

Timebox2- Edit other users

Oversigt

OpgaveNavn	Implementering af redigering af andre bruger		
Implementering af krav	Delvis implementering af WEB-2 og WEB-4		
Udført af	Jan	Dato	29-09-2021
Timebox	2	Område	Website

Contents

INTRODUKTION.....	1
ANALYSE.....	2
DESIGN.....	2
IMPLEMENTERING	4
VERIFIKATION	7
TESTRESULTAT.....	8
KONKLUSION	10
REFERENCER	ERROR! BOOKMARK NOT DEFINED.

Introduktion

Der vil blive gået igennem analyse og implementering, af de hjemmesidesider der gør det muligt for en "office" user at redigere eksisterende brugere.

Analyse

For en "office user" kan lave ændringer til andre brugere, vil der skulle være to hjemmesidesider:

1. Filter der gør det nemmere at finde en bruger.
2. En redigeringside for den ønskede bruger.

Ved punkt 1, vil der kunne blive lavet et HTML felt, hvor backenden til disse danner queries til databasen med en filter funktion, der ligger standard i Django/databaser, der så begrænser et nyt HTML felt med bruger valgmuligheder, der sender en valgt mulighed med en "POST request".

Ved punkt 2, vil der kunne blive lavet HTML felter, hvor backenden til disse danner queries ind til databasen begrænset med en "GET request", som henter "POST requesten" tilhørende punkt 1.

Ved punkt 2 skal det være muligt at:

- Redigere database felter for eksisterende brugere:
 - navn
 - email
 - telefon nummer
 - firma
- Hvis den redigerede bruger er "field user", skal han kunne opgraderes til "office user".
- Hvis den redigerede bruger er "field user", skal brugeren kunne slettes

Design

Designet vil være opdelt i 2 (backend og frontend).

Backend:

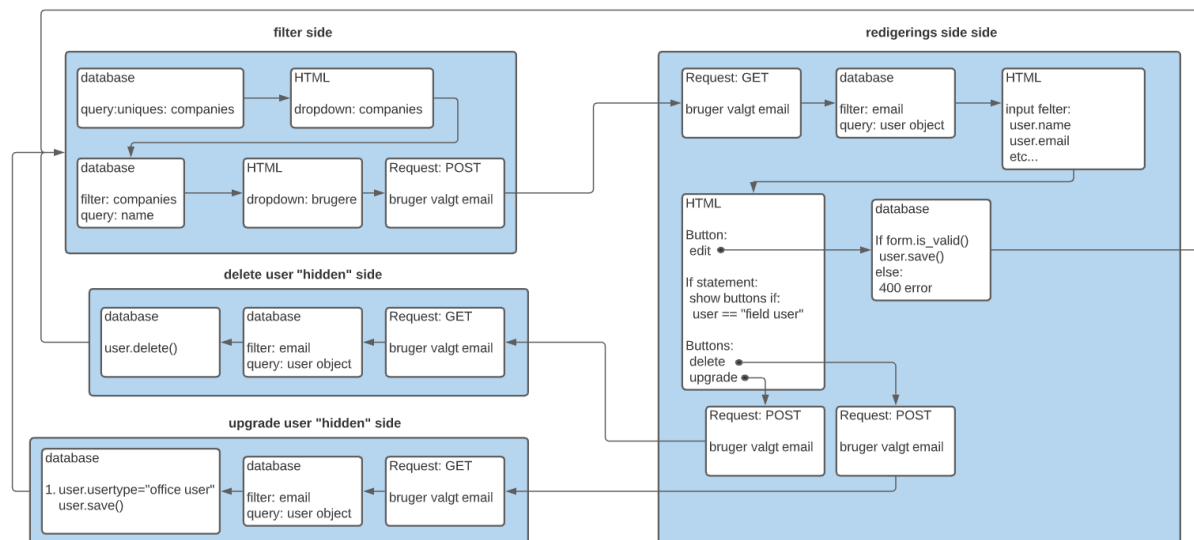
Den første side vil stå for:

1. At filtrere brugere, og gøre det nemmere at finde en specifik bruger, hvilket bliver gjort ved at vælge et firma ud fra en liste hentet gennem database queries med en filter funktion der søger efter firmanavne i alle user database entries, og returner derefter en liste med disse firmanavne, uden duplikeringer, ved at bruge filer argumentet "unique".
2. Der bliver igen søgt i databasen med et filter, baseret på et valgt firmanavn, som vil returnere en liste af brugere tilhørende et firma.
3. Når en bruger er valgt, vil der blive lavet en "POST request" der poster email adressen på en bruger (e-mailadressen er et unik databasefelt, og sat som en brugers "username"), for derefter at lave et redirect til en bruger redigerings side.

