



Persoonlijk Dossier: *Biodiversity Oman*

Project Werk

Naam: Lenny Donnez

Schooljaar 2014-1015

Inleiding	4
Mijn aandeel	5
DaUserAccount Functies	6
Connectie Back -en Front End	8
Event Handling	10
Login Systeem	11
UX/UI	12
Admin Interface	12
Hoofd website	14
Optimalisaties	16
Problemen	17
Probleem 1: Dynamische tabel met actie knoppen	17
Probleem 2: Insert organisme	18
Probleem 3: Griekse karakters.	19

Inleiding

We hebben Java gebruikt voor back-end, waaronder we de JDBC Library hebben gebruikt om de connectie te verzorgen tussen de databank en onze Java code. Deze beslissing werd genomen aangezien de meeste van de groepsleden met deze Library het beste overweg konden en vooralsnog een deadline hadden.

Als databank hebben we MySQL gebruikt, die na enkele hevige maar productieve discussies snel op orde kwam.

Om onze data naar onze front-end te sturen hebben we een 'middenlaag' van JQuery AJAX calls gebruikt. Deze is een eenvoudige maar doeltreffende manier om data dynamisch op onze pagina te laden. Verder is er ook gebruik gemaakt van Twitter Bootstrap 3. Vooral van de handige grid hebben we gretig gebruik gemaakt om onze applicatie op een eenvoudige manier responsive te maken.

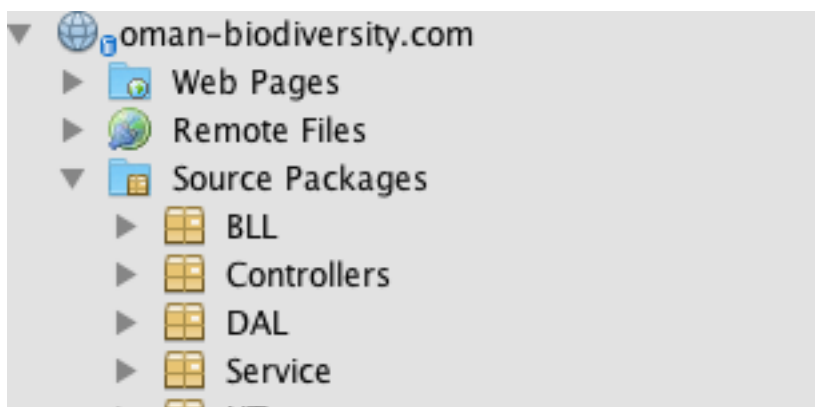
Mijn Aandeel

Na onze laatste vergadering voor het ontwerp van onze databank zijn we meteen aan onze back-end begonnen en hebben we ons klasse diagram in de praktijk omgezet naar onze BLL.

Hierbij heeft ieder van ons een verantwoordelijkheid gekregen. Ik zelf heb de back-end code geschreven voor onze gebruikers management interface.

Omdat we initieel dachten dat de organismen ingevoerd gingen worden door 'stagiairs' en professoren, hebben we gekozen voor twee soorten gebruikers namelijk de gewone gebruiker en de admin. Deze wordt tevens in de databank onderscheiden door een eenvoudige boolean.

Aangezien ik één van de eerste was dat ging werken aan de back-end heb ik het project initieel opgezet, en onze mappenstructuur aangebracht. Hiervoor heb ik gekozen om ons aan het MVC patroon te houden. Met dit als resultaat



Onze mappenstructuur is verdeeld onder

- BLL: Business Logic
- DAL: Data Acces met al onze functies die onze connectie met de DB verzorgen
- Service: Die als ons model dient en onze Data klaarmaakt om naar onze DAL te sturen.
- Controllers: dispatcher van Get en Post requests.

Verder heb ik de DAL klasse voor de user interface onder verdeeld in enkele functies. Deze kan je uiteraard vinden in het DAL package onder DaUserAccount.

