|  |  |
| --- | --- |
| <<Projectgroep 1>> | |
| Document | Ontwerp.doc | |
| Datum | 01-01-01 | |
| Versie | 1.0 | |
| Groepsleden | Groepslid 1 | |
|  | Groepslid 2 | |
|  | Groepslid 3 | |
|  | Groepslid 4 | |

CoTexThen

<<ondertitel/type document>>

# 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. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Houd hier versiebeheer bij van dit document.

Contents

[Versiebeheer 2](#_Toc475188391)

[1. Introductie 4](#_Toc475188392)

[Aanleiding 4](#_Toc475188393)

[Doel van dit document 4](#_Toc475188394)

[Doel van de applicatie 4](#_Toc475188395)

[Doelgroep 4](#_Toc475188396)

[2. Project beschrijving 5](#_Toc475188397)

[Startpunt van het project 5](#_Toc475188398)

[Gekozen technieken 5](#_Toc475188399)

[Design principes 5](#_Toc475188400)

[Design patterns 5](#_Toc475188401)

[3. Systeemarchitectuur 6](#_Toc475188402)

[Doel van de systeemarchitectuur 6](#_Toc475188403)

[Gedetailleerde systeemarchitectuur 6](#_Toc475188404)

[<<Module 1>> 7](#_Toc475188405)

[<<Module 2>> 7](#_Toc475188406)

[<<Module 3>> 7](#_Toc475188407)

[<<Module 4>> 7](#_Toc475188408)

[Systeemspecificaties 7](#_Toc475188409)

[4. Software-architectuur 8](#_Toc475188410)

[Doel van de software-architectuur 8](#_Toc475188411)

[UML Class Diagram(s) 8](#_Toc475188412)

[5. Technische gegevens structuur 10](#_Toc475188413)

[Doel van de technische gegevens structuur 10](#_Toc475188414)

[Logisch model 10](#_Toc475188415)

[Technisch model 11](#_Toc475188416)

[6. Begrippenlijst 12](#_Toc475188417)

[Bronvermelding 13](#_Toc475188418)

# 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 plaatsvinden door gebruik te maken van MySQL Connector/Python

De visualisatie zelf vindt plaats door de gebruiker webpagina’s te retouneren bestaande uit HTML, CSS en JavaScript vanuit de Python scripts om overzichtelijke en esthetisch verantwoorde diagrammen te tonen.

## 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.

## <<Module 1>>

Beschrijf wat deze module doet, verantwoordelijkheden? Taal? Protocol?

### <<Module 2>>

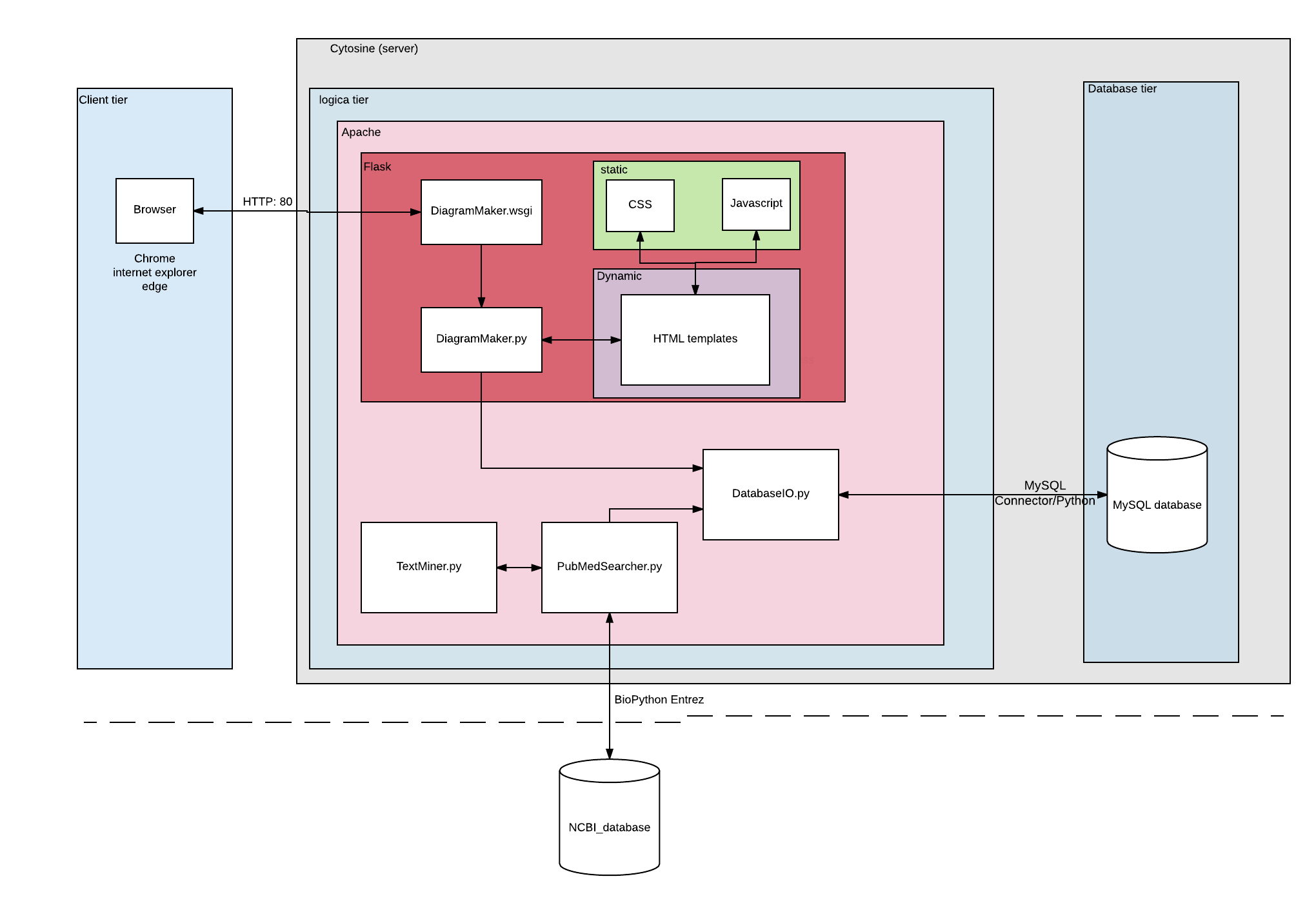
Beschrijf wat deze module doet, verantwoordelijkheden? Taal? Protocol?

