PERSOONLIJK DOSSIER

Tom Adriaens – Oman Biodiversity

TOM ADRIAENS
2014-2015

Project werk
CVO ANTWERPEN – CAMPUS HOBOKEN
Informatica – specialisatie programmeren

Inhoud

Inleiding

Zoals u in het algemeen dossier hebt kunnen lezen was de opdracht met een begeleidende website te maken voor een tentoonstelling over de biodiversiteit van Oman. De tentoonstelling werd vooral bezocht door scholen en had een doelpubliek van 8 tot 12 jarigen.

De informatie die over de tentoongestelde dieren werd en word ingevuld door wetenschappers en stagiaires en deze moest op een dynamische wijze op de website komen. Hier moesten we ook rekening houden dat deze informatie moest gevalideerd worden.

De dieren moesten een logische samenhang hebben zo bestond de tentoonstelling uit 4 vier delen de leefwerelden waar de dieren in voorkomen namelijk planten, landdieren, microben en zeedieren. De dieren behoren tot een familie en een variëteit. Deze komen dan weer voor in een bepaald gebied in Oman en mogelijk in een bepaald seizoen.

De website moest ook interactief zijn zo moesten gebruikers de mogelijkheid hebben te posten op de website waar een bepaald dier gespot werd maar hier heeft de opdrachtgever achteraf vanaf gezien en gekozen om interactie te hebben met de sociaal media zoals Facebook en Instagram. Bezoekers aan de website moeten de mogelijkheid hebben zich in te schrijven voor een nieuwsbrief.

De keuze van de backend taal is JAVA geworden met een databank in MySql. Onze groep had net de cursus programmeren 4 JAVA achter de rug en de host van de opdrachtgever ondersteunde enkel PHP waar niemand van onze groep ervaring mee had. We zijn toen in opdracht van onze de opdrachtgever op zoek gegaan naar een andere host en kwamen tot de ontdekking dat hosting met C# en een Sql databank zeer duur zou zijn. Tijdens de verdere verloop van het project hebben we daar goedkopere oplossingen voor gevonden maar we stonden van in het begin van het project onder tijdsdruk dus was de keuze gemaakt.

In de frontend hebben we gekozen om Bootstrap te gebruiken. 100% responsive design was een must omdat er op de tentoonstelling gebruik werd gemaakt van QR codes om meer informatie op te vragen over de tentoongestelde objecten.

Voor mij was het hele project een heel steile leercurve de meeste van de gebruikte technieken zijn wel aanbod gekomen tijdens de studies maar zeker niet in deze mate. Ik heb degelijk moeten bij schaven Jquery, Ajax, javascript en Boostrap maar het was zeker de moeite waard. Ik dank al zeker mijn teamgenoten de mij geweldig gesteund hebben en me veel en herhaaldelijk uitleg gegeven hebben.