DaUserAccount Functies

public static void insert(UserAccount user)

Deze functie neemt een User object en slaat deze vervolgens op in de databank

public static UserAccount selectByUsername(String username)

Deze functie neemt een gebruikersnaam en retourneert dan een User object terug met al zijn gegevens behalve het wachtwoord.

public static void update(UserAccount user)

Neemt een bestaand User object en overschrijft het met de bestaande gegevens in de databank.

public static void delete(String username)

Neemt een username en verwijderd al zijn gegevens.

updatePassword(String password, String username)

Update het bestaande wachtwoord met een nieuwe hashed wachtwoord(BCrypt)

public static boolean checkPassword(String username, String password)

Neemt het wachtwoord van een gebruikersnaam om na te gaan of het overeen komt met het hashed wachtwoord in de databank.(Bcrypt)

public static boolean checkUsername(String username)

Kijkt na of deze gebruikersnaam nog niet bestaat

public static void setSuperUser(String username)

zet de Admin Boolean in de databank op 1 om een gebruiker een admin te maken.

public static List<UserAccount> selectAll()

Selecteert alle Users uit de databank en steekt de gegevens van elke gebruiker in een Object waarna alle User objecten in een List geretourneerd worden

public static boolean isAdmin(String username)

Kijkt na of een gebruikersnaam Admin is of niet

public static void setNormalUser(String username)

Zet de admin boolean in de databank op 0 om een Admin terug een gewone gebruiker te maken.

public static List search(String keyword)

Neemt een keyword en zoekt in de UserAccount tabel, of dit keyword overeen komt met voornaam, achternaam, email of voornaam+achternaam. Omdat de voor -en achternaam allebei in een aparte tabel zitten heb ik CONCATENATE in onze query moeten gebruiken om deze 2 waarden samen te kunnen voegen.

public static boolean isLastUser()

Deze functie kijkt na of een een gebruiker de laatste gebruiker is in onze databank, om te vermijden dat deze nooit kan verwijderd worden (Deze functie werd geschreven door Bert).

De objecten dat meegestuurd worden in deze DAL functies worden in onze Service laag klaargemaakt. Deze functies kan je vinden in de Service package onder ServUserAccount. Elke parameter van deze functies wordt meegegeven door zijn aparte controller. Waarbij het Model(Services) Deze in objecten steekt en vervolgens doorstuurt naar onze DAL.

Connectie Back -en Front End

Om onze gebruikers de beste User Experience te geven hebben we AJAX gebruikt om onze data te presenteren in onze views. Hiervoor hebben we de ingebouwde AJAX functies van de JQuery Library gebruikt. Deze hebben we in aparte functies gestopt om zo onze code makkelijk uit te kunnen breiden zonder conflicten.

Omdat niemand van mijn groep ervaring had met AJAX calls te schrijven heb ik op voorhand enkele AJAX calls voor de user management gedocumenteerd.

Ik heb gekozen om al deze functies in twee aparte bestanden te plaatsen. Deze kan je vinden in de Web Pages map in de submap JS. Waarin drie bestanden staan genaamd loaders.js, ajax.js en view.js. De loaders bevat alle AJAX functies die de data in de DOM laden tijdens het laden van de DOM.

