opgespoord, wat de toolbox waardevoller maakt voor databankbeheerders.

De toolbox is daarmee beter afgestemd op de dagelijkse werking van VLIZ en heeft voldoende ruimte voor verdere uitbreidingen.

9 RESULTATEN EN IMPACT

9.1 PERFORMANTIEVERBETERINGEN

De uitvoeringstijden van verschillende kwaliteitscontroles waren bij aanvang een concreet knelpunt. Sommige controles hadden uitvoeringstijden van tientallen seconden, in uitzonderlijke gevallen zelfs meer dan twee minuten, wat het werken met grote datasets tijdrovend maakte.

Via een combinatie van query-optimalisaties en cachingstrategieën konden die tijden aanzienlijk worden teruggebracht. Eerst werden de onderliggende SQL-queries geoptimaliseerd via vroegtijdige filtering, tijdelijke datasets, indexering en herschreven querylogica. Voor controles die daarna nog een hoge uitvoeringstijd hadden, werd caching ingezet om eerder berekende resultaten te hergebruiken.

Kwaliteitscontrole	Oorspronkelijk	Geoptimaliseerd	Cache
authorityAndSourceNotMatching	~108 s	~20 s	~1.6 s
DisplaynameOutOfSync	>120 s	~20 s	~0.7 s
CompareValues	~12.6 s	<1 s	n.v.t.
DeletedParent	~12 s	<1 s	n.v.t.

Tabel 9.1: Evolutie van de uitvoeringstijden voor geselecteerde kwaliteitscontroles.

De tabel toont een duidelijke stapsgewijze verbetering. Voor verschillende kwaliteitscontroles zorgden query-optimalisaties al voor een sterke daling van de uitvoeringstijd; voor de meest complexe controles bracht caching daar nog een bijkomende versnelling bij.

Dit toont aan dat caching niet als vervanging van query-optimalisatie werd ingezet, maar als aanvullende techniek voor controles waarvan de uitvoeringstijd na optimalisatie nog relatief hoog bleef. Voor queries zoals *CompareValues* en *DeletedParent* werd geen caching toegepast, omdat de uitvoeringstijden na query-optimalisatie al voldoende laag waren om rechtstreekse uitvoering te rechtvaardigen.

Dat had ook een direct effect op de gebruikerservaring. Gebruikers hoeven minder lang te wachten op resultaten en kunnen sneller tussen controles navigeren, wat de toolbox beter bruikbaar maakt tijdens dagelijkse kwaliteitscontroles.

9.2 VERBETERDE ONDERHOUDBAARHEID

Naast performantie werd ook de onderhoudbaarheid van de codebasis verbeterd. De oorspronkelijke implementatie was sterk gecentraliseerd rond één grote bibliotheek waarin kwaliteitscontroles, gebruikersinterfacefunctionaliteit en ondersteunende logica door elkaar liepen.

Door de architecturale herwerking werden verantwoordelijkheden duidelijker gescheiden en werd de code opgesplitst in gespecialiseerde componenten. Wijzigingen aanbrengen zonder onverwachte neveneffecten in andere onderdelen te veroorzaken werd daarmee eenvoudiger.

Ook het debuggen verbeterde. Ontwikkelaars kunnen sneller bepalen in welk onderdeel een probleem zich bevindt, waardoor fouten efficiënter kunnen worden onderzocht en opgelost.

De verbeterde structuur maakt de codebasis ook toegankelijker voor toekomstige ontwikkelaars. Nieuwe functionaliteit toevoegen vereist niet langer een volledig begrip van de interne werking van de oorspronkelijke bibliotheek.

9.3 VERBETERDE SCHAALBAARHEID

De herwerkingen maakten de QC-toolbox ook beter voorbereid op toekomstige groei. De modulaire architectuur maakt het mogelijk om nieuwe kwaliteitscontroles toe te voegen zonder bestaande functionaliteit ingrijpend te wijzigen.

Ook de cachingarchitectuur draagt bij aan schaalbaarheid. Door zware kwaliteitscontroles selectief te voorzien van snapshots blijft de belasting van de databank beheersbaar, ook wanneer datasets verder groeien of bijkomende controles worden toegevoegd.