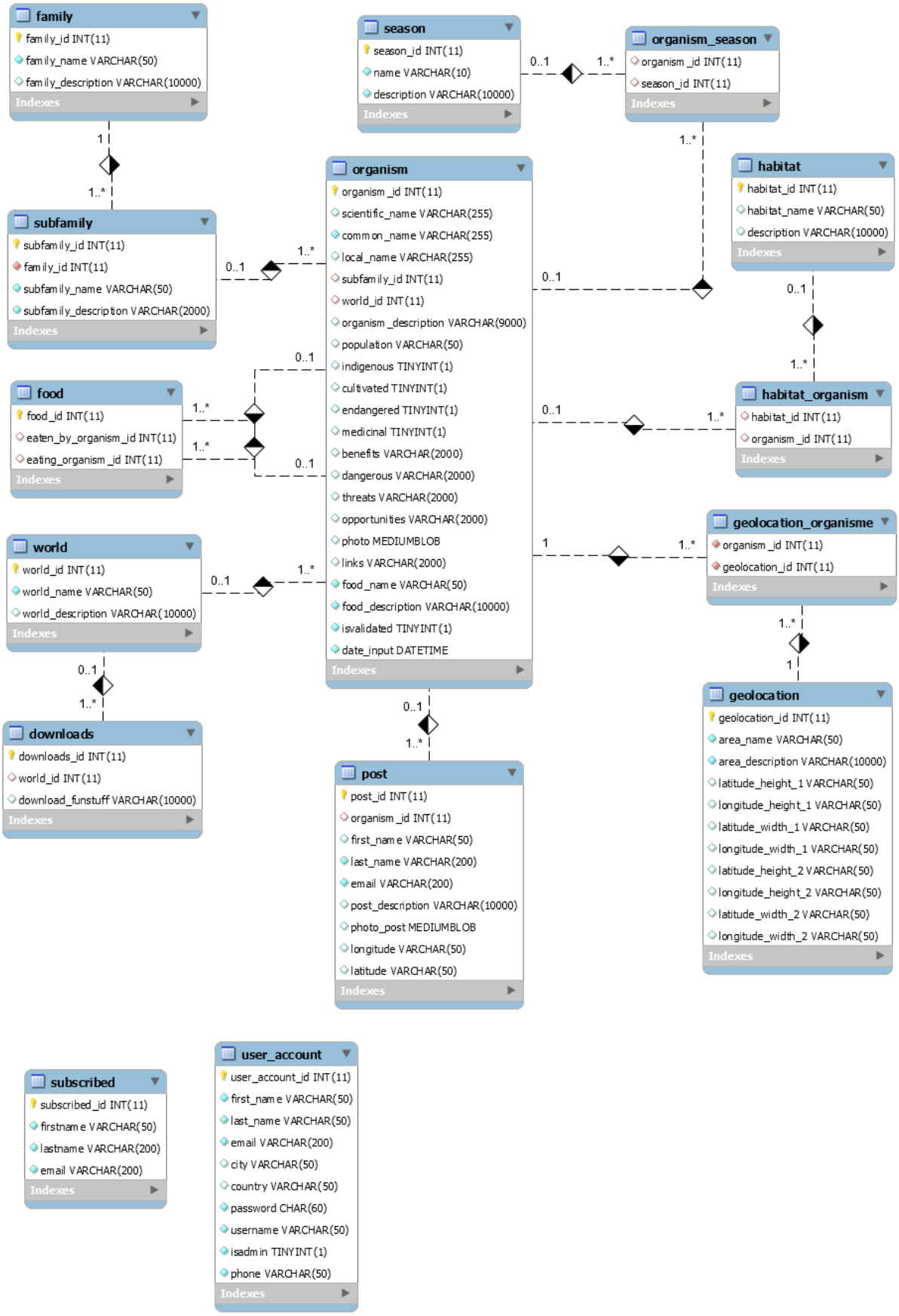
Databanken

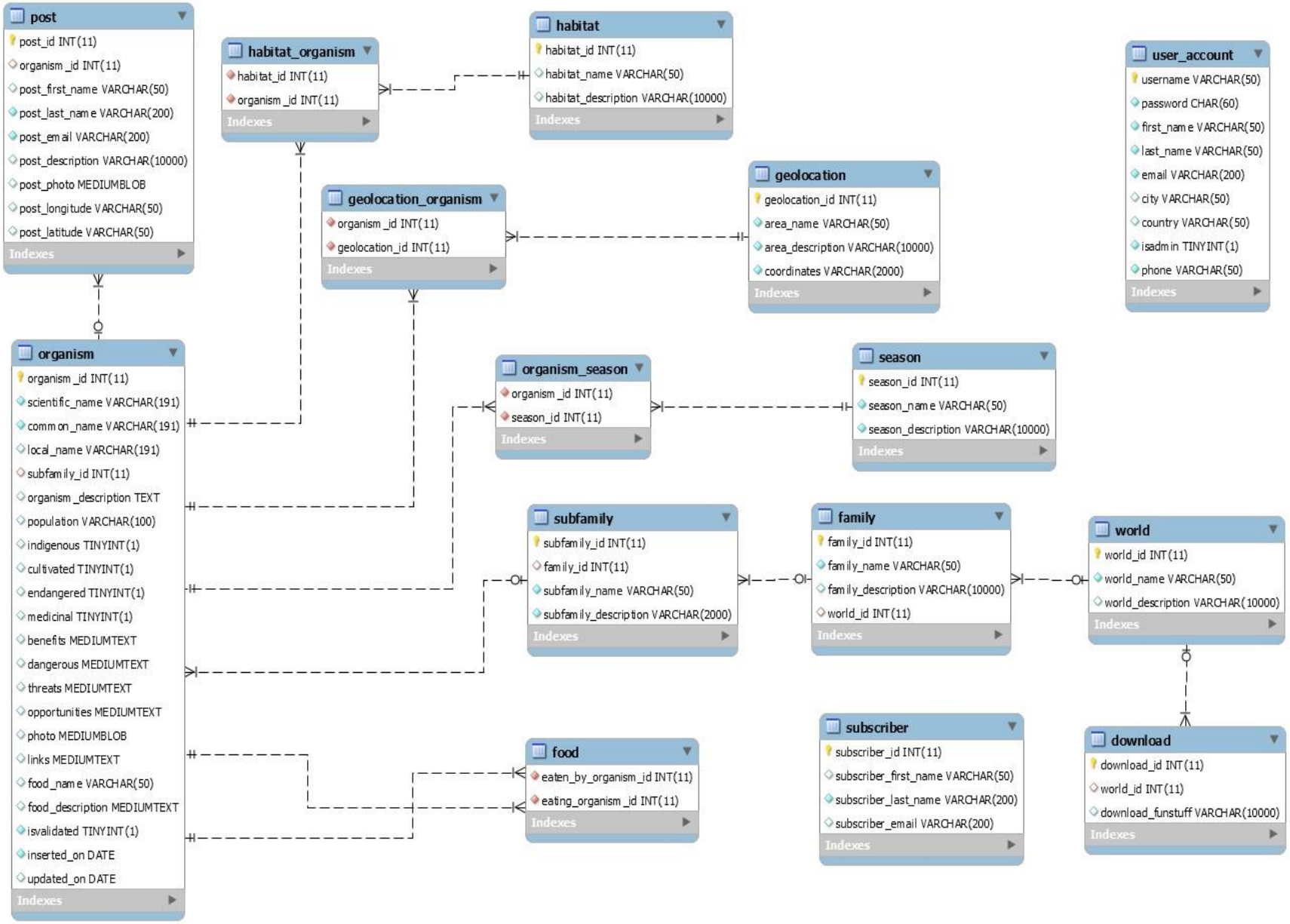
De opstelling van de databank was niet makkelijk. Dat we voor de web applicatie een login systeem moesten hebben met een gewone gebruiker en administratieve gebruiker met alle rechten lag voor de hand. De tabel voor de nieuwsbrief was ook gemakkelijk, maar de samenhang van de dieren was net iets moeilijker.

Lange en heftige discussies waren het gevolg. Hier heb ik heel goed leren omgaan met phpMyAdmin en Workbench, om de discussies te verkorten heb ik telkens de voorgestelde verandering met phpMyAdmin zonder omslachtig invoeren van code kon ik snel veranderingen aanbrengen en dan via Workbench een databank diagram gemaakt. Aan deze diagrammen werd snel herkend waar nog velden moesten toegevoegd worden of welke overbodig waren als ook relaties die vereenvoudigd konden worden of eerder een many to many relatie moesten hebben. Zo zijn er zeker een dertigtal voorstellen de revue gepasseerd voor we aan ons uiteindelijke resultaat zijn gekomen. Natuurlijk was phpMyAdmin niet voldoende om alle problemen op te lossen en zo zijn er natuurlijk aan het Sql script nog manuele veranderingen aangebracht. Zeker waar relaties moesten bewerkt worden zoals bv. cascade on delete.

