|  |  |
| --- | --- |
| Projectgroep 11 | |
| Document | Ontwerp\_Bi8aGroep11.docx | |
| Datum | 19-05-2017 | |
| Versie | 1.0 | |
| Groepsleden | Koen van der Heide | |
|  | Rick Beeloo | |
|  | Thomas Reinders | |
|  |  | |

CoTexThen

Automatische PubMed analyse

# Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Datum | Auteur | Beschrijving |
| 0.1 | 15-05-2017 | Koen van der Heide | Inleiding uitgewerkt. |
|  |  |  |  |
| 0.4 | 18-05-2017 | Koen van der Heide | Project beschrijving toegevoegd. |
| 0.5 | 19-05-2017 | Koen van der Heide | Technische structuur toegevoegd. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[Versiebeheer 2](#_Toc482996881)

[1. Introductie 5](#_Toc482996882)

[Aanleiding 5](#_Toc482996883)

[Doel van dit document 5](#_Toc482996884)

[Gehanteerde richtlijnen 5](#_Toc482996885)

[Doel van de applicatie 5](#_Toc482996886)

[Doelgroep 5](#_Toc482996887)

[2. Project beschrijving 6](#_Toc482996888)

[Startpunt van het project 6](#_Toc482996889)

[Gekozen technieken 6](#_Toc482996890)

[Design principes 6](#_Toc482996891)

[Design patterns 6](#_Toc482996892)

[3. Systeemarchitectuur 7](#_Toc482996893)

[Doel van de systeemarchitectuur 7](#_Toc482996894)

[Visualisation 7](#_Toc482996895)

[Data storage and retrieval 7](#_Toc482996896)

[Textmining 7](#_Toc482996897)

[PubMed Searching 8](#_Toc482996898)

[Systeemspecificaties 9](#_Toc482996899)

[4. Software-architectuur 10](#_Toc482996900)

[Doel van de software-architectuur 10](#_Toc482996901)

[UML Class Diagram(s) 10](#_Toc482996902)

[5. Lay-out web applicatie 12](#_Toc482996903)

[Doel van de Lay-out web applicatie 12](#_Toc482996904)

[Basis lay-out 12](#_Toc482996905)

[Voorbeelden 13](#_Toc482996906)

[6. Technische gegevens structuur 14](#_Toc482996907)

[Doel van de technische gegevens structuur 14](#_Toc482996908)

[Logisch model 14](#_Toc482996909)

[Technisch model 16](#_Toc482996910)

[7. Begrippenlijst 17](#_Toc482996911)

[Bronvermelding 18](#_Toc482996912)

# Introductie

## Aanleiding

Anthocyanen zijn kleurstoffen geproduceerd door planten en zijn onder andere verantwoordelijk voor de rijke kleuren aanwezig in bloemen. Echter kunnen anthocyanen ook dienen als bescherming voor de plant wanneer deze stress ervaart, deze laatste eigenschap maken anthocyanen interessant voor plantentelers aangezien stress bestendige planten tot een betere opbrengst voor de telers kunnen leiden. Er is echter nog geen database aanwezig waarin vastgelegd is welke planten anthocyanen onder stress produceren, welke genen hierbij betrokken zijn en welke stress omstandigheden precies tot anthocyanen productie leiden. Hierdoor moeten biologen artikelen over anthocyanen tot nu toe handmatig analyseren om relevante informatie over anthocyanen te ontdekken. In dit project zal dit probleem opgelost worden door een dergelijke database op te zetten waarbij wetenschappelijke artikelen automatisch op de relevante termen zijn doorzocht om deze database te vullen. De data uit deze database zal vervolgens op een overzichtelijke manier voor de gebruiker gepresenteerd worden.

## Doel van dit document

In dit document is vastgelegd hoe de applicatie opgezet gaat worden. Hierbij zijn de eisen uit de analyse vertaald naar de technische functionaliteiten van het systeem. Deze functionaliteiten worden hier nader toegelicht op een manier begrijpbaar voor ontwikkelaars en beheerders door middel van een systeemarchitectuur, software-architectuur en de technische gegevens structuur.