Voorbeeld:

```
8 function loadUsers() {
9
10     var $table = $('#users-table');
11     $.ajax({
12         url: 'SelectAllUserAccounts',
13         type: 'GET',
14         dataType: 'json',
15         cache: false,
16         async: true
17     }).done(function (data) {
18         $table.html('');
19         $table.append('<tr>\n\
20             <th>Username</th>\n\
21             <th>Firstname</th>\n\
22             <th>Lastname</th>\n\
23             <th>City</th>\n\
24             <th>Country</th>\n\
25             <th>Email</th>\n\
26             <th>Phone</th>\n\
27             <th>Admin</th>\n\
28             <th></th>\n\
29             </tr>');
30         data.forEach(function (user) {
31             $table.append('<tr>\n\
32                 <td>' + user.userName + '</td>\n\
33                 <td>' + user.firstName + '</td>\n\
34                 <td>' + user.lastName + '</td>\n\
35                 <td>' + user.city + '</td>\n\
36                 <td>' + user.country + '</td>\n\
37                 <td>' + user.email + '</td>\n\
38                 <td>' + user.phone + '</td>\n\
39                 <td>' + user.isAdmin + '</td>\n\
40                 <td>\n\
41                 <button class="no-button" id="delete-user-btn" type="submit" value="' + user.userName + '"><span clas
42                 <button class="no-button" id="make-admin-btn" type="submit" value="' + user.userName + '"><span class
43                 <button class="no-button" id="make-normal-btn" type="submit" value="' + user.userName + '"><span clas
44                 </td>\n\
45                 </tr>');
46         });
47     });
```


Deze functie roept de JQuery AJAX functie op met al zijn instellingen. Als de connectie lukt, wordt er een done(promise) terug gestuurd waarna de users tabel in de DOM wordt geplaatst. Alle andere data wordt op een nagenoeg zelfde manier geladen en zo heeft elke entiteit zijn aparte functie zoals loadOrganism, loadHabitats, loadSeasons enz.

Standaard heeft de Servlet API geen JSON encoder daarvoor heb ik een aparte Library moeten toevoegen. genaamd Gson(google). Hiermee kon ik na het importen van de Library met een eenvoudige lijn code onze Lists en Objecten naar JSON converteren.

Voorbeeld:

```
List users = ServUserAccount.selectAll();  
response.getWriter().write(new Gson().toJson(users));
```

Event Handling

Onze Event handling heb ik in één apart bestand gestoken. Omdat JavaScript geen events kan opvangen tijdens het laden van de DOM, staan al deze functies in de `$(document).ready` scope. Alle onze interactie met buttons forms enz. wordt opgevangen met een event.

Voorbeeld:

```
// create user form usermanagement.jsp
$('#create-user-form').submit(function (e) {

    var $message = $('#create-user-message');
    $message.show();
    $.ajax({
        url: 'InsertUserAccount',
        type: 'POST',
        dataType: 'text',
        data: $('#create-user-form').serialize()
    }).done(function (data) {
        if (data === 'succes') {
            $message.append('<div class="alert alert-success" role="alert"><a href="#" class="close" data-dismiss="alert">&
            loadUsers();
            $('#create-user-form')[0].reset();
            setTimeout(function () {
                $('#create-user-form').find('label[class=\`col-sm-2 control-label\`]').parent().attr('class', 'form-group')
            }, 2800);
        } else if (data === 'exists') {
            $message.append('<div class="alert alert-danger" role="alert"><a href="#" class="close" data-dismiss="alert">&t
        } else if (data === 'sql') {
            $message.append('<div class="alert alert-danger" role="alert"><a href="#" class="close" data-dismiss="alert">&t
        } else if (data === 'required') {
            $message.append('<div class="alert alert-danger" role="alert"><a href="#" class="close" data-dismiss="alert">&t
        }
        setTimeout(function () {
            $message.fadeOut('slow');
            $message.empty();
        }, 2800);
    });
    e.preventDefault();
});
```

Hier wordt het submit event opgevangen van de create-user form. De form data wordt in de AJAX Call geserialized om vervolgens een message te ontvangen van de daarbij horende controller. Ons Model zorgt voor de messages zo zijn er 3 mogelijkheden namelijk exists - sql of required.

- exists wil zeggen dat iets al bestaat in de databank
- sql wil zeggen dat er iets mis is met de databank (connectie enz)
- required wil zeggen dat er een verplicht niet veld niet is ingevuld.

Elke message wordt enkele seconde weergegeven in een slide onder de forms.

