Ontwerp Applicatie PG6 Course 8

Richard Jansen, Rick Medemblik, Stan Wehkamp, Alex Staritsky

May 2017

Contents

1 Introductie

1.1 Aanleiding

HAN Biocentre doet onderzoek naar lepoxygenases. lepoxygenases hebben vele toepassingen maar deze zijn tot nu toe niet gezamelijk gedocumenteerd. Om verder onderzoek voor deze toepassingen te versimpelen is een tool ontwikkels die de lepoxygenases en hun toepassingen uit artikelen kan halen en op een overzichtelijke manier kan weergeven.

1.2 Doel van dit document

Dit document bevat het ontwerp van de applicatie. het doel van dit document is om de opbouw van de applicatie verduidelijken.

1.3 Doel van de applicatie

Het doel van de applicatie is om geschreven beschreven toepassingen van lepoxygenases uit artikelen te halen.

1.4 Doelgroep

De applicatie is bedoeld voor het gebruik door biologen. Voor de gebruikers dient dit document om de benodigde systeemeisten te achterhalen. Verder kunnen softwareontwikkelaars dit document gebruiken als leidraad voor het ontwikkelen van de applicatie. Tot slot kan de sorftware beheerders dit document gebruiken om een beeld te krijgen van de opbouw van deze applicatie.

2 Project beschrijving

2.1 Startpunt van het project

We beginnen from scrach met het opbouwen van de applicatie. Er begonnen met het maken van een webpagina waar GUI in is verwerkt. De om de gebruiker met de webpagina te verbinden word de module python-flask gebruikt

2.2 Gekozen technieken

Taal: Python 2.7 [3] , Python-Flask[2], Biopython ver. 1.69 of hoger[1]
Platform: servlet
Communicatie protocollen: FTP, HTTP

2.3 Design principes

De applicatie zal door de gebruiker gebruikt kunnen worden door middel van een GUI in de vorm van een webpagina. Omdat over het verloop naar tijd de hoeveelheid gevonden data toeneemt zal er meer opslagruimte nodig zijn om de efficentie van het programma en de voledigheid van de data te bewaren. Om hier rekening mee te houden staat in dit document een database beschreven die geïmplementeerd kan worden als meer opslagruimte wenselijk is.

2.4 Design patterns

De gebruiker kan via zijn computer de webapplicatie starten. De webapplicatie zal vervolgens de gewenste resultaten uit de NCBI database halen en deze op de computer van de gebruiker weergeven. De gebruiker kan deze resultaten lokaal opslaan.

3 Systeemarchitectuur

3.1 Doel van de systeemarchitectuur

Om een goed beeld te krijgen van de werking van de applicatie is de systeemarchitectuur een belangrijk aspect van het ontwerp. De applicatie is ontworpen voor makkelijke implementatie en gebruik door meerdere gebruikers. Om dit te realiseren is er gekozen voor het gebruik van een servlet op basis van Python-Flask.

