

# Aplicatie de manageriere a unui spital

## 1. Descrierea aplicatiei

Aplicatia web dezvoltata ofera toata serviciile esentiale de care are nevoie un spital, afisarea unei pagini web de intampinare atunci cand un utilizator acceseaza site-ul web, o pagina de *Contact* cu toate informatiile de care un pacient are nevoie pentru a lua legatura cu spitalul, o pagina *Despre* in care este descris spitalul, optiunile de *inregistrare*, *autentificare* si *delogare*. In aditie acestor pagini, aplicatia mai ofera si posibilitatea a trei tipuri de utilizatori, *Pacient*, *Dictor* si *Administrator*. Fiecare utilizator are atributii diferite, utilizatorul de tip *Pacient* poate vedea specializarile spitalului si doctorul atribuit fiecarei specializari, dar si sa creeze noi programari in functie de disponibilitatea doctorului respective. Utilizatorul de tip *Doctor* poate accepta, refuza sau sterge programari si isi poate modifica programul. Utilizatorul de tip *Administrator* are rolul de a manageria aplicatia web si poate adauga sau sterge pacienti si doctori.

## 2. Dezvoltarea aplicatiei

### 2.1. Setarea mediului de lucru

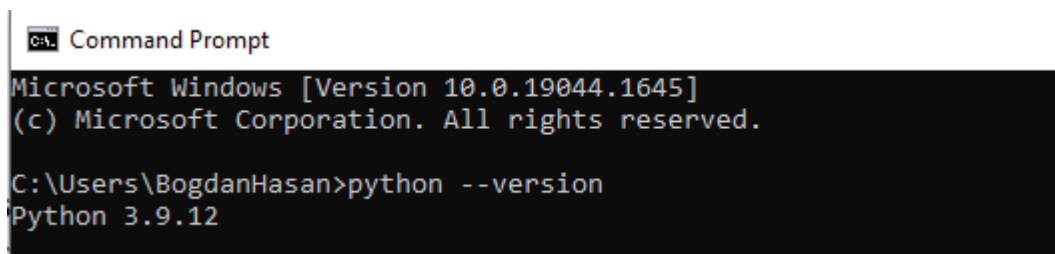
Inainte de a ma apuca de dezvoltarea efectiva a aplicatiei trebuiei sa imi setez un mediu de lucru si sa descarcam si instalam tehnologiile si aplicatiile folosite. Voi incepe cu descarcarea tehnologiilor si aplicatiilor.

Desi pentru a dezvolta aceasta aplicatie web putem folosi orice editor de text, este utila folosirea unui editor de text mai avansat care sa poata fi configurat astfel incat sa se potriveasca cat mai mult preferintelor noastre.

Pentru a descarca si instala editorul de text *Sublime Text Editor* se va accesa versiunea dupa sistemul de operare, in cazul acestei aplicatii *Windows*, accesand acest [link](#) catre pagina oficiala a editorului.

Inainte de a descarca si instala platforma de lucru Django trebuie sa ma asigur mai intai ca am limbajul de programare Python instalat. Pentru aceasta pot rula comanda “python –version” intr-o fereasta de command window sau intr-un terminal.

Daca Python este instalat, comanda de mai sus ar trebui sa ne afiseze ce versiune de Python avem instalata ca in figura de mai jos.



```
Command Prompt
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

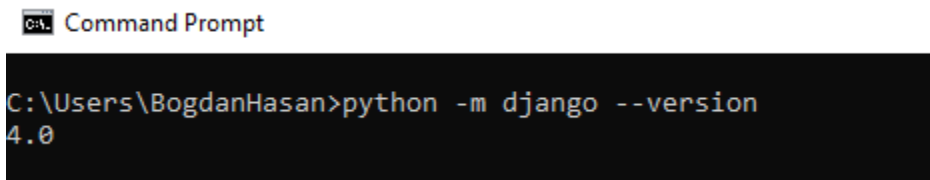
C:\Users\BogdanHasan>python --version
Python 3.9.12
```

figura 1 Verificare instalare Python

Daca mesajul este unul de eroare atunci Python se poate instala de pe site-ul oficial al acestuia accesand acest [link](#).