Login Systeem

De eerste taak om onze admin interface te ontwikkelen, was het opzetten van een veilig login systeem. Hiervoor heb ik een sessie object aangemaakt die alle attributen behalve het wachtwoord bevat. Dit object wordt in de adminheader.jsp telkens nagekeken als er van pagina wordt gewisseld.

Voorbeeld:

```
<%  
    String username = null;  
    UserAccount us = (UserAccount) session.getAttribute("user");  
    if (us != null) {  
        username = us.getUserName();  
    } else {  
        String message = "You are not logged in";  
        session.setAttribute("error", message);  
        response.sendRedirect("admin.jsp");  
        return;  
    }  
%>
```

Omdat er zoals eerder al vermeld twee soorten gebruikers zijn. Vond ik het belangrijk dat er visueel ook een verschil was na het inloggen. Zo wordt er nagekeken of een gebruiker admin is of niet.

Voorbeeld:

```
<script type="text/javascript">  
    var adminuser = document.getElementById('adminuser');  
    var adminpublish = document.getElementById('adminpublish');  
    <%if (us.getIsAdmin() == true) {%>  
    adminuser.innerHTML += '<a href="usermanagement.jsp"><span class="icon-users"></span><strong>User Management</strong>  
    adminpublish.innerHTML += '<a href="publish.jsp"><span class="icon-book"></span><strong>Publish</strong></a>';  
    <%} else {%>  
    adminuser.innerHTML += '';  
    adminpublish.innerHTML += '';  
    <%}%>  
</script>
```

Hier wordt het User object (us) nagekeken of de gebruiker admin is of niet. Waarna het de menu items *user management* en *publish* in de header plaatst wanneer de gebruiker een Admin is. Of deze menu items weg laat wanneer een gewone gebruiker inlogt.

Admin Interface

Nadat mijn back-end code op orde was ben ik begonnen met onze login pagina en admin interface te ontwikkelen. Hiervoor heb ik een klein beetje bootstrap gebruikt en enkele JQuery plugins. Ik heb gekozen voor een tab interface zodat we onze entiteiten makkelijk konden onder verdelen, en om een simpele User Experience te presenteren voor onze gebruikers.

Voorbeelden:

[Dashboard](#)

Version 0.2.2 - For any problems click here to mail

[Publish](#)

[User Management](#)

admin

[Log out](#)

Organisms

Worlds

Families


Breed


Seasons


Habitats










Geolocations

Pending for evaluation

Press the  for details

Press the  to insert an organism



Common name	Scientific name	Inserted on	Updated on	
Arabian Tahr	Arabitragus jayakari	Mar 15, 2015	undefined	
Spiny Lobster	Panulirus Homarus	Mar 31, 2015	undefined	
Octopus	Octopus Vulgaris	Mar 31, 2015	undefined	
Sea Cucumber	Holothuroidea	Mar 31, 2015	undefined	
Thresher Shark	Alopias	Mar 31, 2015	undefined	
Jellyfish	Cnidaria Scyphozoa Aurelia	Apr 2, 2015	undefined	
Crown-of-thorns Starfish	Acanthaster Planci	Apr 2, 2015	undefined	
Red Sea Star	Fromia Milleporella	Apr 2, 2015	undefined	
San Dollar	Echinarachnius Parma	Apr 2, 2015	undefined	

[Dashboard](#)

Version 0.2.2 - For any problems click here to mail

[Publish](#)


[User Management](#)


admin


[Log out](#)










Users

Create User

Press the  button to delete a user (Caution! no confirmation screen)