3.2 Gedetailleerde systeemarchitectuur

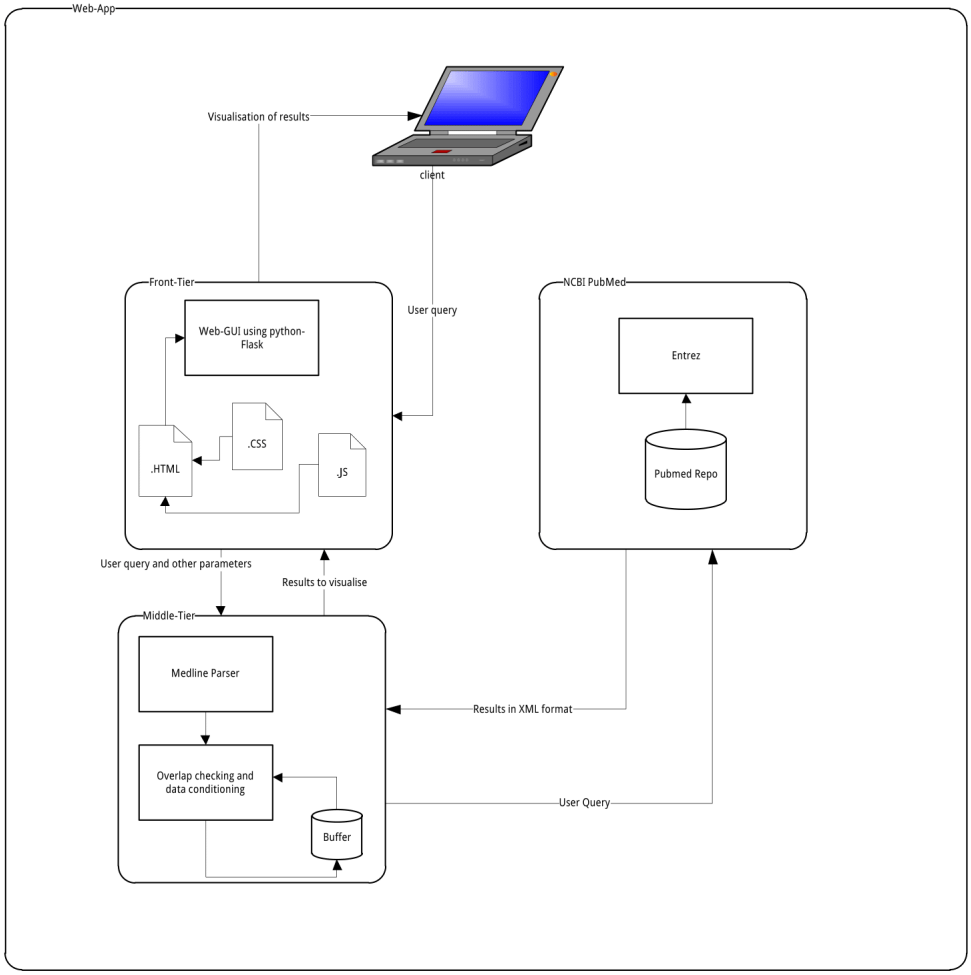

Figure 1: Systeemarchitectuur schematisch weergeven

3.2.1 Front-tier

De front-tier, zie bovenstaande figuur, is het gedeelte van de applicatie dat verantwoordelijk is voor de verwerking van gebruikers-input en het visualiseren van de output. Er is gekozen voor een web-based applicatie en dat betekend dat correcte werking van de applicatie berust op de aanwezigheid van deze vereisten:

- Web-browser
- Internetverbinding (of evt. Intranet)

De front-end gebruikt HTML, CSS en Java Script voor de visualisatie van data en input van gebruikers.

3.2.2 Middle-tier

De zoekopdracht van de eindgebruiker die door de front-tier wordt doorgestuurd naar de middle-tier wordt vervolgens verwerkt to een entrez-query en de output daarvan bevat metadata dat gebruikt kan worden om gerelateerde data op te vragen zoals gerelateerde sequenties en dergelijke. Indien de gebruiker het resultaat tijdelijk op wilt slaan is het mogelijk om gebruik te maken van de buffer, bestaande uit een MySQL of JSON database, die vervolgens de data per gebruiker archiveert. De volgende software is noodzakelijk voor de middle-tier:

- Python 2.7
- Python-Flask ver. 0.12.1 of hoger
- Biopython ver. 1.69 of hoger
- MySQL Server ver. 5.5 of hoger

3.2.3 NCBI PubMed

De applicatie maakt gebruik van Entrez, een API om de NCBI databases te doorzoeken. In dit geval vormt de middle tier een entrez-query vanuit de gebruikersinput, vervolgens genereert de entrez server een XML-document met alle resultaten en stuurt het terug naar de middle tier. De eerste zoekopdracht levert basale metadata op van de gevonden documenten. Vervolgens kiest de gebruiker van welke documenten hij/zij meer informatie wilt. Vervolgens genereert de middle-tier een entrez-query om de sequentie en dergelijke op te vragen.