## Gehanteerde richtlijnen

Het UML class diagram en ERD diagram zijn ontworpen op basis van de richtlijnen zoals vermeld in “Praktisch UML”( Warmer, J., Kleppe, A.(2011)) en “Normalizing with Entity Relationship Diagramming” (Kaula, R.(2007)), respectievelijk.

## Doel van de applicatie

Zoals in de aanleiding is genoemd zal deze applicatie overzichtelijk aan zijn gebruikers presenteren welke planten anthocyanen produceren onder stress, welke genen daarbij betrokken zijn en welke (stress) condities hier op van invloed zijn.

## Doelgroep

Dit document richt zich tot de ontwikkelaars en beheerders van de geplande applicatie. Hierbij wordt er voor de ontwikkelaars uitgelegd hoe de applicatie geprogrammeerd dient te worden en voor de beheerders aangetoond uit welke onderdelen de applicatie bestaat en hoe deze zich tot elkaar verhouden. Verder kan het logische ERD ook gebruikt worden als communicatiemiddel met de opdrachtgever vanwege het tamelijk hoge abstractieniveau.

# Project beschrijving

## Startpunt van het project

De logica zal geprogrammeerd worden in Python 3, hierbij worden voor de text mining de nltk en retina-sdk modules toegepast en het script voor de visualisatie zal Flask gebruiken. De text mining vereist een connectie met de NCBI PubMed database, om dit mogelijk te maken wordt de BioPython module gebruikt, vanzelfsprekend is een internet verbinding hiervoor dus ook noodzakelijk. Verder worden de resultaten van de textmining opgeslagen in een MySQL database, hiervoor is de MySQL Connector/Python vereist.

## Gekozen technieken

Zoals gezegd zal als programmeertaal Python 3 gebruikt worden aangezien Python zeer geschikt is om grote hoeveelheden tekst te doorzoeken. De Python scripts zullen draaien op een externe server (logica tier) waarop Apache draait, de gebruiker kan hier vanaf zijn of haar eigen pc (client tier) verbinding mee maken via HTTP(S). Sommige Python scripts zullen ook verbinding maken met de MySQL database die op dezelfde externe server staat (database tier), deze verbinding zal tot stand komen door gebruik te maken van MySQL Connector/Python.

De visualisatie zelf vindt plaats door de gebruiker de resultaten van de Python scripts te tonen in overzichtelijke en esthetisch verantwoorde diagrammen. Deze diagrammen zijn benaderbaar via het web en worden opgebouwd uit HTML, CSS en/of JavaScript.

## Design principes

Het belangrijkste principe is dat de visualisatie voor de gebruiker overzichtelijk is en bij een toename van de data ook overzichtelijk blijft. De grootte van de diagrammen die de visualisatie verzorgen zal dus dynamisch moeten zijn waarbij een toename van de hoeveelheid data leidt tot een groter diagram. Ook wordt het belangrijk geacht dat, in verband met de tijdsduur, de gebruiker niet zelf telkens de zoekopdracht hoeft uit te voeren. De database zal dus van tevoren al gevuld zijn met de data verkregen door de text mining en er hoeft dan enkel nog gevisualiseerd te worden. Om meerdere gebruikers de mogelijkheid te bieden de data te raadplegen zal deze database, en de logica noodzakelijk voor de visualisatie en text mining, op een externe server geplaatst worden.

## Design patterns

De applicatie bestaat uit 3 tiers: de client tier, logica tier en database tier. De client tier is de lokale computer van de gebruiker, deze tier dient enkel om verbinding te maken met de externe server. Deze server bevat vervolgens de logica en database tiers, de logica tier is verantwoordelijk voor het uitvoeren van de text mining en visualisatie en de database tier slaat de resultaten van de text mining op om de visualisatie mogelijk te maken.

# Systeemarchitectuur

## Doel van de systeemarchitectuur

Het hoofddoel van de systeemarchitectuur is het beschrijven van het “skelet” van de applicatie.