Press the  button to promote a user to admin

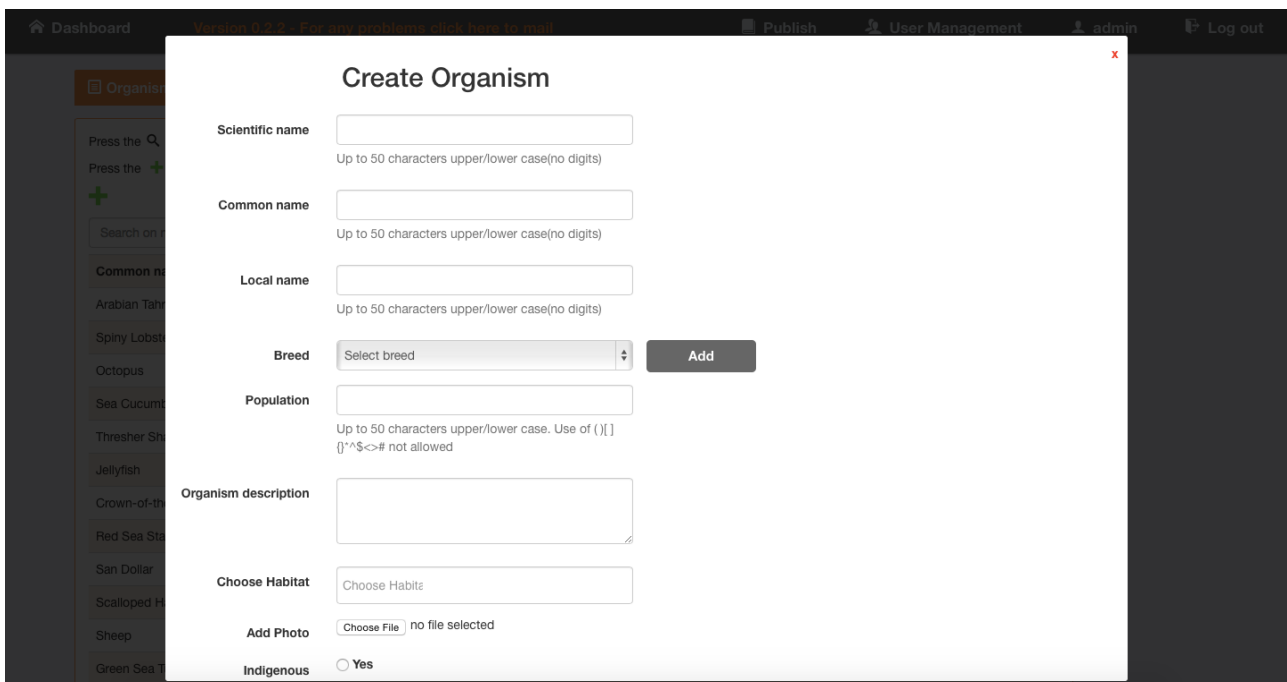
Press the  button to demote a user to normal

Username	Firstname	Lastname	City	Country	Email	Phone	Admin	
admin	admin	admin	unknown	unknownp	admin@admin...	0123456789	true	  
developer	Developer	Developer	Antwerp	Belgium	developerstea...	0123458	true	  
TomAdriaens	Tom	Adriaens	HALEN	Belgium	tomadriaens@...	0032472967402	true	  

Elke tab buiten de create user tab heeft een dynamische tabel waarin de attributen van elke entiteit wordt weergegeven. Deze tabellen hebben ook actie knoppen als laatste kolom en bij de organisme tab en gebruikers tab bevindt zich ook een zoekbalk.

Toevoegen organisme.

Elke entiteit(tab) heeft een + knop. Wanneer er op geklikt wordt komt er een Modal tevoorschijn waarmee de gebruiker alle velden kan invullen om een record toe te voegen.



Form Validatie

Elke validatie wordt op onze front end na elke interactie met het input field gevalideerd. Hiervoor heb ik een bootstrap compatibele library toegevoegd aan ons project genaamd validator.js. Als een gebruiker een input veld fout invult wordt deze rood en komt er een kleine foutmelding tevoorschijn.