De duidelijkere scheiding tussen data-opvraging, verwerking en presentatie maakt het bovendien eenvoudiger om toekomstige optimalisaties gericht toe te passen, zonder dat fundamentele architecturale wijzigingen noodzakelijk zijn.

9.4 TOEGEVOEGDE WAARDE VOOR VLIZ

De verbeteringen aan de QC-toolbox zijn op meerdere vlakken merkbaar voor VLIZ. Lagere uitvoeringstijden betekenen dat kwaliteitscontroles sneller worden uitgevoerd en databankbeheerders minder tijd kwijt zijn aan wachten.

Het uitgebreide aantal kwaliteitscontroles verhoogt ook de functionele dekking. Meer soorten datakwaliteitsproblemen worden automatisch opgespoord, waardoor potentiële fouten sneller aan het licht komen.

De vernieuwde interface draagt daar ook aan bij. Overzichtelijkere pagina's, uitgebreidere resultatenweergaven en bijkomende contextinformatie maken het eenvoudiger om kwaliteitsproblemen te analyseren en op te lossen.

Op langere termijn maakt de vernieuwde architectuur toekomstige uitbreidingen ook eenvoudiger, zodat de toolbox bruikbaar blijft als instrument voor kwaliteitsbewaking binnen Aphia.

9.5 DEPLOYMENT EN PRODUCTIEGEBRUIK

Aan het einde van de stage werden de gerealiseerde verbeteringen uitgerold naar de productieomgeving van VLIZ. De vernieuwde interface en de geoptimaliseerde kwaliteitscontroles werden daarmee opgenomen in de bestaande workflow van de databankbeheerders.

Hoewel de verbeteringen in productie werden uitgerold, bleven de kwaliteitscontroles voorlopig uitsluitend toegankelijk voor DMT-gebruikers. Hierdoor kon de nieuwe functionaliteit eerst binnen de bestaande kwaliteitsworkflow worden ingezet voordat een eventuele bredere beschikbaarstelling wordt overwogen.

Met de gerealiseerde performantieverbeteringen, functionele uitbreidingen en een onderhoudbare architectuur is de toolbox klaar voor verdere uitbreiding en langdurig gebruik binnen Aphia.

10 REFLECTIE IN HET LEERPROCES

10.1 TECHNISCHE GROEI

Tijdens de stage kon ik mijn technische vaardigheden uitbreiden op manieren die bij eerdere projecten zelden aan bod kwamen. Waar die projecten vaak beperkt bleven tot afgebakende opdrachten of nieuwe toepassingen, werkte ik hier voornamelijk binnen een bestaande codebasis die al deel uitmaakte van het Aphia-ecosysteem.

Het analyseren en optimaliseren van complexe SQL-queries was daarin een belangrijk onderdeel. Ik leerde execution plans interpreteren, performantieproblemen identificeren en gerichte optimalisaties toepassen. Tegelijk deed ik ervaring op met cachingstrategieën, benchmarking en het afwegen van de impact van technische keuzes op de prestaties van een applicatie.

De architecturale herwerking van de QC-toolbox leerde me ook hoe softwareontwikkeling in een professionele omgeving werkt. Werken met bestaande code vraagt meer dan alleen nieuwe functionaliteit toevoegen: de onderhoudbaarheid van wat er al staat, is minstens even belangrijk.

10.2 DOMEINKENNIS EN INTERDISCIPLINAIR LEREN

Het leerproces tijdens de stage bleef niet beperkt tot technische vaardigheden. Om kwaliteitscontroles correct te kunnen ontwikkelen en uitbreiden, moest ik me ook verdiepen in de taxonomische structuur van Aphia en de bijhorende regels.

Verschillende kwaliteitscontroles waren gebaseerd op nomenclatorische principes en taxonomische relaties [10]. Ik moest taxonomische begrippen opzoeken, wetenschappelijke documentatie raadplegen en nomenclatorische regels bestuderen om de achterliggende logica van bepaalde controles te begrijpen.

Mijn achtergrond ligt voornamelijk in cybersecurity en softwareontwikkeling, maar tijdens de stage merkte ik hoe sterk domeinkennis de kwaliteit van een technische oplossing kan beïnvloeden. Snel inwerken in een nieuw vakgebied en die kennis vertalen naar een concrete implementatie beschouw ik als een van de belangrijkste dingen die ik tijdens deze stage heb geleerd.