Het resultaat is een genormaliseerde databank misschien wel tot op het punt waar u zal denken dat het overdreven is maar wij moesten er wel rekening mee houden dat deze website nog in het Arabisch word vertaald. Zo hebben we er bv. ook de werelden of seizoenen ingestoken terwijl dit ook hard coded gekund had.

Hieronder vind u het eerste en het uiteindelijk databank design.


family
family_id INT(11)
family_name VARCHAR(50)
family_description VARCHAR(10000)
Indexes
season
season_id INT(11)
name VARCHAR(10)
description VARCHAR(10000)
Indexes
organism_season
organism_id INT(11)
season_id INT(11)
Indexes
subfamily
subfamily_id INT(11)
family_id INT(11)
subfamily_name VARCHAR(50)
subfamily_description VARCHAR(2000)
Indexes
organism
organism_id INT(11)
scientific_name VARCHAR(255)
common_name VARCHAR(255)
local_name VARCHAR(255)
subfamily_id INT(11)
world_id INT(11)
organism_description VARCHAR(9000)
population VARCHAR(50)
indigenous TINYINT(1)
cultivated TINYINT(1)
endangered TINYINT(1)
medicinal TINYINT(1)
benefits VARCHAR(2000)
dangerous VARCHAR(2000)
threats VARCHAR(2000)
opportunities VARCHAR(2000)
photo MEDIUMBLOB
links VARCHAR(2000)
food_name VARCHAR(50)
food_description VARCHAR(10000)
isvalidated TINYINT(1)
date_input DATETIME
Indexes
habitat
habitat_id INT(11)
habitat_name VARCHAR(50)
description VARCHAR(10000)
Indexes
food
food_id INT(11)
eaten_by_organism_id INT(11)
eating_organism_id INT(11)
Indexes
habitat_organism
habitat_id INT(11)
organism_id INT(11)
Indexes
geolocation_organisme
organism_id INT(11)
geolocation_id INT(11)
Indexes
world
world_id INT(11)
world_name VARCHAR(50)
world_description VARCHAR(10000)
Indexes
geolocation
geolocation_id INT(11)
area_name VARCHAR(50)
area_description VARCHAR(10000)
latitude_height_1 VARCHAR(50)
longitude_height_1 VARCHAR(50)
latitude_width_1 VARCHAR(50)
longitude_width_1 VARCHAR(50)
latitude_height_2 VARCHAR(50)
longitude_height_2 VARCHAR(50)
latitude_width_2 VARCHAR(50)
longitude_width_2 VARCHAR(50)
Indexes
downloads
downloads_id INT(11)
world_id INT(11)
download_funstuff VARCHAR(10000)
Indexes
post
post_id INT(11)
organism_id INT(11)
first_name VARCHAR(50)
last_name VARCHAR(200)
email VARCHAR(200)
post_description VARCHAR(10000)
photo_post MEDIUMBLOB
longitude VARCHAR(50)
latitude VARCHAR(50)
Indexes
subscribed
subscribed_id INT(11)
firstname VARCHAR(50)
lastname VARCHAR(200)
email VARCHAR(200)
Indexes
user_account
user_account_id INT(11)
first_name VARCHAR(50)
last_name VARCHAR(50)
email VARCHAR(200)
city VARCHAR(50)
country VARCHAR(50)
password CHAR(60)
username VARCHAR(50)
isadmin TINYINT(1)
phone VARCHAR(50)
Indexes