Voorbeeld:

Scientific name

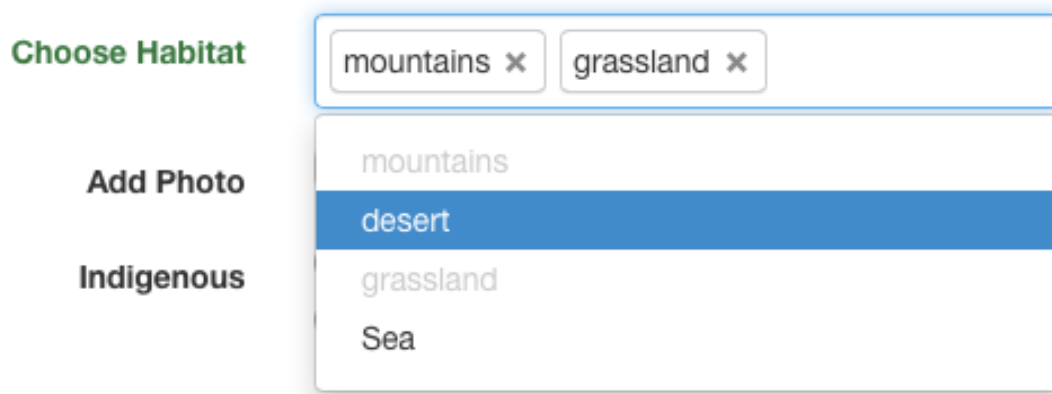
pattern mismatch

Ik zelf heb de validatie verzorgd van de user create, de rest van mijn groepsgenoten hebben de validatie gedaan van de andere forms. Ook hebben zij voor onze server side validatie gezorgd wat je kan bekijken in onze Service Package.

Many to Many in de Front-end

We hebben enkele many-to-many relaties in onze databank bijvoorbeeld, seasons en habitats. Hiervoor moesten we een systeem bedenken, om het onze gebruikers zo makkelijk mogelijk te maken. Zo kwam Bert met een idee dat hij van een andere website had gezien. Hierbij werd Data in 'blokjes' in een input field gestoken. Dit vonden we een schitterend idee, en zo ben ik op zoek gegaan om deze manier na te bootsen. Hierbij ben ik een library tegengekomen genaamd chosen.js. Alle Data wordt in een lijst geladen, waarin de gebruiker meerdere opties kan kiezen om toe te voegen.

Voorbeeld:



The image shows a web form with a section titled 'Choose Habitat'. Below this title is a dropdown menu that is currently open. The dropdown menu displays a list of habitat options: 'mountains', 'desert', 'grassland', and 'Sea'. The 'desert' option is highlighted with a blue background. Above the dropdown menu, there are two buttons labeled 'mountains x' and 'grassland x', indicating that these two options have been selected. To the left of the dropdown menu, there are two labels: 'Add Photo' and 'Indigenous'.

Hoofd website

Na de vierde sprint werd het tijd om na te denken hoe we de website gingen ontwikkelen. Hiervoor ben ik met het idee gekomen om een Single Page Website te maken. Dit geeft als voordeel dat de website makkelijk responsive te maken is en dat we de roadshow in Oman ten volste kunnen promoten. Na de sprint vergadering was iedereen overtuigd met het idee.

Kort nadien ben ik begonnen met het designen van de website. Deze is in de volgende secties ingedeeld: Event, Calendar, Worlds, Fun stuff en About us.

Event

Deze sectie bevat info over de roadshow en wat info over DNA en Taxonomy. Hierbij heb ik ook enkele afbeeldingen gebruikt van het kinderboek. Om de website in dezelfde stijl te houden, want dit werd gevraagd van de opdrachtgever(Philippe).

Calendar

Deze sectie bevat een grote kalender gelinkt aan een google calendar die alle datums bevat van waar de roadshow zich bevind. Hiervoor heb ik FullCalendar.js library gebruikt die het heel makkelijk maakte om de google API aan te koppelen.