10.3 ZELFSTANDIGHEID EN COMMUNICATIE

Een groot deel van het werk voerde ik zelfstandig uit. Dat gaf me de ruimte om problemen zelf te analyseren, oplossingen uit te werken en technische beslissingen te nemen binnen de grenzen van het project.

Die zelfstandigheid vergrootte mijn vertrouwen in het aanpakken van complexe problemen. Ik leerde systematisch te werk gaan, verschillende oplossingsrichtingen te vergelijken en keuzes te onderbouwen aan de hand van meetbare resultaten.

Tegelijk maakte de stage me bewust van het belang van proactieve communicatie. De vrijheid die zelfstandig werken biedt, heeft een keerzijde: in sommige situaties had tijdigere afstemming met begeleiders of stakeholders sneller duidelijkheid gegeven. Dat is iets wat ik wil meenemen naar toekomstige projecten.

10.4 TERUGBLIK OP DE STAGE

Terugkijkend was de stage een echte kennismaking met softwareontwikkeling in een professionele omgeving. Het werk ging verder dan alleen nieuwe functionaliteit ontwikkelen: performantieproblemen aanpakken, de architectuur herwerken en de gebruikerservaring verbeteren vormden een belangrijk deel van het project.

Het traject liet me ook een volledige ontwikkelingscyclus doorlopen: van analyse en ontwerp tot implementatie, testing en uitrol naar productie. Dat gaf me een concreter beeld van wat het betekent om software te onderhouden en uit te breiden binnen een grotere organisatie.

Technisch heb ik veel bijgeleerd, maar de stage leerde me ook hoe belangrijk het is om oplossingen af te stemmen op de context en de gebruikers waarvoor ze bedoeld zijn. Dat maakt deze stage voor mij een goede voorbereiding op een professionele carrière.

11 CONCLUSIE

11.1 CONCLUSIE

Tijdens de stage werd gewerkt aan de verdere ontwikkeling van de Aphia Quality Control Toolbox bij VLIZ. De focus lag op vier aspecten: de performantie van bestaande kwaliteitscontroles, de onderhoudbaarheid van de codebasis, de functionele dekking en de gebruiksvriendelijkheid van de interface.

De architecturale herwerking verdeelde verantwoordelijkheden duidelijker en splitste de code op in gespecialiseerde componenten, waardoor de codebasis overzichtelijker en beter uitbreidbaar werd.

Voor performantie werden execution plans geanalyseerd en queries herschreven via vroegtijdige filtering, tijdelijke tabellen en gerichte indexering. De zwaarste kwaliteitscontroles kregen aanvullend een cachingmechanisme om herhaalde uitvoeringen verder te versnellen.

Functioneel werd de toolbox uitgebreid met nieuwe kwaliteitscontroles die bijkomende datakwaliteitsproblemen automatisch detecteren. Tegelijkertijd werd de gebruikersinterface herwerkt om kwaliteitscontroles en resultaten beter leesbaar te presenteren.

Die verbeteringen samen resulteerden in een toolbox die performanter, onderhoudbaarder en bruikbaarder is dan de oorspronkelijke versie, en die een betere basis biedt voor toekomstige ontwikkelingen binnen Aphia.

11.2 AANBEVELINGEN VOOR TOEKOMSTIGE ONTWIKKELING

Ondanks de gerealiseerde verbeteringen zijn er nog richtingen voor verdere ontwikkeling van de QC-toolbox.

Op het vlak van performantie kunnen toekomstige optimalisaties zich richten op kwaliteitscontroles die ook na optimalisatie nog relatief hoge uitvoeringstijden vertonen. Naarmate de Aphia-databank verder groeit, kunnen bovendien nieuwe performantie-uitdagingen ontstaan die bijkomende analyse vereisen.

Ook caching biedt nog ruimte voor uitbreiding. Tijdens de stage werd caching selectief toegepast op de meest performantiegevoelige kwaliteitscontroles. In de toekomst kan onderzocht worden of bijkomende controles baat hebben bij caching of andere vormen van voorverwerking.