post
post_id INT(11)
organism_id INT(11)
post_first_name VARCHAR(50)
post_last_name VARCHAR(200)
post_email VARCHAR(200)
post_description VARCHAR(10000)
post_photo MEDIUMBLOB
post_longitude VARCHAR(50)
post_latitude VARCHAR(50)
Indexes
habitat_organism
habitat_id INT(11)
organism_id INT(11)
Indexes
habitat
habitat_id INT(11)
habitat_name VARCHAR(50)
habitat_description VARCHAR(10000)
Indexes
user_account
username VARCHAR(50)
password CHAR(60)
first_name VARCHAR(50)
last_name VARCHAR(50)
email VARCHAR(200)
city VARCHAR(50)
country VARCHAR(50)
isadmin TINYINT(1)
phone VARCHAR(50)
Indexes
geolocation_organism
organism_id INT(11)
geolocation_id INT(11)
Indexes
geolocation
geolocation_id INT(11)
area_name VARCHAR(50)
area_description VARCHAR(10000)
coordinates VARCHAR(2000)
Indexes
organism
organism_id INT(11)
scientific_name VARCHAR(191)
common_name VARCHAR(191)
local_name VARCHAR(191)
subfamily_id INT(11)
organism_description TEXT
population VARCHAR(100)
indigenous TINYINT(1)
cultivated TINYINT(1)
endangered TINYINT(1)
medicinal TINYINT(1)
benefits MEDIUMTEXT
dangerous MEDIUMTEXT
threats MEDIUMTEXT
opportunities MEDIUMTEXT
photo MEDIUMBLOB
links MEDIUMTEXT
food_name VARCHAR(50)
food_description MEDIUMTEXT
isvalidated TINYINT(1)
inserted_on DATE
updated_on DATE
Indexes
organism_season
organism_id INT(11)
season_id INT(11)
Indexes
season
season_id INT(11)
season_name VARCHAR(50)
season_description VARCHAR(10000)
Indexes
subfamily
subfamily_id INT(11)
family_id INT(11)
subfamily_name VARCHAR(50)
subfamily_description VARCHAR(2000)
Indexes
family
family_id INT(11)
family_name VARCHAR(50)
family_description VARCHAR(10000)
world_id INT(11)
Indexes
world
world_id INT(11)
world_name VARCHAR(50)
world_description VARCHAR(10000)
Indexes
food
eaten_by_organism_id INT(11)
eating_organism_id INT(11)
Indexes
subscriber
subscriber_id INT(11)
subscriber_first_name VARCHAR(50)
subscriber_last_name VARCHAR(200)
subscriber_email VARCHAR(200)
Indexes
download
download_id INT(11)
world_id INT(11)
download_funstuff VARCHAR(10000)
Indexes

Use-casses

Op één na heb ik use cases voor mijn rekening genomen, deze vindt u terug in het algemeen dossier. Als oudste van de groep heb ik de meeste jaren als technisch analfabeet achter de rug. De meeste CRUD use casses zijn zoals verwacht enkel met de Delete functies heb ik erop aangedrongen dat bij World, Family en Breed dubbele bevestigingen werden toegevoegd met heel duidelijke boodschappen wat er gebeurd als deze verwijderd worden. Ook vond ik het belangrijk om te vermijden dat de laatste administratieve gebruiker kon worden verwijderd. Terwijl voor ons het gevolg duidelijk was vond ik dat voor gewone gebruikers deze samenhang en het gevolg niet voor de hand liggend zijn maar wel nefast voor de databank.

Data Access Layer

Hier hebben we nagedacht om Hibernate, JPA of JDBC. Na een lang overleg hebben we gekozen voor JDBC met als belangrijkste reden dat dit door alle teamleden gekend was. Ik heb hier DaPost, DaWorld, DaSubscriber, DaFamily, DaHabitat en DaSeason voor mijn rekening genomen.

Elke DAL Klasse bevat minstens:

- insert
- delete
- selectAll
- selectOneById
- update
- checkIfExist

```
public static void insert(World world) throws SQLException {

        try {
                conn = DataSource.getConnection();
                conn.setAutoCommit(false);
                stmt = conn.prepareStatement("INSERT INTO world (world_name, world_description) VALUES (?,?)");
                stmt.setString(1, world.getWorldName());
                stmt.setString(2, world.getDescription());
                stmt.executeUpdate();
                conn.commit();
        } catch (SQLException ex) {
                conn.rollback();
                System.out.println(ex.getMessage());
        } finally {
                conn.setAutoCommit(true);
                conn.close();
        }
}
```

Om verbinding te maken met de databank hebben we gebruik gemaakt van de klasse DataSource met de statische methode getConnection() deze hebben we telkens opnieuw gebruikt in al onze DAL klasse. Ook hebben we AutoCommit pas aangezet als de hele Query doorlopen is. In de bovenstaande insert methode is het misschien overbodig maar zeker niet in een selectAll() Query.