### <<Module 3>>

Beschrijf wat deze module doet, verantwoordelijkheden? Taal? Protocol?

### <<Module 4>>

Beschrijf wat deze module doet, verantwoordelijkheden? Taal? Protocol?



## Systeemspecificaties

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| <<Applicatie>> | <<Deelsysteem 1>> <<welk fysiek apparaat, adres?>> | | <<software>> | <<versienr.>> |
|  |  |
|  |  |
|  |  |
|  | <<Module 1>> |  |  |
| <<Module 2>> |  |  |
| <<Module 3>> |  |  |
| <<Deelsysteem 2>> <<welk fysiek apparaat, adres?>> | |  |  |
|  |  |
|  |  |
|  | <<Module 4>> |  |  |
| <<Module 5>> |  |  |
|  |  |  |

Beschrijf in de tabel hierboven de systeemspecificaties van alle onderdelen uit de te ontwikkelen software. Niet de software om je product mee te maken (zoals IDE’s), maar software noodzakelijk om je applicatie te “draaien” (bv, java installaties). Maak de tabel zo compleet mogelijk!

# 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 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.

#### 

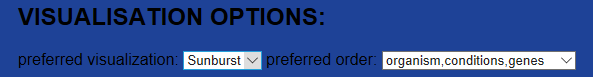
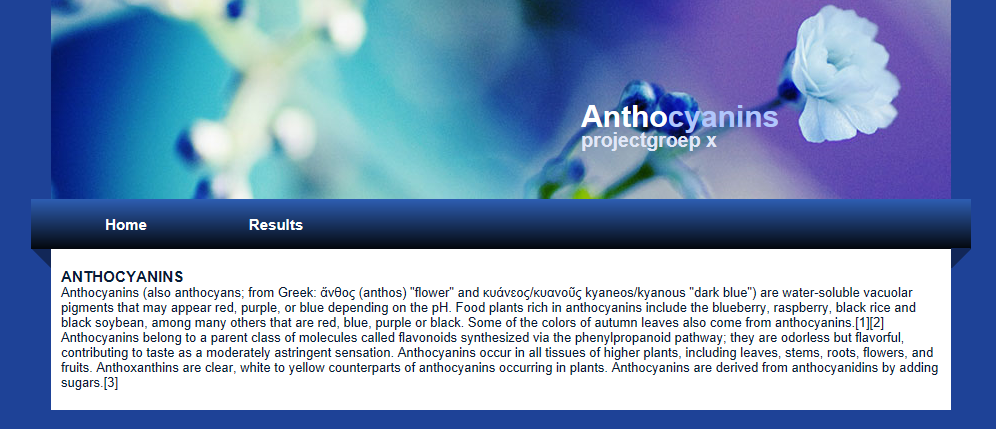
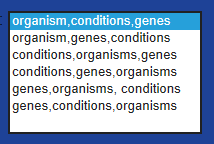
**Figuur 1. UML Class diagram. I**n dit class diagram zijn de classes die gebruikt worden om de functies van de modules te realiseren weergegeven.