Den anden side vil stå for:

1. Lave en "GET request", for at få bruger email'en fra den foregående sides "POST request".
 2. Bruger emailen vil blive brugt til at lave en database query, efter et specifikt brugerobjekt.
 3. Objektet vil blive brugt til input fields i HTML'en, med default værdier fra objektet.
 4. HTML edit knappen vil passere input felter værdierne ind i en form, baseret på User databasemodellen, og overskrive ændrede værdier, hvis værdierne overholder formens krav, og ellers give en error 400 meddelelse, der specificerer hvilket felt/felter der er problematiske.
- I tilfælde af at bruger objektets userType felt er "field user", vil et if statement få HTML'en til at vise 3 ekstra knapper:
 - Delete user
 - Upgrade user
 - "Delete user" knappen vil genere en "POST request" med brugerens email adresse, for derefter at dirigere ind på en "skjult" side (der er ingen html for denne side), som får email adressen med "GET request", laver en database query efter brugeren med pågældende email adresse, for så og slette den pågældende database entry.
 - "Upgrade user" knappen vil genere en "POST request" med brugerens email adresse, for derefter at dirigere ind på en "skjult" side (der er ingen HTML for denne side), som får email adressen med GET request, laver en database query efter brugeren med pågældende email adresse, får så at ændre brugerens userType felt til "Office user".

De før beskrevne "flows" kan ses i Figur 1.



Figur 1 backend flow

Frontend:

Frontenden vil overordnet set være som beskrevet i "TB1-website setup and landing page.docx", i hensyn til edit users.

Implementering

Dette afsnit vil forklare den grundlæggende kode til edit users siderne.

Til hver funktion i backenden vil der blive brugt dekoratørerne vist på Figur 2:

- `@login_required` = tjekker om hjemmesidebrugeren er logget ind.
- `@user_controller` = tjekker om brugeren kan se hjemmesidesiden

```
72 @login_required
73 @user_controller
```

Figur 2 backend - decorators

Funktionen på Figur 3, viser flowet som beskrevet i design afsnittet for backend, omkring filter siden:

- Linje 85: få et objekt med alle bruger tilknyttet firmaer nævnt i databasen, uden duplikeringer.
- Linje 88: Hent alle brugere, i tilfælde intet firma er valgt.
- Linje 91-103: vis et firma er valgt hent brugere for det specifikke firma, og render hjemmesiden forfra.
- Linje 105-114: vis intet firma er valgt, render hjemmesiden med alle brugere i databasen.

```
74 def office_edit_user(request):
75     """Page for choosing a user, that shall be edited..."""
81     # Dummy parameter
82     comp = 'comp'
83
84     # Query all unique companies in the database
85     company = Users.objects.order_by().values_list('company', flat=True).distinct()
86
87     # No company chosen yet, so query all users in the database
88     users = Users.objects.order_by()
89
90     # Company is chosen:
91     if request.method == 'POST':
92         data = request.POST['company_list']
93         if data != "None":
94             # Query database for users, filtered by chosen company
95             users = Users.objects.order_by().filter(company=data)
96             user_header_text = "Users for: " + str(data)
97             # Render page with filtered users, instead of all users
98             render_dict = {...}
103            return render(request, "office_edit_user.html", render_dict)
104
105     # Text for a header
106     user_header_text = "All users:"
107
108     # Default render with all companies, and all users
109     render_dict = {...}
114    return render(request, "office_edit_user.html", render_dict)
```

Figur 3 backend - filter page

For frontend HTML'en kan der ses på Figur 4:

- Linje 29 – 34: laver en dropdown liste, hvor valgmulighederne blive lavet med en for løkke, ud fra firmaobjektet.
- Linje 36: sender en " POST request", så bruger mulighederne er begrænset til et enkelt firma.