3.3 Systeemspecificaties

4 Software-architectuur

4.1 Doel van de software-architectuur

Het doel van de software-architectuur is om voor de programmeurs duidelijk te krijgen hoe het programma er ongeveer uit zal gaan zien. Er wordt in een UML class diagram weergeven hoe alle classes eruit gaan zien, wat voor methoden en attributen ze hebben en welke relaties tussen de classes zitten. Het mooie van de UML is dat het de wereldwijde standaard voor dit type diagrammen is en dat het niet gebonden is aan één programmeertaal.

4.2 UML Class Diagram(s)

4.2.1 SearchRequest

De SeachRequest class is een class die om een normaal Flask request object gedaan wordt om eenvoudig de informatie uit het request object te halen zoals de query van de gebruiker en instellingen die via de webbrowser gekozen zijn.

4.2.2 Article

Bij het maken van een Article object wordt een PubMed identifier gevraagd. Aan de hand van de PubMed identifier wordt het artikel gedownload met meta-informatie. Met de ingebouwde parser in het object wordt de informatie uit de gedownloade data gehaald. Deze informatie wordt overzichtelijk opgeslagen in het object en kan gemakkelijk met hulp van getters opgevraagd worden.

4.2.3 Protein

Het Protein object werkt op een soortgelijke manier als het Artikel object. Bij de aanmaak van een Protein object wordt een identifier gevraagd. In dit geval gaat het om een eiwit-identifier van NCBI. Een GenPept bestand wordt aan de hand van de identifier opgevraagd en met de ingebouwde parser in het object wordt alle (nuttige) informatie uit het GenPept bestand opgehaald en opgeslagen in de attributen van het object. Deze informatie van het GenPept bestand kan eenvoudig met getters opgehaald worden.

4.2.4 TextMiner

Bij de creatie van een TextMiner object word een SeachRequest object gevraagd. Aan de hand van de informatie die uit een SearchRequest object komt worden verschillende id's van artikelen en eiwitten opgehaald uit Entez van NCBI. Vervolgens worden uit deze id's in het SeachRequest object omgezet in Article en Protein objecten waarin de daadwerkelijke artikelen en GenPept bestanden ook gedownload en geparst worden. Deze objecten worden gebruikt in combinatie met verschilende (nog onbepaalde) algoritmes om keywords uit de abstracts te halen en deze keywords aan elkaar linken en plaatsen in een graaf datastructuur.

4.2.5 Result

De Result class is een class waarin de verschillende resultaten van het TextMiner object in opgeslagen worden. Specifieker alleen de resultaten die uiteindelijk in de tabel voor de gebruiker terugkomen. Denk hierbij aan attributen als organisme, sequentie en publicaties die daarbij horen.

4.2.6 ResultSet

De ResultSet class is een soort lijst is met extra functionaliteit. Hiebij wordt niet alleen een lijst van alle resultaten in opgeslagen, maar ook meta-informatie over die resultaten zoals de SearchRequest die door de TextMiner gebruikt is om de resultaten in de ResultSet te genereren. Verder zit er ook een graaf verwerkt in de ResultSet die de verschillende relaties tussen de artikelen moet voorstellen voor de uiteindelijke visualisatie.

4.2.7 Graph

De Graph class is een class met een graaf datastuctuur en methoden waarmee data (in de vorm van JSON) of markup language (in de vorm van HTML5) voor de visualisatie van de graaf gegenenereerd kan worden. Deze data of markup kan vervolgens naar de front-end gestuurd worden om de gebruiker een visualisatie te weergeven. Een instantie van de graaf kan uit een ResultSet gehaald worden.

4.2.8 Table

Een Table object kan uit een ResultSet object opgevraagd worden. De Table class bevat de informatie van een ResultSet die voor een tabel gebruikt kan worden. Het Table object bevat dan methoden om HTML voor de tabel te genereren die naar de front-end van de applicatie opgestuurd kan worden.