# 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



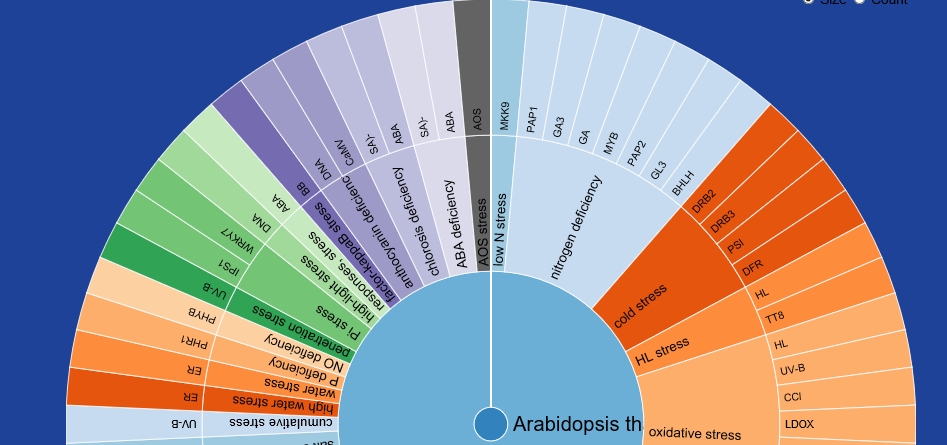
**Figuur 3.** **Basis lay-out**. Deze figuur laat de basis lay-out van de website zien. Voor een voorbeeld van de resultaat weergaven zie figuren 4 en 5.

## Voorbeelden

#### Sunburst

Dit typen van visualisatie geeft een hiërarchische weergaven van de organisme, 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 *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 in is gezoomd op *Arabidopsis Thaliana.*



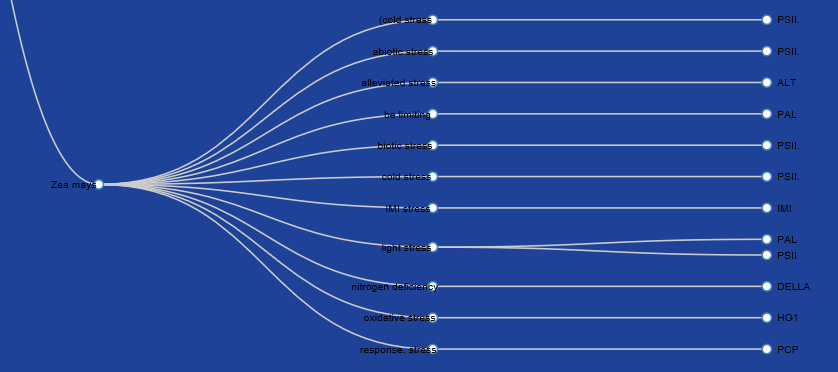


**Figuur 4.** **Voorbeeld weergaven sunburst.** Deze figuur laat het sunburst diagram zien dat gebruikt zal worden voor de visualisatie van de data. Links in de figuur is de volledige sunburst weergegeven en rechts is er ingezoomd op “Arabidopsis Thaliana”.

#### 

#### Tree

Dit typen van visualisatie geeft ook een hiërarchische weergaven van de organisme, 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*.



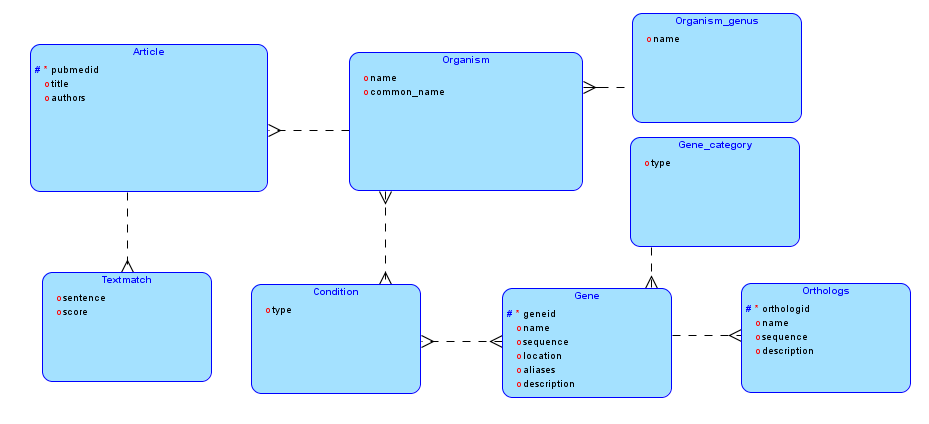
**Figuur 5.** **Voorbeeld weergaven tree.** Deze figuur laat het tree diagram zien dat gebruikt zal worden voor de visualisatie van de data. Rechts in de figuur is de volledige tree weergegeven en links is er ingezoomd op “Zea mays”.

# Technische gegevens structuur

## Doel van de technische gegevens structuur

Hier wordt beschreven hoe de geplande MySQL database van de applicatie gestructureerd zal worden. De tabellen waarin de data wordt opgeslagen en hun relaties worden hier toegelicht, de attributen in deze tabellen en hun data types worden hier ook behandeld.

## Logisch model

 Het logische model geeft een globaal overzicht van de tabellen en hun relaties, hier wordt nog niet specifiek ingegaan op de attributen.

Figuur 2: 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

Figuur 3: technisch ERD van de MySQL database. Zie figuur 2 en bovenstaande tekst voor uitleg van de tabellen en relaties.

# Begrippenlijst

|  |  |
| --- | --- |
| Begrip | Betekenis |
| ORF | Open reading frame |
|  | --ook technische begrippen-- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 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.