Verder is er ook een kleine Social sectie voorzien. Deze bevat een Instagram feed die door Bert is ontwikkeld en een Facebook feed die ik heb geïmplementeerd.

Worlds

Deze sectie bevat de info over de verschillende werelden met daarin alle hun organisme. Hiervoor heb ik een slider gebruikt die open gaat op het klikken van een wereld.

Fun stuff

Deze sectie bevat tekeningen en kleine opdrachtjes voor kinderen, dit is gedesignen door Tom.

About us

Info over ons en Philippe.

Wanneer er op een menu item geklikt wordt, wordt er naar de sectie 'geslide'. Dit wordt gedaan door simpele javascript/Jquery code.

Nadat ongeveer alle content en secties op de pagina stonden heb ik mij bezig gehouden met nieuwe type faces toe te voegen aan de pagina om onze content nog meer aantrekkelijker en leesbaarder te maken. De website gebruikt 3 verschillende type faces namelijk: Montserrat voor de menu items, Museo voor alle text content, en Helvetica-Neue voor headers buiten de content (Social Media).

Optimalisaties

De website gaat vooral op mobiele toestellen bezocht worden, aangezien het mobiele internet niet snel is in Oman heb ik enkele optimalisaties ingevoerd door middel van de Google pagespeed test te gebruiken. Deze tool geeft een score op basis van de snelheid van je website en je gebruik van bandbreedte. Zo heb ik al onze JS en CSS code geminified om bandbreedte te besparen en heb ik compressie op onze Glassfish server ingeschakeld. Daarnaast heb ik ook een Servlet aan onze code toegevoegd om client side caching in te schakelen(CacheFilter), dit gaf een enorme performance boost.

Om '.jsp' uit onze URL's te halen zijn ze herschreven met urlrewritefilter library, deze kan je vinden onder "WEB-INF/lib/". De rewriting zelf staat in een apart XML bestand genaamd urlrewrite.xml die je kan vinden in de WEB-INF map.

Problemen

Doorheen mijn Biodiversity verhaal ben ik samen met mijn groepsgenoten op enkele problemen aangelopen. Hierbij heb ik steeds hulp gehad van mijn teamgenoten en heb ikzelf uiteraard ook mijn steentje bijgedragen om hun problemen op te lossen.

Probleem 1: Dynamische tabel met actie knoppen

Toen ik mijn eerste dynamische tabel (user management) aan het ontwikkelen was kwam ik op het probleem dat de HTML5 specificatie geen forms toelaat in en tabel. Zo kon ik voor onze actie buttons geen forms gebruiken met hidden values en buttons. Dit heb ik opgelost door de buttons dynamisch te laten laden met als value het id van de einiteit.

voorbeeld:

```
data.forEach(function (user) {
    $table.append('<tr>\n\
        <td>' + user.userName + '</td>\n\
        <td>' + user.firstName + '</td>\n\
        <td>' + user.lastName + '</td>\n\
        <td>' + user.city + '</td>\n\
        <td>' + user.country + '</td>\n\
        <td>' + user.email + '</td>\n\
        <td>' + user.phone + '</td>\n\
        <td>' + user.isAdmin + '</td>\n\
        <td>\n\
        <button class="no-button" id="delete-user-btn" type="submit" value="' + user.userName + '"><span class="icon-
        <button class="no-button" id="make-admin-btn" type="submit" value="' + user.userName + '"><span class="icon-p
        <button class="no-button" id="make-normal-btn" type="submit" value="' + user.userName + '"><span class="icon-
        </td>\n\
    </tr>');
});
```

Hier wordt dus de user.userName als value doorgegeven met de button

in

op

```
$(document).on('click', '.table #delete-user-btn', function () {
    var username = $(this).attr("value");
    bootbox.confirm("<center>Delete this user?", function (result) {
        if (result === true) {
            $.ajax({
                url: 'DeleteUserAccount?username=' + username,
                type: 'POST',
                dataType: 'text',
                cache: false,
                async: true
            }).done(function () {
                loadUsers();
            });
        }
    });
});
```

(username is de PK de user accounts tabel). Op deze manier kon ik de value uit de button halen wanneer er geklikt wordt op deze manier:

de lijn `$(this).attr("value")`); haalt de value uit de button wanneer er op gelikt wordt om zo door te sturen met AJAX naar de server.