- Linje 41-50: viser det samme som de 2 foregående punkter, dog med forskellen at det er bruger objektet i stedet for firmaobjektet der bliver itereret, og aktionen for form'en sender til siden der håndterer redigeringer.

```

25 <!-- BOX-DROPDOWN: company -->
26 <h1>Choose a company</h1>
27 <form action={% url 'office_edit_user' %} method="post">
28   {% csrf_token %}
29   <select name="company_list">
30     <option value="None" selected hidden>Choose company</option>
31     <option value="None">All companies</option>
32     {% for val in company %}
33       <option value={{ val }}> {{ val }} </option>
34     {% endfor %}
35   </select>
36   <input type="submit" value="Filter">
37 </form>
38 <br>
39
40 <!-- BOX-DROPDOWN: user -->
41 <h1>{{ header_text }}</h1>
42 <form action={% url 'office_edit_user_final' %}>
43   {% csrf_token %}
44   <select name="users_list">
45     {% for val in users %}
46       <option value="None" selected hidden>Choose user</option>
47       <option value={{ val.email }}> {{ val.name }} </option>
48     {% endfor %}
49   </select>
50   <input type="submit" value="Edit user">
51 </form>

```

Figur 4 frontend - filter page

Funktionen på Figur 5, viser flowet som beskrevet i design afsnittet for backend, omkring redigerings siden:

- Linje 126: laver "GET request" for at få brugeren, valgt på foregående side's email.
- Linje 139-130: vis ingen bruger er valgt, diriger til foregående side.
- Linje 133: hent bruger entry med en specifik e-mailadresse.
- Linje 136-143: vis alle field entries i HTML'en er valide, så overskriv bruger entry, ellers diriger til error 400 page med specifik fejlmeddelelse.

```

118 def office_edit_user_final(request):
119     """Page for editing a chosen user...."""
120     # Get user chosen from "office_edit_user"
121     data = request.GET['users_list']
122
123     # If there's no user chosen, redirect back to "office_edit_user"
124     if data == 'None':
125         return redirect("office_edit_user")
126
127     # Make query based on chosen user from "office_edit_user"
128     user_choice = Users.objects.get(email=data)
129
130     # Run if statement, if "edit button" is clicked on website
131     if request.method == 'POST':
132         form = AllUsersFields(request.POST, instance=user_choice)
133         # Save changes to model entry, and redirect to "office_edit_user"
134         if form.is_valid():
135             form.save()
136             return redirect("office_edit_user")
137         else:
138             return HttpResponseRedirect("Couldn't save user input, because of following error/errors: " + str(form.errors))
139
140     return render(request, "office_edit_user_final.html", {'user': user_choice})

```

Figur 5 backend - edit page

På Figur 6, kan der ses tabellen vist for brugeren på hjemmesiden, hvor linje 28-29 viser hvordan et "field" i bruger objektet bliver vist, samt giver et felt til at skrive ændringer i.

```
24 <form action="" method = "POST">
25   {% csrf_token %}
26   <table>
27     <tr>
28       <td><label for="name">Name:</label></td>
29       <td><input type="text" id="name" name="name" value="{{ user.name }}"></td>
30     </tr>
31     <tr...>
32     <tr...>
33     <tr...>
34     <tr...>
35     <tr...>
36     <tr...>
37     <tr...>
38     <tr...>
39     <tr...>
40     <tr...>
41     <tr...>
42     <tr...>
43     <tr...>
44     <tr...>
45     <tr...>
46     <tr...>
47     <tr...>
48     <tr...>
49     <tr...>
50     <tr...>
51     <tr...>
52     <tr...>
53     <tr...>
54   </table>
55   <br>
56   <br>
57 </form>
```

Figur 6 frontend - edit page - user info

På Figur 7 kan de to knapper til delete og upgrade user ses, samt hvordan der på linje 58 og 70, bliver bestemt om knapperne er relevant for en pågældende bruger.

```
58 {% if user.userType == "Field user" %}
59   <form action={% url 'upgrade_field_user' %}>
60     {% csrf_token %}
61     <button type="submit" name="upgrade_user" value="{{ user.email }}"
62       style="...">
63       Upgrade user</button>
64   </form>
65 <br>
66 <br>
67 </form>
68 {% endif %}
69
70 {% if user.userType == "Field user" %}
71   <form action={% url 'delete_field_user' %}>
72     {% csrf_token %}
73     <button type="submit" name="delete_user" value="{{ user.email }}"
74       style="...">
75       Delete user</button>
76   </form>
```