Un ultim pas mai este necesar inainte de a putea incepe dezvoltarea propriu-zisa a aplicatiei, si anume descarcarea si instalarea platformei Django. Aceasta se face prin utilizarea administratorului de pachete al Python, si anume pip. Se ruleaza intr-o fereastra de *Command Window* sau *terminal* comanda: *pip install django*.

Pentru a verifica ca instalarea a functionat, putem executa comanda: *python -m Django --version* care afiseaza versiunea de Django instalata.

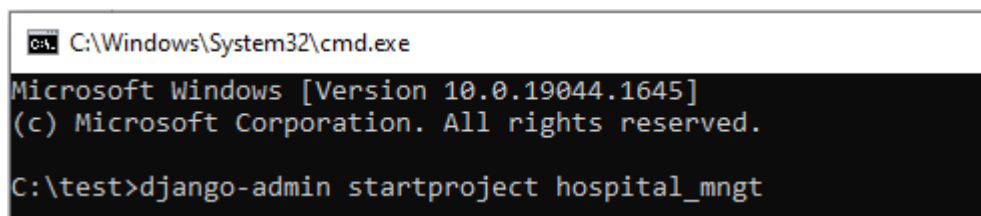


```
C:\Users\BogdanHasan>python -m django --version
4.0
```

figura 2 Verificare instalare versiune Django

## 2.2. Crearea unui proiect Django nou

Pentru a putea incepe sa dezvolt aplicatia, trebuie sa imi creez un proiect Django nou. Acest proiect ne va permite sa creem aplicatii in interioriul lui. Pentru a crea un proiect nou, intr-un director gol deschid o fereastra de comanda si execut comanda: *django-admin startproject hospital\_mngt*. Aceasta va crea un proiect nou cu numele "hospital\_mngt".



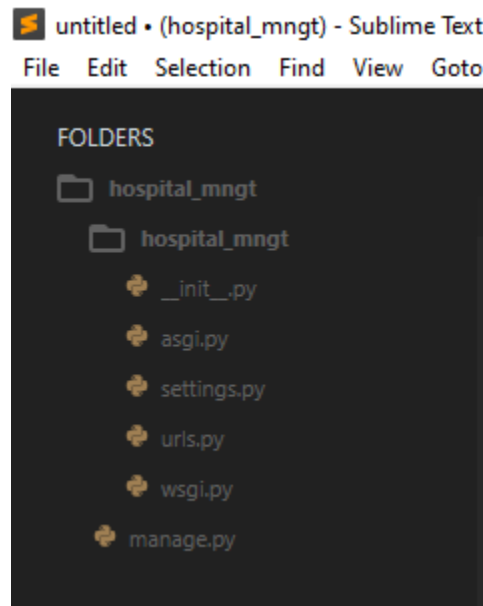
```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1645]
(c) Microsoft Corporation. All rights reserved.

C:\test>django-admin startproject hospital_mngt
```

figura 3 Crearea unui nou Proiect Django

Se poate observa ca in directorul in care am proiectul s-a creat un director nou cu numele proiectului meu. Prin *drag & drop* se aduce proiectul nou in Sublime.

In bara din stanga a editorului de text putem acum naviga prin structura proiectului ca in figura de mai jos.



