Timebox 4- CRON

Oversigt

OpgaveNavn	Implementering af CRON: create user code, clean up		
Implementering af krav	Hel implementering af kravene WEB-11: *"Det skal være muligt at vedligeholde database gennem hjemmesiden."* WEB-15: *"Det skal være muligt at beskytte mod "uautoriseret" bruger oprettelse."*		
Udført af	Marc	**Dato**	15-10-2021
Timebox	4	**Område**	Website

Contents

Introduktion

Det er nu muligt at implementere CRON på hjemmesiden, hvor den er deployed på AWS-serveren. Der skal implementeres fra Time Box 2 "create user code", så den kører hver 24. time. Der skal udvikles og implementeres et "clean up" af adgangen til faciliteterne hver 24. time.

Analyse og Design

For at bruge ”scheduleren” anvendes biblioteket ”django-crontab”[1], som skal tilføjes i filen settings.py i mappen website, se Figur 1.

```
35  INSTALLED_APPS = [
36      'website',
37      'create_user',
38      'field_user',
39      'office_user',
40      'rest_api',
41      'rest_framework',
42      'rest_framework.authtoken',
43      'djoser',
44      'django_crontab',
45      'django.contrib.admin',
46      'django.contrib.auth',
47      'django.contrib.contenttypes',
48      'django.contrib.sessions',
49      'django.contrib.messages',
50      'django.contrib.staticfiles',
```

Figur 1 Settings.py: Installed apps

Fra Time Box 2 blev filen cron.py med funktionen ”gen_user_code” udviklet, derudover skal der udvikles en ny funktion som vil blive kaldet ”clean_access”, begge funktion skal eksekveres, klokken 00:01 hverdag, den indstilling sættes i settings.py filen, som på Figur 2.

```
53  CRONJOBS = [
54      ('1 00 * * *', 'website.cron.gen_user_code'),
55      ('1 00 * * *', 'website.cron.clean_access')
56  ]
```

Figur 2 Instillingerne for hvornår CRON skal kører

For at sikre at tidspunktet er korrekt skal tidszonen indstillet, som på Figur 3, dette forgår i også settings.py.

```
144  TIME_ZONE = 'Europe/Copenhagen'
```

Figur 3 Tidszone

Implementering

Funktion ”clean_access”, se Figur 4, henter den nuværende dato og trækker en dag fra, derefter filtreres JoinTable efter den dato, hvor alle med den dato slettes.

```
25  def clean_access():
26      """
27      Cleans facility access
28      Removes all JoinTable that have expired
29      :return:
30      """
31      today = date.today() - timedelta(days=1)
32      yesterday = today.strftime("%Y-%m-%d")
33      JoinTable.objects.all().filter(timer=yesterday).delete()
34      pass
```

Figur 4 Funktionen "clean_access"

Verifikation

Test er som i Tabel 1
Alle tests forudsætter at:

- Tre bruger er lavet (admin, field og office) og at man har adgang til dem.
- 3 test faciliter er i databasen
- Serveren kører

Tabel 1: Tests til verifikation af opgave

Test	Test Steps	Pre-requisites	Pass-betingelser	Resultat
Gen_user_code	1. login på brugeren office og vent i 3 minutter	1. Man har lavet en ændring i settings.py så cron kører hvert andet minut.	- Create user code har ændret sig	Bestået
Clean_access	1. login på brugeren office 2. lave tre facilitets adgange: I. brugeren field skal have adgang til test fac1 til 14-10-2021 II. brugeren field skal have adgang til test fac2 til 15-10-2021 III. brugeren office skal have adgang til test fac3 til 14-10-2021 3. log ud og login på brugeren admin og vent 3 min	1. Man har lavet en ændring i settings.py så cron kører hvert andet minut	- Man kan se de tre JoinTable - De to JoinTable med test fac1 og test fac3 forsvinder	Bestået

Konklusion

De to funktioner er blevet implementeret og kører nu med CRON, så de aktiveres hver 24. time.

Referencer

[1]	django-crontab https://pypi.org/project/django-crontab/ (sidste besøgt 15-10-2021)