Probleem 2: Insert organisme

Bij het ontwikkelen van het inserten van een organisme kregen we steeds nul pointer exception fouten. Na enkele debuggen sessies met Bert, Eric en Oualid kwam het er op neer dat het probleem aan onze many-to-many relaties lag(habitat, season...enz). Deze worden namelijk in de controller in een lege int array gestoken waarna deze naar de databank wordt gestuurd. Om dit op te lossen moesten we elke request van deze many-to-many values nakijken of die niet leeg is.

Voorbeeld:

```
int[] habitatIds = null;
int[] seasonIds = null;
int[] eatenByOrganismIds = null;
int[] eatingOrganismIds = null;
int[] geolocationIds = null;
int subfamilyId = 0;

if (request.getParameterValues("organism-habitat-id") != null) {
    habitatIds = new int[request.getParameterValues("organism-habitat-id").length];
    for (int i = 0; i < request.getParameterValues("organism-habitat-id").length; i++) {
        habitatIds[i] = Integer.parseInt(request.getParameterValues("organism-habitat-id")[i]);
    }
}
```

De `request.getParameter` neemt alle values, deze moest worden gecontroleerd anders kregen we een `NullPointerException`.

Nadat we dit hadden geïmplementeerd kwamen we nog steeds op `NullPointerException`s. Na een nieuwe debugging sessie was het al meteen duidelijk dat we hetzelfde principe moesten implementeren in onze Service en onze DAL.

Voorbeeld Service oplossing:

```
if (habitatid != null) {
    for (int i = 0; i < habitatid.length; i++) {
        habitat.add(new BLL.Habitat(Array.getInt(habitatid, i)));
    }
}
```

Voorbeeld DAL oplossing:

```
// habitat
if(!organism.getHabitat().isEmpty()) {
    stmt = conn.prepareStatement("INSERT INTO habitat_organism (habitat_id, organism_id) VALUES(?,?)");
    for (Habitat h : organism.getHabitat()) {
        stmt.setInt(1, h.getHabitatId());
        stmt.setInt(2, orgId);
        stmt.executeUpdate();
    }
}
```

Probleem 3: Griekse karakters.

Toen Philippe begonnen was met organisme toe te voegen hadden we er niet bij stil gestaan dat hij Griekse benamingen ging gebruiken voor bepaalde organisme. Daarentegen dachten we dat MySQL, die echter standaard op UTF-8 stond ingesteld geen problemen ging hebben met Griekse/Arabische karakters. Blijkbaar was dit niet het geval en moesten we dus een oplossing zoeken. Hiervoor heb ik eerst de databank en al zijn tabellen op UTF8-mb4 moeten zetten. Dit zorgt er wel echter voor dat alle karakters op 4-bits wordt opgeslagen ipv 3bits. Hierdoor moesten de VARCHAR velden op een maximum van 191 karakters staan ipv 255.

Nadat de databank juist was ingesteld moesten we ook onze Java code aanpassen. Een standaard String in Java is UTF-16 dit moest aangepast worden voor onze velden waar Griekse karakters moesten komen. Dit heb ik opgelost door deze data in een nieuw string object te stoppen en te converteren naar UTF-8.

Voorbeeld:

```
new String(request.getParameter("organism-common-name").getBytes("iso-8859-1"), "UTF-8"),
new String(request.getParameter("organism-local-name").getBytes("iso-8859-1"), "UTF-8"),
new String(request.getParameter("organism-description").getBytes("iso-8859-1"), "UTF-8"),
```