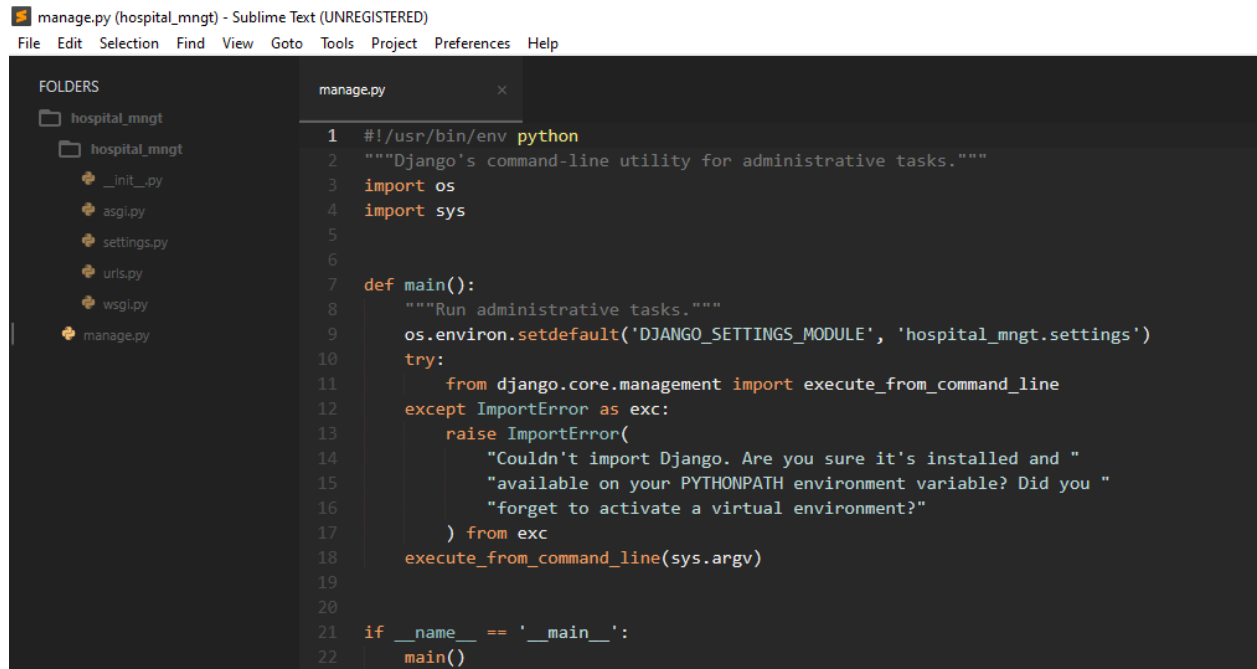
*figura 4 Listare structura proiect Django*

Django foloseste o structura de directoare pentru a aranja diferite parti ale aplicatiei web. Pentru aceasta, Django creeaza un director nou pentru fiecare proiect sau aplicatie. Atunci cand creem un proiect nou in Django, Django insusi creeaza un director radacina cu numele proiectului dat de noi si care contine fisierele necesare de baza ale functionarii unei aplicatii web.

Structura proiectului Django este explicata in subcapitolele care urmeaza.

### 2.2.3. Fisierul manage.py

Fisierul *manage.py* contine codul necesar pornirii serverului dar si al unor comenzi pe care le vom folosi pentru a porni serverul, depana, lansa, crea migrari, salvati migrari dar si multe altele.



```
manage.py (hospital_mngt) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS
hospital_mngt
  hospital_mngt
    __init__.py
    asgi.py
    settings.py
    uris.py
    wsgi.py
    manage.py

manage.py
1  #!/usr/bin/env python
2  """Django's command-line utility for administrative tasks."""
3  import os
4  import sys
5
6
7  def main():
8      """Run administrative tasks."""
9      os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'hospital_mngt.settings')
10     try:
11         from django.core.management import execute_from_command_line
12     except ImportError as exc:
13         raise ImportError(
14             "Couldn't import Django. Are you sure it's installed and "
15             "available on your PYTHONPATH environment variable? Did you "
16             "forget to activate a virtual environment?"
17         ) from exc
18     execute_from_command_line(sys.argv)
19
20
21 if __name__ == '__main__':
22     main()
```

figura 5 Fisierul manage.py

### 2.2.4. Fisierul \_\_init\_\_.py\_\_

Acesta este un fisier gol. Utilitatea acestui fisier este aceea ca interpretorul de Python va stii prin prezenta acestui fisier ca in interiorul aceluui director este un pachet si asadar prezenta lui face ca directorul in care se afla sa fie tratat ca un proiect.