Hieronder valt de manier van data opslag, programmeertaal, mogelijke platforms, modules, de

interactie tussen deze modules (inclusief protocol) en de manier van data opslag. Het doel hiervan is dan

ook om een overzicht te geven van de interacties tussen de verschillende modules en hoe deze modules

zich verhouden tot het geheel (in welke “laag” deze zich bevinden). Voor meer details over de software

achter de modules zie hoofdstuk 4 en 5.

### Visualisation

|  |  |
| --- | --- |
| **Module name** | Visualisation |
| **Functie** | Deze module stelt de applicatie in staat een Sunburst, graph of tree te laten bouwen. De module bestaat uit 2 scripts (DiagramBuilder en Tree in UML) die dit realiseren. De module wordt aangeroepen vanuit de Visualisation.wsgi en weergegeven met behulp van CSS en Javascript in HTML. |
| **Samenvatting** | Module die de data visualiseert |
| **Communicatie library** | Geen |
| **Taal** | Python |

### Data storage and retrieval

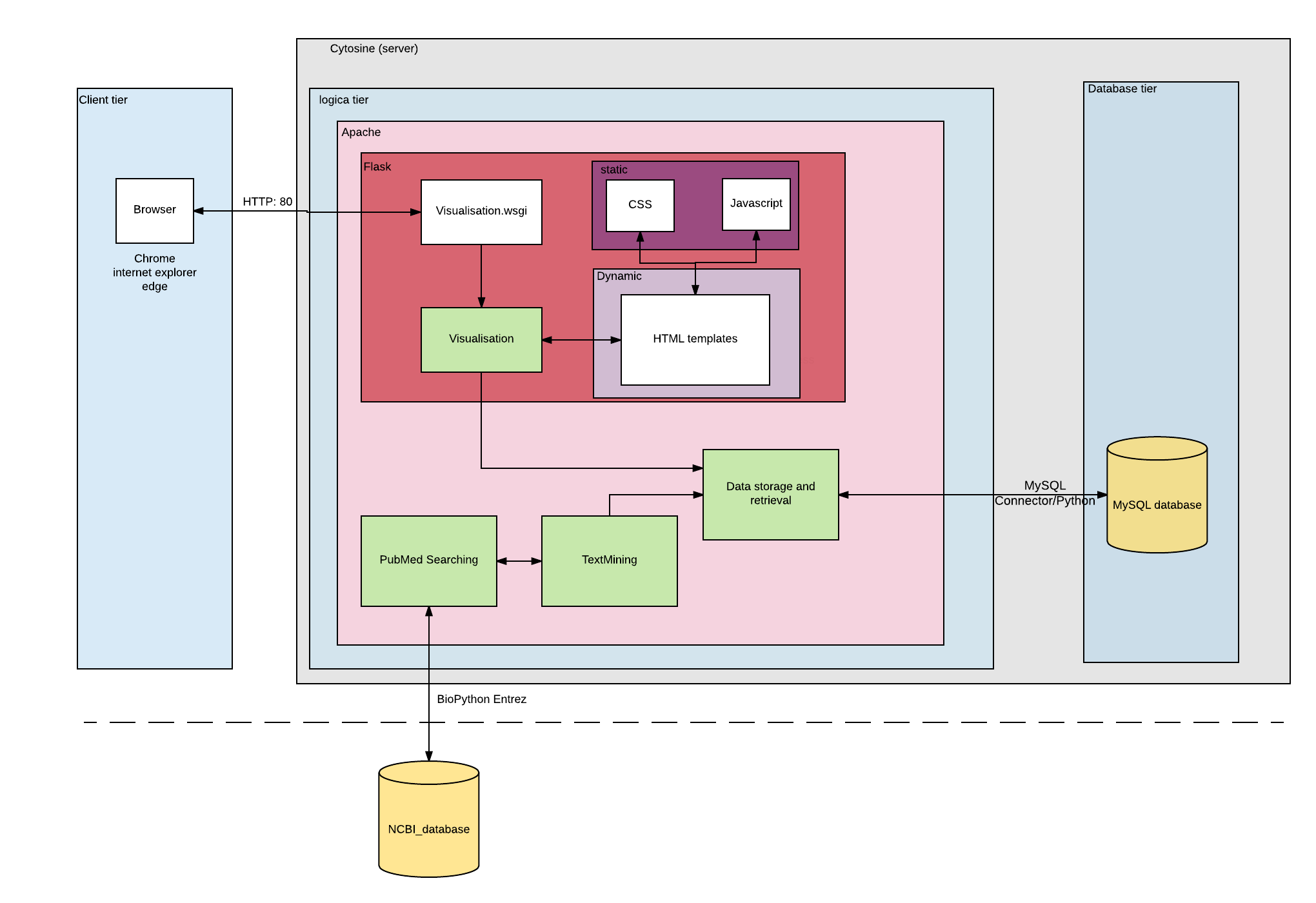
|  |  |
| --- | --- |
| **Module name** | Data storage and retrieval |
| **Functie** | Deze module bestaat uit 2 scripts die ervoor zorgen dat er een connectie met de MySQL database wordt gemaakt (ConnectionHandler) en de queries worden uitgevoerd. De QueryBuilder zorgt ervoor dat deze queries worden gemaakt zodat de applicatie in staat is om de data op te slaan en op te halen uit de MySQL database. De data wordt vervolgens geretourneert naar de visualisatie module. |
| **Samenvatting** | Module die de data verwerkt en kan opslaan in de database |
| **Communicatie library** | MYSQL connector/PYTHON |
| **Taal** | Python |