Rollback zorgt ervoor dat alle ingevoegde data terug wordt gezet moest er iets foutlopen in de Query en alle gebruikte resources weer vrij gemaakt worden.

Om Sql-injection op dit niveau te vermijden hebben we er opgelet dat dat alle parameters gebonden zijn doormiddel van een vraagteken.

Boodschappen doorgeven naar de frontend gebeurd in de Service, zo word daar een daar een fout met de databank opgevangen met een Sql Exception vertaald in een leesbare boodschap voor de gebruiker. Deze zit in elke methode in gebakken maar om te kijken of een object reeds bestaat moest een extra methode geschreven worden namelijk checkIfExists.

```
public static boolean checkIfExists(String worldName) throws SQLException {

        boolean match;
        conn = DataSource.getConnection();
        stmt = conn.prepareStatement("SELECT COUNT(*) world_name FROM world WHERE world_name = ?");
        stmt.setString(1, worldName);
        ResultSet rs = stmt.executeQuery();
        rs.next();
        match = rs.getInt(1) == 1;
        conn.close();
        return match;
}
```

Door gebruik te maken van SELECT COUNT(*) met de te zoeken parameter en deze Query 1 maal te doorlopen moet het resultaat 1 zijn indien niet zal de boolean match 0 zijn en weten we dat de parameter niet bestaat.

Switch met verschillende databanken

Als eerste moesten we een verbinding maken met de databank en een mogelijkheid voorzien dat de applicatie in twee talen ging werken dus ook een tweede databank voorzien. Een aparte DAL klasse voor de verbinding met een statische methode getConnection() leek ons het beste idee en een simpele if/else leek ons de makkelijkste oplossing.

Eerste moest de Engelstalige databank klaar zijn en hadden we niet verder nagedacht over een andere databank toe te voegen. Nu we echter nog een tweede moesten toevoegen en de andere Dal functies reeds geschreven waren waar we gebruik maakten van de statische getConnection() methode konden we daar niet telkens een extra parameter meegeven, dus daar een overal een boolean English/Arabic aan toevoegen leek ons geen goed idee.

Daarom zijn Lenny en ik op het idee gekomen deze te integreren via de login pagina. Om het zo ook t voor de gebruiker makkelijk te maken en niet op elke pagina te moeten kiezen op welke taal de databank moest staan hebben en konden we het ook makkelijk toevoegen aan het reeds gebruikte Sessie Object.

```
int language = Integer.parseInt(request.getParameter("language"));
String username = request.getParameter("username");
HttpSession session = request.getSession();
try {
    if (language == 1) {
        ServDataSource.switchDataSource(true);
    } else {
        ServDataSource.switchDataSource(false);
```

Nu moesten we enkel nog een Service aanmaken zodat het ook in de DAL klasse Datasource kon worden opgevangen.

```
public class ServDataSource {
    public static void switchDataSource(boolean language) throws SQLException{
        DataSource.setLanguage(language);

    }
}
```

Daar waren twee extra methodes voor nodig:

```
public static boolean isLanguage() {
    return language;
}
public static void setLanguage(boolean aLanguage) {
    language = aLanguage;
}
```

Om uiteindelijk te komen tot:

```
public static java.sql.Connection getConnection() throws SQLException {
    if (!isLanguage()) {
        return (DriverManager.getConnection("jdbc:mysql://localhost:3306/omandb?useUn
    } else {
        return(DriverManager.getConnection("jdbc:mysql://localhost:3306/omandbarabic?
    }
}
```

Achteraf gezien is dit een eenvoudige oplossing maar het heeft ons toch wat tijd gekost om er achter te komen.

Bootbox

Omdat ik erop aangedrongen had dat voor de delete functies zekere boodschappen moesten weer gegeven worden met dubbele confirmaties heb ik de een andere plugin van Bootstap geïmplementeerd namelijk Bootbox. Deze creëert, managet en verwijderd dialoog boxen of JS event handelers zonder in te grijpen op de DOM elementen.