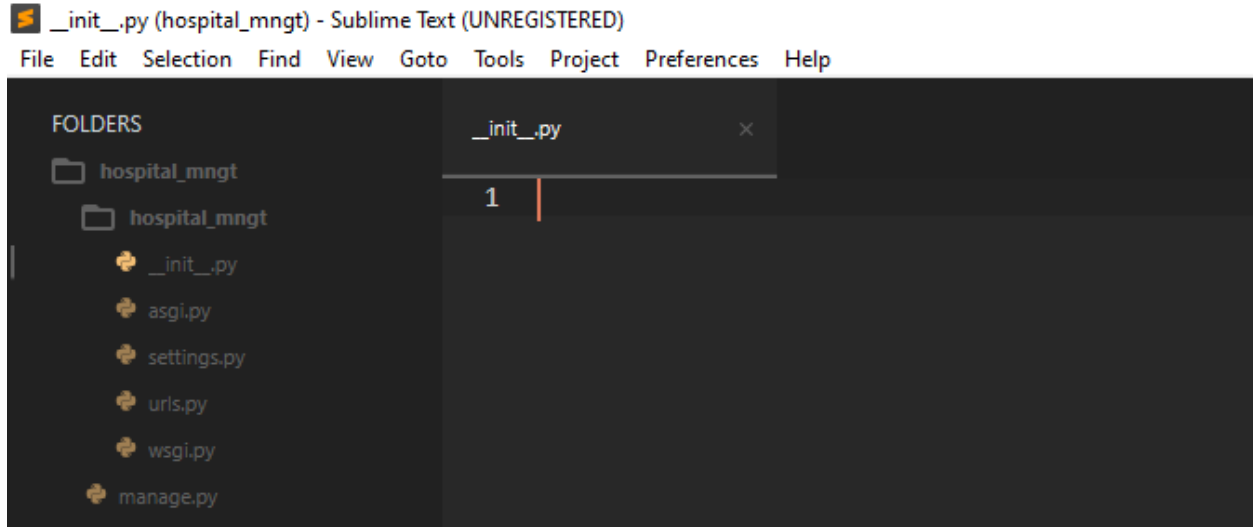
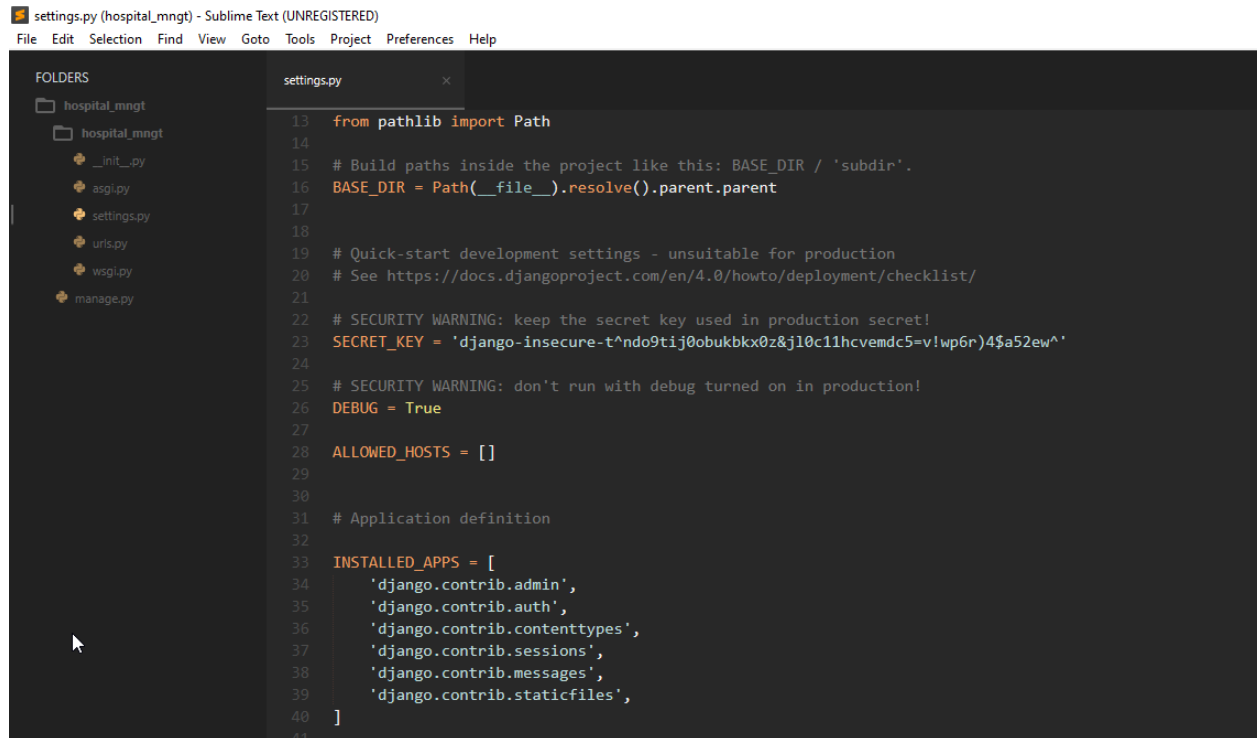


figura 6 Fisierul \_\_init\_\_.py\_\_

### 2.2.5. Fisierul settings.py

Acesta este cel mai important fisier, deoarece contine toate configuratiile proiectului si este folosit ca sa lege aplicatiile de proiect.