### Textmining

|  |  |
| --- | --- |
| **Module name** | Textmining |
| **Functie** | Deze module bestaat uit 3 co-operende scripts. (Nadere specificaties te vinden in UML). Deze zijn verantwoordelijk voor het zoeken van genen, organismen en stress condities in de abstracts van de gevonden artikelen. |
| **Samenvatting** | Textmining van artikelen |
| **Communicatie library** | Geen |
| **Taal** | Python |

### PubMed Searching

|  |  |
| --- | --- |
| **Module name** | PubMed Searching |
| **Functie** | Deze module zorgt voor de verbinding met de NCBI PubMed en Gene database en tevens het ophalen van informatie uit deze database. |
| **Samenvatting** | Connectie en doorzoeken van databases |
| **Communicatie library** | Geen |
| **Taal** | Python |



**Figuur 1.** **Systeemarchitectuur**. In deze figuur is de systeem architectuur van de applicatie weergegeven. De module PubMedSearching is voornamelijk verantwoordelijk voor het ophalen van artikelen die betrekking hebben op anthocyanen productie. Vervolgens zal er in de abstracts van deze artikelen worden gezocht naar condities, genen en organismes(TextMining module). Als deze informatie aanwezig is zal deze d.m.v. de “Data storage and retrieval” module worden opgeslagen in MySQL database. De informatie gevonden gedurende deze reeks van stappen kan vervolgens worden bekeken via een website. Echter moet hiervoor eerste de data worden opgehaald uit de database en worden omgezet in een Tree structuur, de onderliggende structuur voor een sunburst, graph en tree. Deze tree structuur wordt vervolgens omgezet in het gewenste diagram en verwerkt in een HTML pagina die geretourneerd wordt naar de client.

## Systeemspecificaties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CoTexThen | Client tier  Lokaal | | HTML5 compatibel browser | N.V.T. |
|
|
| Logica tier Cytosine.nl | | Python compatibel OS | N.V.T. |
| Apache | 2.2.22 |
| Python | 3.6.1 |
|
|  | Visualisation | Flask | 0.12.2 |
| PubMed Searching | BioPython | 1.69 |
| TextMining | NLTK | 3.2.3 |
| Retina-sdk | 1.0.0 |
| Data storage and retrieval | MySQL Connector/Python | 2.1.6 |
|
| Database tier Cytosine.nl | | MySQL compatibel OS | N.V.T. |
| MySQL | 5.7.18 |
|
|  | |
|
|

# Software-architectuur

## Doel van de software-architectuur

Deze architectuur beschrijft de opbouw van de software. Hoewel de systeem architectuur beschrijft hoe

verschillende modules interactie met elkaar aangaan ligt de focus bij de software architectuur op de

opbouw van deze modules: welke classes gebruikt zullen worden om de functies van de modules te

realiseren, hoe deze classes zijn opgebouwd en hoe de verschillende classes samenhangen.