Om hier mee van start te gaan moesten we de een kleine javascript library importeren. Alle events worden gestuurd vanuit onze AJAX.js in de $(document).ready(function () dus ook dit moest van hieruit werken.

Door het click event op de knop met de id “delete-family-btn” word er een confirm box geactiveerd met een boodschap indien de gebruiker OK aangeeftwordt de callback result gelijk aan true. In onderstaand voorbeeld triggerd dit echter tweede confirm box indien deze keer result ook op true gezet word de AJAX functie gestart en wordt de family met de id die gelijk gesteld werd aan ($(this).attr("value")) door gegeven aan de controller DeleteFamily die een id verwacht om deze vervolgens door te geven aan de service ServFamily en daarna via de DaFamily verwijderd te worden uit de databank. Daarna worden alle relevante load functies weer geladen vanuit Loaders.js.

```
// functie voor delete family btn in dashboard.jsp
$(document).on('click', '.table #delete-family-btn', function () {
    var id = ($(this).attr("value"));
    bootbox.confirm("<center>This will delete all references to this family \n\
                    in breeds & linked organisms.<br><br> <b>Are you sure?</b></center>\n\
                     ", function (result) {
        if (result === true) {
            bootbox.confirm("<center><b>ARE YOU VERY SURE?</b><br><br>This \n\
                        will delete all references to this family in breeds\n\
                         & linked organisms.</center>  ", function (result) {
                if (result === true) {
                    $.ajax({
                        url: 'DeleteFamily?id=' + id,
                        type: 'POST',
                        dataType: 'text',
                        cache: false,
                        async: true
                    }).done(function () {
                        loadFamilies();
                        loadSubFamilies();
                    });
                }
            });
        }
    });
});
// functie voor delete habitat btn in dashboard.jsp
```

Omdat de standaard kleuren niet pasten in ons design moesten we enkel nog een kleine aanpassing doen in Bootbox.js en in de admin.css

```
if (!button.className) {
  if (total <= 2 && index === total-1) {
    // always add a primary to the main option in a two-button dialog
    button.className = "button-grey";
  } else {
    button.className = "button-grey";
  }
}
```

```
.button-grey {
        font-weight: bold;
        height: 2em;
        color: white;
        font-size: 1.1em;
        border: 0px solid #000000;
        border-radius: 0.3em 0.3em 0.3em 0.3em;
        background-color: #666666;
        height: 2.3em;
        min-width: 8em;
}

.button-grey:hover{
        color: white;
        background-color: #444444;
}
```

Modal

Nadat de admin pagina een tijdje in gebruik was hebben we de feedback gekregen dat de gebruikers het moeilijk vonden om een familie of een variëteit toe te voegen tijdens het ingeven van een organisme. Men moest het tabblad verlaten om één of twee anderen te open en daar deze eerst toe te voegen.

Om dit op te lossen hebben we gebruik gemaakt van dialog functie van Bootstrap namelijk Modal. Deze zat reeds in onze bootstrap.js dus er moest niet extra toegevoegd worden. Modal werkt op hetzelfde principe als BootBox maar is iets uitgebreider. Dus als de gebruiker in het create-organism form naast Breed op de Add knop drukt verscheen er een nieuw venster.

```
<button type="button" class="button-grey" data-toggle="modal" data-target="#add-family-breed">Add</button>
```

Als er op deze knop gedrukt wordt de classe aria-hidden = true op false gezet. De opmaak van een Modal werkt net hetzelfde als de rest van bootstrap met rows en colums, ik heb er wel de meeste van de standaard opmaak moeten uithalen want deze was te uitgebreid.