Acesta contine cateva nume de variabile, *sqlite3* ca baza de date in mod implicit, dar care poate fi schimbata in *Mysql*, *PostgreSQL* sau *MongoDB* in functie de aplicatia web pe care dorim sa o dezvoltam.

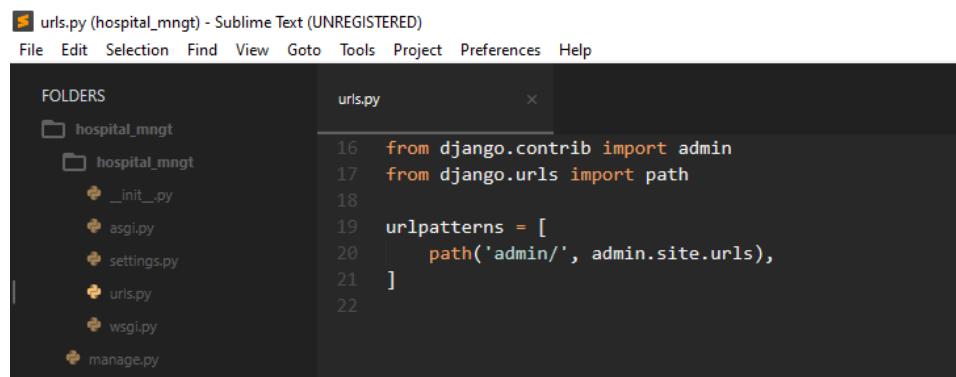


```
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17
18
19 # Quick-start development settings - unsuitable for production
20 # See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/
21
22 # SECURITY WARNING: keep the secret key used in production secret!
23 SECRET_KEY = 'django-insecure-t^ndo9tij0obukbkx0z&jl0c11hcvemdc5=v!wp6r)4$a52ew^'
24
25 # SECURITY WARNING: don't run with debug turned on in production!
26 DEBUG = True
27
28 ALLOWED_HOSTS = []
29
30
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40 ]
41
```

figura 7 Fisierul settings.py

## 2.2.6. Fisierul urls.py

Acesta contine toate destinatiile folosite in aplicatia noastra web. In cuvinte mai simple, acest fisier este folosit pentru a redirectiona utilizatorii care acceseaza un anumit link catre pagina respectiva.



```
16 from django.contrib import admin
17 from django.urls import path
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21 ]
22
```

figura 8 Fisierul urls.py

### 2.2.7. Fisierul wsgi.py

WSGI vine de la Web Server Gateway Interface, si descrie modul in care serverul interactioneaza cu aplicatia. Vom folosi acest fisier in momentul in care terminam de dezvoltat aplicatia si urmeaza sa o hostam. Este un pas simplu, trebuie doar sa importam componentele de mijloc in concordanta cu serverul pe care il vom folosi. Pentru fiecare server, exista o componenta Django de mijloc care rezolva toate problemele de integrare si conectivitate pentru noi.