4.2.9 UML Class Diagram

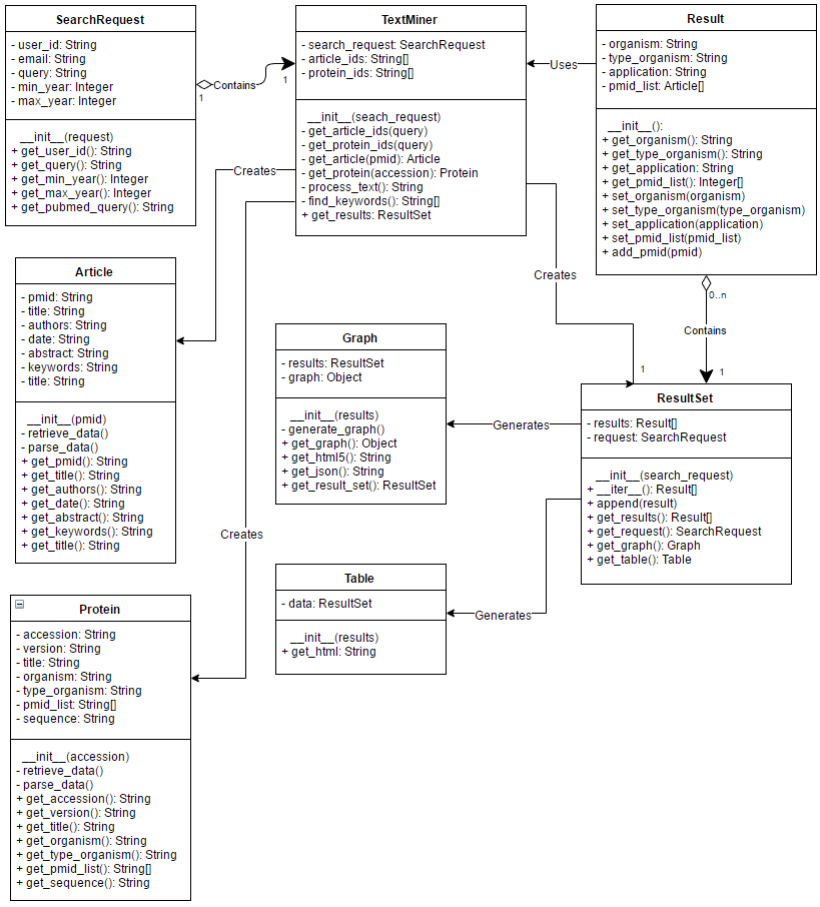

Figure 2: UML class diagram van webapplicatie. Van links naar rechts en van boven naar beneden zijn de volgende classes te zien. SearchRequest die het makkelijk maakt om de query en instellingen van de gebruiker op te vragen. TextMiner waar alle logica zit voor het zoeken naar id's op NCBI voor zowel PubMed artikelen als GenPept eiwitten, verder zit er ook de logica voor de textmining en koppeling van artikelen. Result dat een onderdeel is van de Result lijst in ResultSet; bevat de informatie voor een rij in de tabel voor de gebruiker. Article waarmee een artikel gedownload en geparst kan worden; vervolgens kan de informatie met getters opgehaald worden. Graph dat een graafstructuur met extra functionaliteit moet voorstellen; functionaliteit waarbij er JSON of HTML5 voor de front-end visualisatie gegenereerd kan worden. ResultSet die een lijst met Result objecten bevat van de TextMiner (is indirect een Tabel object), een SearchRequest die gebruikt is om de resultaten te genereren en de Graph van het koppelen van artikelen. Protein waarmee een GenPept gedownload en geparst kan worden en de informatie met getters opgehaald kan worden. Ten slotte is er nog Table dat een tabel moet voorstellen; deze bevat logica om uit de ResultSet data HTML voor de front-end te genereren.

5 Technische gegevens structuur

5.1 Doel van de technische gegevens structuur

Het doel van de technische gegevens structuur is het laten zien hoe onze database opgezet gaat worden. Er zijn meerdere entiteiten die aan elkaar gelinkt zijn. Hier worden de logische en technische modellen weergegeven en uitgelegd wat de keuzes voor de opzet van de database. Een kant tekening moet wel worden gemaakt. De database die in dit hoofdstuk is uitgelegd wordt mogelijk niet gebruikt voor het project. Dit model is een proof of concept.