De velden zijn eveneens dezelfde als van de forms create-family en create-subfamily evenals de verwerking via AJAX. Hier moest de respectievelijke functies herhaald worden met de id add-family-breed.

```
// functie inserten van family. in modal dashboard.jsp
$('#create-family-modal').submit(function (e) {

    var $message = $('#create-family-message-modal');
    $message.show();
    $.ajax({
        url: 'InsertFamily',
        type: 'POST',
        dataType: 'text',
        data: $('#create-family-modal').serialize()
    }).done(function (data) {
        if (data === 'succes') {
            $message.append('<div class="alert alert-success" role="alert">\n\
            <a href="#" class="close" data-dismiss="alert">&times;</a>\n\
                Family successfully created.</div>');
            loadFamilies();
            $("#create-family-modal")[0].reset();
            setTimeout(function () {
                $('#create-family-modal').find('label').parent().attr('class', 'form-group');
            }, 2800);
        } else if (data === 'exists') {
            $message.append('<div class="alert alert-danger" role="alert">\n\
                <a href="#" class="close" data-dismiss="alert">&times;</a>\n\
                    Family name already exists!</div>');
        } else if (data === 'sql') {
            $message.append('<div class="alert alert-danger" role="alert">\n\
            <a href="#" class="close" data-dismiss="alert">&times;</a>\n\
            Service unavailable!</div>');
        } else if (data === 'required') {
            $message.append('<div class="alert alert-danger" role="alert">\n\
                <a href="#" class="close" data-dismiss="alert">&times;</a>\n\
            Fill in the required fields!</div>');
        }
        setTimeout(function () {
            $message.fadeOut('slow');
            $message.empty();
        }, 2800);
    });
    e.preventDefault();
});
```

De dropdown listen werden gevuld via de AJAX functies loadWorlds en LoadFamilies hier moest er alleen voor gezorgd worden dat deze de gelijke id kregen als in het originele form. Dit was dan ook het mooie om met AJAX te werken zodra er één werd geüpdatet volgde ook de andere.

```
// functie vult tabel in World tab in dashboard.jsp
function loadWorlds() {

    var $table = $('#worlds-table');
    var $ddl = $('#world-ddl-insert-organism, #world-ddl-insert-family, \n\
          #world-ddl-update-family, #world-ddl-pending, #world-ddl-published,\n\
          #world-ddl-queue');
    $.ajax({
        url: 'SelectAllWorlds',
        type: 'GET',
        dataType: 'json',
        cache: false,
        async: true
    }).done(function (data) {
        $table.html('');
        $ddl.html('');
        $ddl.append('<option value="" disabled selected>Select World</option>');
        $table.append('<tr>\n\
                       <th>Name</th>\n\
                       <th>Description</th>\n\
                       <th></th>\n\
                       </tr>');
        data.forEach(function (world) {
            $ddl.append('<option value="' + world.worldId + '">'
                    + world.worldName + '</option>');
            $table.append('<tr>\n\
                          <td>' + world.worldName + '</td>\n\
                          <td>' + world.description + '</td>\n\
                          <td>\n\
                          <button class="no-button" id="update-world-btn" \n\
                              type="submit" value="' + world.worldId + '">\n\
                               <span class="icon-pencil2"></span></button>\n\
                          <button class="no-button" id="delete-world-btn" \n\
                              type="submit" value="' + world.worldId + '">\n\
                                  <span class="icon-cross"></span></button>\n\
                          </td>\n\
                          </tr>');
        });
        adminCheck();
    });
};
```

Besluit

Het was een zeer fijne en leerrijke opdracht met heel veel werk. Ik had het geluk deel te zijn van een team dat goed kon samenwerken en ook de tijd nam me uitleg te geven bij de delen die ik niet begreep. . Ik was deel van alle onderdelen project met uitzondering van de foto's, regex en de geolocatie. Zo heb ik heel veel geleerd over Bootstrap, AJAX , JAVASCRIPT en JQUERY dingen die in de lessen niet of bijna niet aanbod zijn gekomen maar naar mij bescheiden mening een must zouden moeten zijn in de leerstof en na kort onderzoek ook op de arbeidsmarkt zeer gevraagd zijn

We zijn allemaal trots op het resultaat en hebben tevreden opdrachtgevers en een eindwerk dat niet misstaat op een Curriculum Vitae.