Een andere mogelijke evolutie is het verschuiven van een periodiek validatiemodel naar een gebeurtenisgestuurde aanpak. In plaats van controles dagelijks opnieuw uit te voeren of snapshots periodiek op te bouwen, zouden relevante controles automatisch kunnen worden uitgevoerd wanneer een record wordt aangemaakt of gewijzigd.

De resultaten van die controles kunnen dan persistent worden opgeslagen, waardoor kwaliteitsproblemen onmiddellijk beschikbaar zijn voor gebruikers zonder bijkomende queryverwerking of cachevernieuwing. Dat zou het risico op verouderde cachegegevens verder beperken en problemen sneller zichtbaar maken.

De praktische haalbaarheid van die aanpak vereist wel bijkomend onderzoek. Niet alle kwaliteitscontroles zijn eenvoudig te koppelen aan individuele wijzigingen, en een aanpassing aan één record kan indirect gevolgen hebben voor gerelateerde records, wat bijkomende validatiemechanismen kan vereisen.

De modulaire architectuur biedt daarnaast ruimte voor nieuwe kwaliteitscontroles die de dekking verder vergroten en bijdragen aan een hogere datakwaliteit binnen Aphia.

11.3 PERSOONLIJKE TOEKOMSTVISIE

De stage gaf me een eerste echte kennismaking met softwareontwikkeling in een professionele omgeving. Naast de technische kant kon ik er ook vaardigheden als communicatie en zelfstandig werken verder ontwikkelen.

Een van de belangrijkste lessen was hoe sterk domeinkennis de kwaliteit van een technische oplossing kan beïnvloeden. Het begrijpen van taxonomische structuren en nomenclatorische regels bleek essentieel om kwaliteitscontroles correct te implementeren. Dat heeft mijn interesse versterkt in projecten waar technische uitdagingen samengaan met domeinspecifieke kennis.

De stage bevestigde ook mijn interesse in performantie-optimalisatie en het analyseren van complexe systemen. Performantieproblemen onderzoeken, optimalisaties uitwerken en de impact ervan meten waren de onderdelen van het project die ik het meest boeiend vond.

In mijn verdere loopbaan wil ik me ontwikkelen in technische rollen waar analyse en probleemoplossing centraal staan. De ervaring die ik tijdens deze stage opdeed geeft daar een goede basis voor.

BRONNEN

[1] Vlaams Instituut voor de Zee (VLIZ), „Over VLIZ,” [Online]. Available: https://www.vliz.be. [Geopend 24 May 2026].

[2] WoRMS Editorial Board, "World Register of Marine Species," 2026. [Online]. Available: https://www.marinespecies.org. [Accessed 24 May 2026].

[3] L. Vandepitte, B. Vanhoorne, W. Decock, S. Vranken, D. Anseeuw and J. Mees, "How Aphia has supported world registers of marine and non-marine species for a decade," *ZooKeys,* no. 524, pp. 147-163, 2015.

[4] D. L. Parnas, "On the Criteria To Be Used in Decomposing Systems into Modules," *Communications of the ACM,* vol. 15, no. 12, pp. 1053-1058, 1972.

[5] R. C. Martin, Clean Architecture: A Craftsman's Guide to Software Structure and Design, Prentice Hall, 2017.

[6] Microsoft, "Query Processing Architecture Guide," [Online]. Available: https://learn.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide. [Accessed 25 May 2026].

[7] Microsoft, "Execution Plans," [Online]. Available: https://learn.microsoft.com/en-us/sql/relational-databases/performance/execution-plans. [Accessed 25 May 2026].

[8] Microsoft, "SQL Server Index Architecture and Design Guide," [Online]. Available: https://learn.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide. [Accessed 25 May 2026].

[9] M. Fowler, Patterns of Enterprise Application Architecture, Addison-Wesley, 2002.

[10] International Commission on Zoological Nomenclature, International Code of Zoological Nomenclature, 4th Edition ed., The International Trust for Zoological Nomenclature, 1999.

[11] International DOI Foundation, "DOI Handbook," [Online]. Available: https://www.doi.org/doi_handbook/. [Accessed 1 June 2026].

howest.be