Figur 7 frontend - edit page - buttons

På Figur 8 vises der hvordan en bruger bliver opgraderet:

- Linje 158: "GET request" for at få bruger email på bruger der bliver redigeret.
- Linje 161: hent bruger objekt for bruger med en pågældende email.
- Linje 164-165: sæt bruger type til "Office user", og overskriv bruger entry.
- Linje 168: diriger hen til bruger filter side.

```

149 def upgrade_field_user(request):
150     """This function is for upgrading a field user to office user..."""
151
152     # Get user email, from "office_edit_user_final"
153     data = request.GET['upgrade_user']
154
155     # Get corresponding user entry
156     user_choice = Users.objects.get(email=data)
157
158     # Change usertype and overwrite entry
159     user_choice.userType = 'Office user'
160     user_choice.save()
161
162     # return to "office_edit_user"
163     return redirect('office_edit_user')

```

Figur 8 backend - upgrade user

På Figur 9 vises der hvordan en bruger bliver slettet:

- Linje 181: "GET request" for at få bruger email på bruger der bliver redigeret.
- Linje 184: hent bruger objekt for bruger med en pågældende email.
- Linje 187: slet bruger entry i databasen.
- Linje 190: diriger hen til bruger filter side.

```

172 def delete_field_user(request):
173     """This function is for deleting a field user..."""
174
175     # Get user email, from "office_edit_user_final"
176     data = request.GET['delete_user']
177
178     # Get corresponding user entry
179     user = Users.objects.get(email=data)
180
181     # Delete user entry
182     user.delete()
183
184     # return to "office_edit_user"
185     return redirect('office_edit_user')

```

Figur 9 backend - delete user

Verifikation

Elementer der skal testes:

- Knapper dirigerer de rigtige steder hen.
- Input fields kan overskrive entries.
- Opgrader bruger og slet bruger vises kun ved "field users".
- Opgrader bruger laver en "field user" til "office user".
- Slet bruger sletter bruger i databasen.
- Ved forkert input i input fields, bliver der givet fejl.

Alle test's antager:

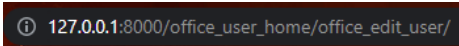
- Server er oppe og kører.
- Bruger er logget ind som "office user".
- Der er dirigeret hen til edit users siden.
- Der er lavet en "test" bruger, som er "field user", og tilhører firmaet "test firma".
- Der er lavet en admin bruger, med navnet admin.

Tabel 1: Tests til verifikation af opgave

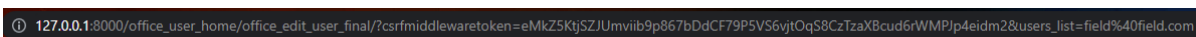
Test	Test Steps	Pre-requisites	Pass-betingelser	Resultat
T1: Knap dirigering	<ol style="list-style-type: none"> vælg et firma og tryk "Filter". vælg bruger og tryk "edit user". tryk "accept edits". vælg bruger og tryk "edit user". tryk "back". tryk "back". 	N/A	URL flowet vil være: <ol style="list-style-type: none"> /office_user_home/office_edit_user/ /office_user_home/office_edit_user_final/ /office_user_home/office_edit_user/ /office_user_home/office_edit_user_final/ /office_user_home/office_edit_user/ /office_user_home/ 	Bestået
T2: Input fields	<ol style="list-style-type: none"> vælg test firma og tryk "Filter". vælg test bruger og tryk "edit user". ændre "name" til "test2". tryk på "accept edits". 	For at tjekke resultat skal der være logget ind som admin, og kigges i users tabellen, på admin siden.	<ol style="list-style-type: none"> navnet for test@test.com er ændret til test2. 	Bestået
T3: Bruger type tjek for knapper	<ol style="list-style-type: none"> vælg test bruger og tryk "edit user". tryk "back". vælg "admin" og tryk "edit user". 	N/A	<ol style="list-style-type: none"> for test bruger kan "upgrade user" og "delete user" ses. for admin bruger kan "upgrade user" og "delete user" ikke ses. 	Bestået
T4: Opgrader field user	<ol style="list-style-type: none"> vælg test bruger og tryk "edit user". tryk på "upgrade user" 	For at tjekke database resultat skal der være logget ind som admin, og kigges i users tabellen, på admin siden.	<ol style="list-style-type: none"> der bliver dirigeret tilbage til user filter siden. i databasen er user type for test@test.com er ændret fra "field user" til "office user". 	Bestået
T5: Slet field user	<ol style="list-style-type: none"> vælg test bruger og tryk "edit user". tryk på "delete user" 	For at tjekke database resultat skal der være logget ind som admin, og kigges i users tabellen, på admin siden.	<ol style="list-style-type: none"> der bliver dirigeret tilbage til user filter siden. i databasen er test@test.com er ikke længere i tabellen. 	Bestået
T6: Input fejl	<ol style="list-style-type: none"> vælg test bruger og tryk "edit user". ændre email til: test@test 		<ol style="list-style-type: none"> der bliver dirigeret ind på en 400 side, med fejlen stående på siden. 	Bestået