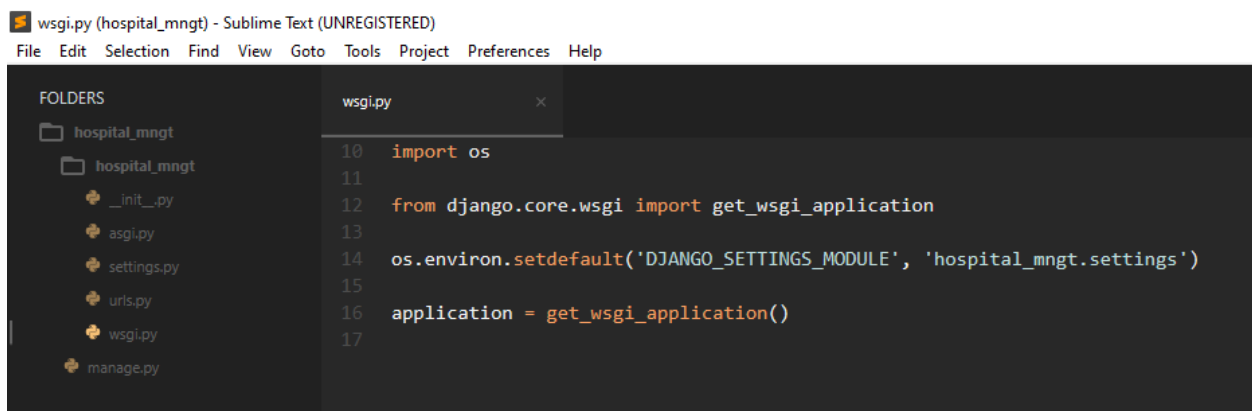


figura 9 Fisierul wsgi.py

### 2.2.8. Fisierul asgi.py

Fisierul *asgi.py* functioneaza in mod similar cu fisierul *wsgi.py* descris in capitolul anterior dar care vine cu cateva functionalitati in plus. *ASGI* vine de la *Asynchronous Server Gateway Interface* si inlocuieste mai nou fisierul *wsgi.py*.



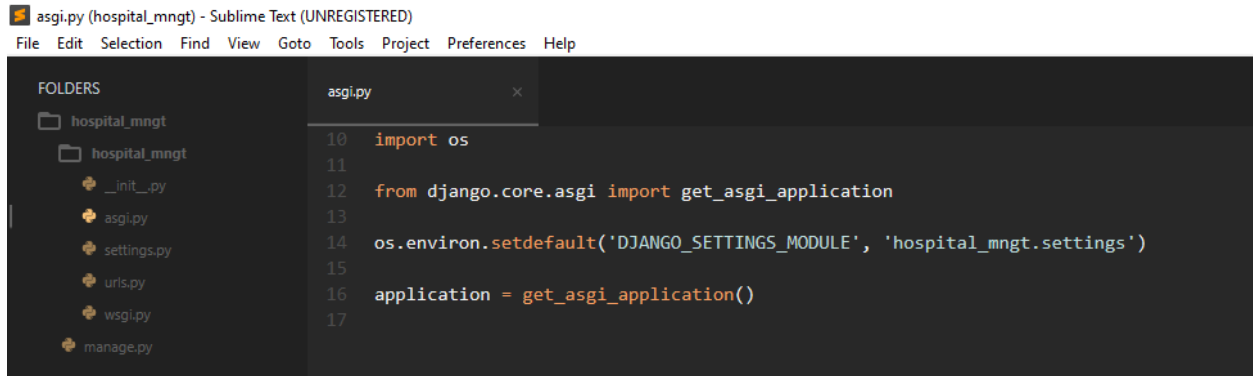


figura 10 Fisierul asgi.py

## 2.3. Crearea unei aplicatii Django noi

Django foloseste conceptul de proiecte si aplicatii pentru a administra codurile scrise si pentru a le prezenta intr-un mod cat mai curat si organizat. Un proiect Django contine una sau mai multe aplicatii, care executa o munca simultan pentru a asigura un mediu de lucru cat mai in.

Pentru a crea o aplicatie Django, in interiorul directorului unde am creat proiectul, intr-un *command window* sau *terminal* se va executa urmatoarea comanda *python manage.py startapp hospital* care va crea o aplicatie noua cu numele *hospital* ca in figura de mai jos.

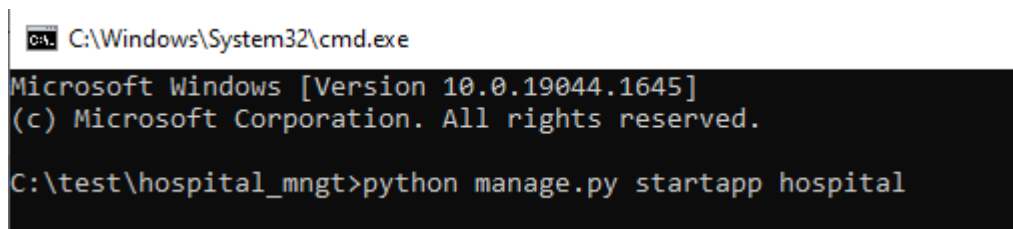


figura 11 Crearea unei aplicatii Django noi

In urma executarii comenzii de mai sus, in directorul proiectului s-a creat un director nou cu numele "*hospital*". De asemenea se mai poate observa ca in momentul creerii aplicatiei, Django a creat in mod automat cateva fisiere Python

pentru noi. Structura unei aplicatii Django si fisierele create vor fi descrise in subcapitolele urmatoare.