## UML Class Diagram(s)

Het UML class diagram beschrijft de interacties tussen de verschillende classes. In dit geval worden de classes weergeven die de textmining uitvoeren, de classes die verbinding maken met de database en de classes die betrokken zijn bij het visualiseren van de gevonden data op de webpagina.

#### ConditionSearcher

Deze class is verantwoordelijk voor het zoeken van condities in de abstracts van de artikelen. Dit wordt gedaan op basis van overeenkomst met zinnen (zie TextMiner) die de gewenste opbouw hebben.

#### PubMedSearcher

Deze class is verantwoordelijk voor het connecteren met de NCBI PubMed en Gene database en tevens het zoeken van informatie in deze databases.

#### GeneSeacher

Deze class is verantwoordelijk voor het zoeken van genen in de abstracts. Dit wordt gedaan door herkenning van woorden in hoofdletters + controle in de NCBI gene database (via PubMedSeacher).

#### TextMiner

Deze class is verantwoordelijk voor het bouwen van modellen op basis van een training set, het vergelijken van tekst met deze modellen en het zoeken naar substrings.

#### QueryBuilder

Deze class is verantwoordelijk voor het opbouwen van queries. Dit kunnen queries zijn om data op te slaan of om data op te halen uit de database.

#### ConnectionHandler

Deze class is verantwoordelijk voor het openen van de connectie met de database, het uitvoeren van queries en het sluiten van de connectie met de database.

#### DiagramBuilder

Deze class is verantwoordelijk voor het ophalen van data uit de database, het scheiden van deze data met een bepaalde delimiter en vervolgens het omzetten van deze data in een JSON tree (m.b.v de Tree class).

#### Tree

Deze class is verantwoordelijk voor het omzetten van data gescheiden door een bepaalde delimiter (zoals: “;”) naar een JSON tree.

#### 

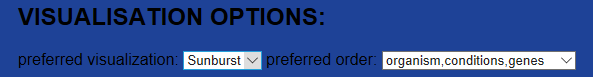
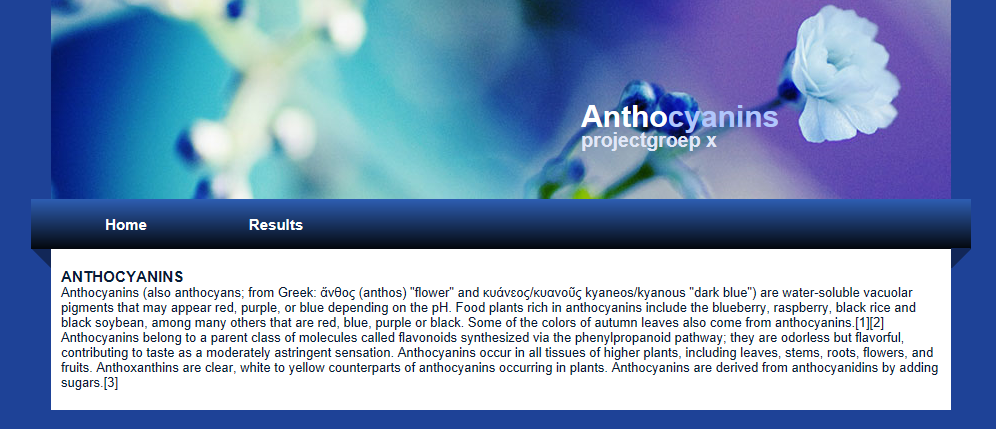
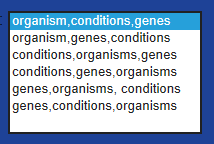
**Figuur 2. UML Class diagram. I**n dit class diagram zijn de classes die gebruikt worden om de functies van de modules te realiseren weergegeven. De titels van de verschillende package (lichtblauw, lichtgrijs, roze en groen) komen overeen met de titels van de modules in de systeem architectuur. In dit class diagram is ook een niet-class script opgenomen. Dit om duidelijk te maken dat alle classes afgebeeld in class diagram worden aangeroepen vanuit het main.py script.