5.2 Logisch model

Het logisch model is een simpel overzicht van de database. Per entiteit wordt weergegeven welke attributen deze bevat. Ook wordt de relatie tussen de entiteit aangegeven.

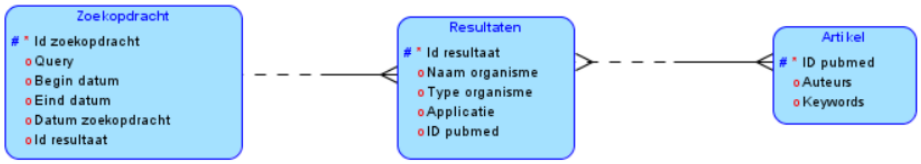

Figure 3: *Logisch ERD van dit project. In het bovenstaand figuur is de proof of concept van het logisch ERD voor de database weergegeven. Deze database bevat 3 entiteiten waarvan er een 1 op meer relatie en een meer op meer relatie aanwezig zijn. Een uitgevoerde zoekopdracht heeft 1 of meer resultaten. De resultaten kunnen meerder artikelen bevatten maar een artikel kan ook bij meerdere resultaten horen. Per entiteit wordt weergegeven welke informatie hierin wordt opgeslagen. Bovenstaande logisch ERD is gemaakt met SQL Developer Data Modeler.*

5.2.1 Zoekopdracht

Deze entiteit bevat een unieke zoekopdracht ID, de zoekopdracht (query), de datum vanaf wanneer gezocht moet worden (Begin datum), de datum tot wanneer gezocht moet worden (Eind datum), de datum wanneer de zoekopdracht uitgevoerd is (Datum zoekopdracht) en het unieke resultaat ID behorend bij de entiteit resultaten.

5.2.2 Resultaten

Deze entiteit bevat een unieke resultaat id, de naam van het organisme, het type organisme (is het bijv. een mens), de applicatie/toepassing en het pubmed ID behorend bij de entiteit artikel.

5.2.3 Artikel

Deze entiteit bevat een unieke pubmed ID, de auteurs en de keywords behorend bij het artikel.

5.3 Technisch model

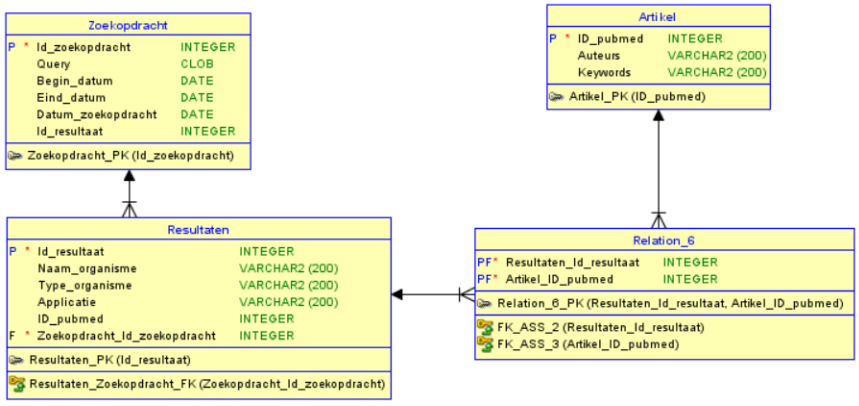

Figure 4: *Technisch ERD van dit project. Dit figuur bevat het technische ERD van de database. Deze database bevat 3 entiteiten waarvan er een 1 op meer relatie en een meer op meer relatie aanwezig zijn. Verder zijn er 3 primary keys en 1 foreign key. Bovenstaande technische ERD is gemaakt met SQL Developer Data Modeler.*

References

[1] Peter JA Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, et al. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, 2009.

[2] Miguel Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2014.

[3] Guido Van Rossum and Fred L Drake Jr. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.