Testresultat

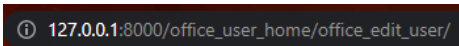
Test T1:

1. 

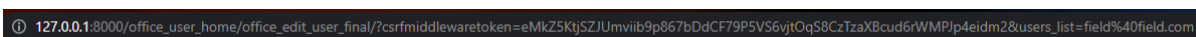
Figur 10 T1.1

2. 

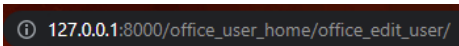
Figur 11 T1.2

3. 

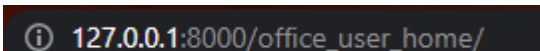
Figur 12 T1.3

4. 

Figur 13 T1.4

5. 

Figur 14 T1.5

6. 

Figur 15 T1.6

Test T2:

test@test.dk

User info	
Email adress:	<input type="text" value="test@test.dk"/>
Name:	<input type="text" value="test2"/>
PhoneNumber:	<input type="text" value="123456789"/>
Company:	<input type="text" value="test"/>
UserType:	<input type="text" value="Field user"/>

Figur 16 T2

Test T3:

Usertype "admin":

EUROWIND ENERGY A/S	
Name:	<input type="text" value="admin"/>
Company:	<input type="text" value="ewe2"/>
Phone number:	<input type="text" value="123456789"/>
Email:	<input type="text" value="admin@admin.com"/>
Password:	<input type="password" value="**** *"/>
User type:	Admin
<input type="button" value="Accept edits"/>	
<input type="button" value="back"/>	

Figur 17 T3 admin

UserType "field user":

EUROWIND ENERGY A/S	
Name:	<input type="text" value="test"/>
Company:	<input type="text" value="test"/>
Phone number:	<input type="text" value="123456789"/>
Email:	<input type="text" value="test@test.dk"/>
Password:	<input type="password" value="**** *"/>
User type:	Field user
<input type="button" value="Accept edits"/>	
<input type="button" value="Upgrade user"/>	
<input type="button" value="Delete user"/>	
<input type="button" value="back"/>	

Figur 18 T3 field user

Test T4:

test@test.dk

User info	
Email address:	<input type="text" value="test@test.dk"/>
Name:	<input type="text" value="test"/>
PhoneNumber:	<input type="text" value="123456789"/>
Company:	<input type="text" value="test"/>
UserType:	<input type="text" value="Office user"/>

Figur 19 T4

Test T5:

<input type="checkbox"/>	NAME	COMPANY	EMAIL ADDRESS	USERTYPE
<input type="checkbox"/>	admin	ewe2	admin@admin.com	Admin
<input type="checkbox"/>	field	ewe1	field@field.com	Field user
<input type="checkbox"/>	office	ewe3	office@office.com	Office user

Figur 20 T5

Test T6:

Couldn't save user input, because of following error/errors:

- email
 - Enter a valid email address.

Figur 21 T6

Konklusion

Alt i edit users funktionaliteterne virker som forventet.