# Lay-out web applicatie

## Doel van de Lay-out web applicatie

In dit hoofdstuk wordt beschreven hoe de website eruit gaat zien en hoe de gebruiker de resultaat weergaven naar wens kan aanpassen. Op basis hiervan kunnen de ontwerpers en de gebruiker de mooiste en gebruiksvriendelijkste lay-out bepalen.

## Basis lay-out



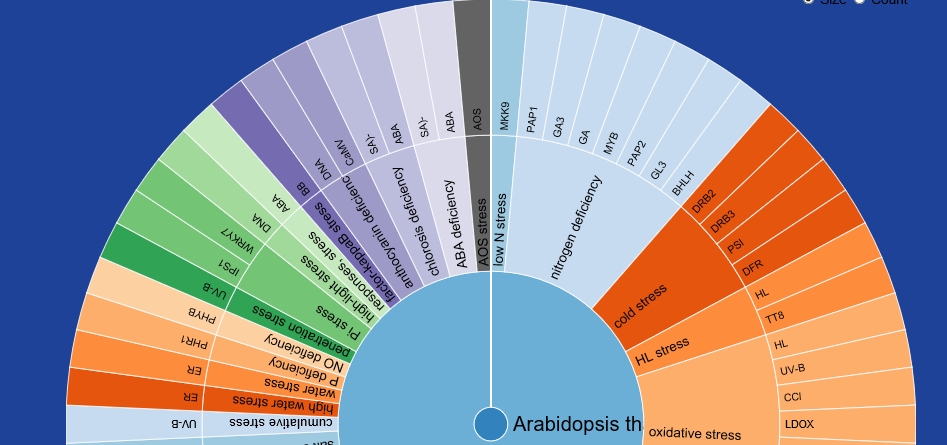
**Figuur 3.** **Basis lay-out**. Deze figuur laat de basis lay-out van de website zien. Voor een voorbeeld van de resultaat weergaven zie figuren 4 en 5.

## Voorbeelden

#### Sunburst

Dit type van visualisatie geeft een hiërarchische weergave van de organismes, condities en genen (of een andere gewenste volgorde). Elke ring in het sunburst diagram is opgedeeld in verschillende categorieën. Zo stelt de eerste ring de organismes voor en is opgedeeld in bijvoorbeeld *Arabidopsis Thaliana*, *Zea mays*, etc. Verder kan er eenvoudig worden ingezoomd op elke categorie in een ring. Een voorbeeld hiervan is weergegeven in de onderstaand figuur, hierbij is er ingezoomd op *Arabidopsis Thaliana.*



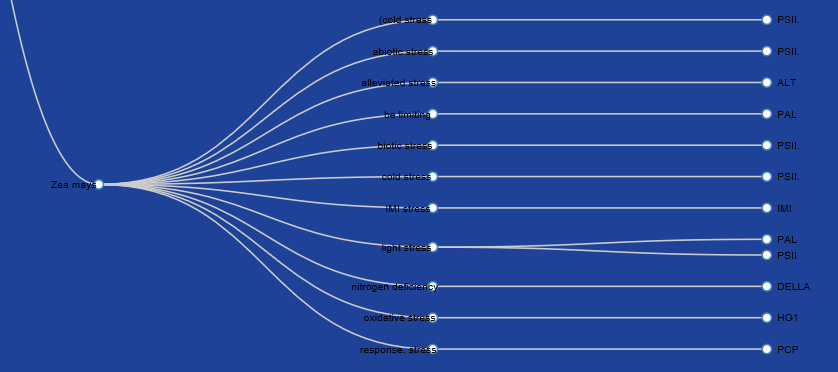


**Figuur 4.** **Voorbeeld weergaven sunburst.** Deze figuur laat het sunburst diagram zien dat gebruikt zal worden voor de visualisatie van de data. Links in de figuur is de volledige sunburst weergegeven en rechts is er ingezoomd op “Arabidopsis Thaliana”.

#### 

#### Tree

Dit type van visualisatie geeft ook een hiërarchische weergave van de organismes, condities en genen (of een andere gewenste volgorde). Echter is deze weergaven niet circulair zoals bij de sunburst. Rechts is de uitgezoomde tree te zien en in de afbeelding hieronder is er ingezoomd op *Zea mays*.



**Figuur 5.** **Voorbeeld weergaven tree.** Deze figuur laat het tree diagram zien dat gebruikt zal worden voor de visualisatie van de data. Rechts in de figuur is de volledige tree weergegeven en links is er ingezoomd op “Zea mays”.

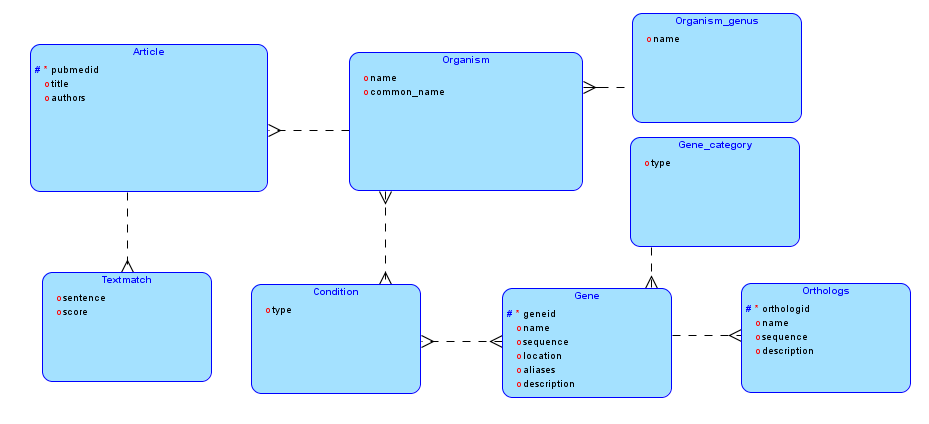
# Technische gegevens structuur

## Doel van de technische gegevens structuur

Het doel van de technische gegevensstructuur is het beschrijven van de structuur van de MySQL database. Hierin wordt beschreven: de samenhang tussen de entiteiten (later tabellen in de database), de data in deze entiteiten (de attributen), de relatie tussen de verschillende entiteiten en de kardinaliteit. Deze technische gegevens structuur kan gevisualiseerd worden als logisch of als technisch model (zie hieronder).

## Logisch model

#### Hoewel het logische model alle bovengenoemde punten beschrijft zijn de meeste technische details weggelaten. Het logische model kan vanwege dit tamelijke hoge abstractie niveau uitstekend gebruikt worden als communicatiemiddel met de opdrachtgever. Hierdoor krijgt de opdrachtgever inzicht in de database en krijgt zo de mogelijkheid om eventuele vak specifieke aanpassingen te maken (bijvoorbeeld welke data van de gebruikte artikelen opgeslagen moet worden).



**Figuur 6: logisch ERD van de MySQL database**. De database bestaat uit acht entiteiten: Article, Textmatch, Organism, Organism\_genus, Condition, Gene, Gene\_category en Orthlogs.De inhoud van deze entiteiten wordt in de tekst hieronder nader toegelicht. Omdat één artikel meerdere keren een matchende zin kan bevatten is er een 1:N relatie tussen Article en Textmatch, één organisme kan ook in verschillende artikelen voorkomen dus er is ook hier een 1:N relatie. Tussen Organism\_genus en Organism bestaat ook een 1:N relatie omdat er hoogstwaarschijnlijk meerdere organismes gevonden zullen worden die tot één genus toebehoren. Een stress conditie kan bij meerdere organismes en genen gevonden worden, omgekeerd kan een organisme of gen ook bij meerdere stress omstandigheden gevonden worden. Er is dus een M:N relatie tussen zowel Organism en Condition als Gene en Condition. Er kunnen meerdere orthlogen gevonden worden voor een enkel gen, hier is dus een 1:N relatie aanwezig. Meerdere genen kunnen in één gen categorie vallen, ook hier is er dus een 1:N relatie.

#### Article

De titel, PubMed id en auteurs van ieder artikel waarin de text miner een match vindt komen hier terecht.

#### Textmatch

De zinnen die volgens de text miner significant overeenkomen met de modelzinnen worden hier met hun bijbehorende significantie score opgeslagen.

#### Organism

Zal de naam en de volksnaam van organisms waarvan relevante genen zijn gevonden bevatten.

#### Organism\_genus

Zal de genus van de bovengenoemde organismes bevatten.

#### Condition

Wordt gevuld met de verschillende stress omstandigheden die tot anthocyanen productie leiden.

#### Gene

Als de text miner een gen identificeert wordt die hier opgeslagen met bijbehorende genid (van NCBI Gene), naam, DNA sequentie, genomische locatie, bekende aliases en een korte omschrijving.

#### Gene\_category

Van de gevonden genen wordt hier het soort gen (bijvoorbeeld “eiwit coderend”) opgeslagen.

#### Orthologs

Van de gevonden genen worden hier bekende orthologen in andere planten opgeslagen.

## Technisch model

**Figuur7: technisch ERD van de MySQL database.** Zie figuur 2 en bovenstaande tekst voor uitleg van de tabellen en relaties.

# Begrippenlijst

|  |  |
| --- | --- |
| Begrip | Betekenis |
| ERD | ERD staat voor entity-relationship diagram. Een ERD geeft de structuur van de database weer: entiteiten, attributen, relaties en regels die gelden (bijvoorbeeld of een relatie of attribuut verplicht is). |
| kardinaliteit | Geeft het aantal elementen in een verzameling weer. |
| BioPython | Een Python library die gebruikt kan worden voor het oplossen van algemene biologische vraagstukken. |
| Library | Verzameling van scripts |
| UML | UML staat voor unified Modeling Language. De Unified Modeling Language, is een taal om objectgeoriënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken. |
| Tier | Een tier verwijst naar de basis van een applicatie. Voorbeelden zijn client tier en server tier. |
| Platform | Dit verwijst naar de basis waarop software ontwikkeld wordt. Dit kan bijvoorbeeld Windows zijn. |
| Module | Een op zichzelf staand stuk code dat een specifieke functie uitgevoerd. |
| Protocol | Een set van regels die bepaalde delen van een systeem in staat stellen om informatie uit te wisselen. |
| NCBI | National Center for Biotechnology Information, beheert onder andere een database die weer bestaat uit een verzameling van andere databases zoals Gene, Protein, Entrez, etc. |
| Class | Een blauwdruk voor objecten van hetzelfde typen. |
| Attributen | Dit worden de kolommen van de bijbehorende tabel in de database |
| Entiteit | Verwijst naar een verzameling van data. In het ERD perspectief staat dit gelijk aan de tabellen die gecreëerd zullen worden in de database. |
| Primary key | Dit is een kolom waarin waardes staat die identificeren zijn voor elke rij in en de tabel. |
| Foreign key | Een waarde in een tabel die kan worden gebruikt als unieke identificatie voor een rij uit een andere tabel. |
| mySQL | Een open source taal die gebruikt wordt voor het opvragen van data uit een database en voor het opslaan van data in een database. |
| CSS | Cascading Style Sheets, bieden opmaak mogelijkheden voor de webpagina(’s).. |
| Javascript | JavaScript is een scripttaal om webpagina's interactief te maken. In dit geval om het diagram aan de client zijde te kunne updaten. |
| HTML | HyperText Markup Language, opmaaktaal voor het wereldwijde web. |
| Python | Hoog-niveau programmeertaal. |
| PubMed | Online database beheerd door NCBI met daarin biomedische literatuur |
| HTTP | Hypertext Transfer Protocol (HTTP) is het protocol voor de communicatie tussen een webclient (meestal een webbrowser) en een webserver. Dit vindt meestal plaats via poort 80. |
| MySQL | Relationele database management systeem gebaseerd op SQL. |

# Bronvermelding

Kaula, R.(2007). Normalizing with Entity Relationship Diagramming. Geraadpleegd op 13-03-2017 van <http://tdan.com/normalizing-with-entity-relationship-diagramming/4583>

Warmer, J., Kleppe, A.(2011). Praktisch UML. Amsterdam. Pearson Benelux.