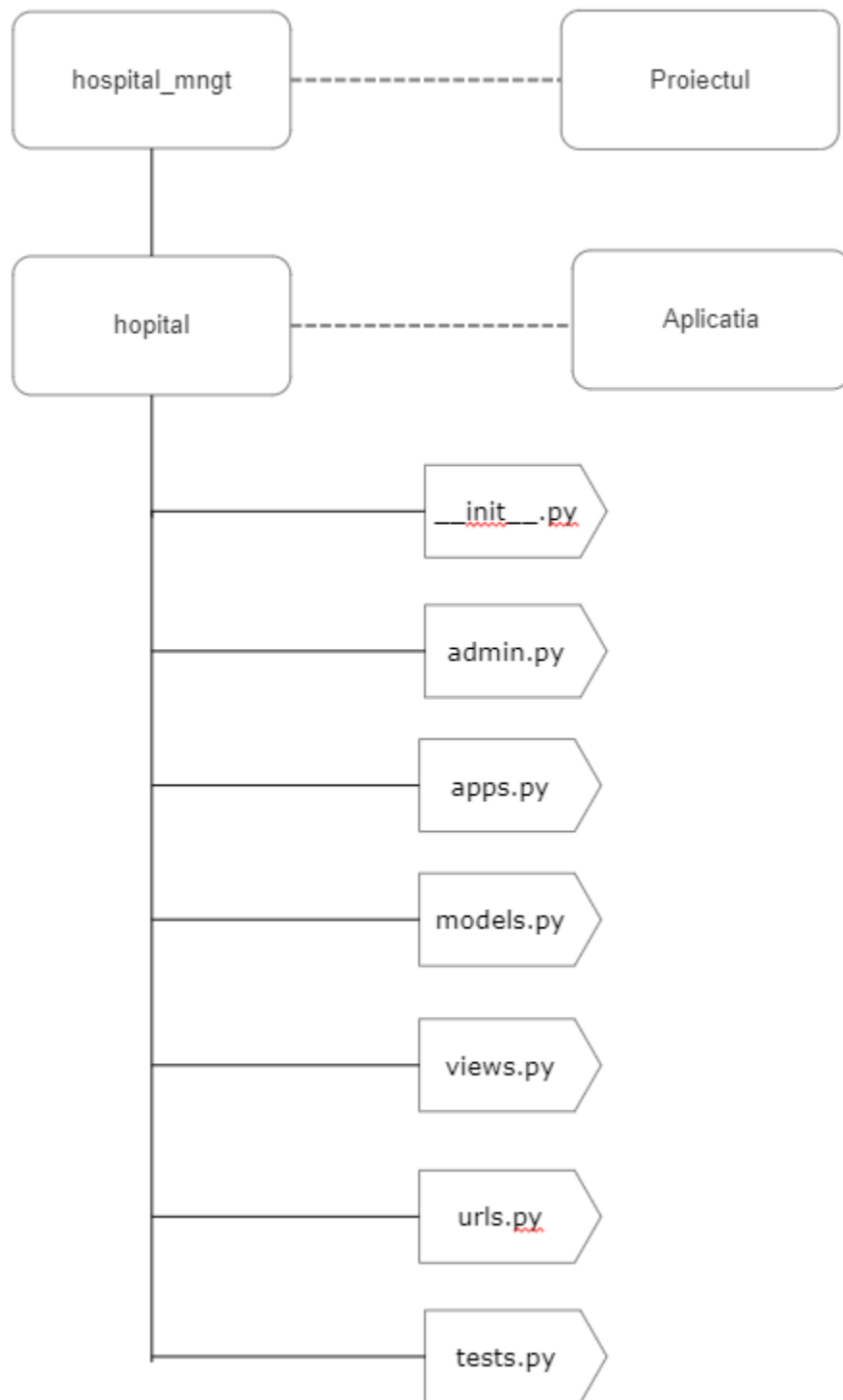


figura 12 Structura unei aplicatii Django

### 2.3.1 Fisierul hospital.\_\_init\_\_.py

Ca si in cazul fisierului “\_\_init\_\_.py” descris anterior in capitolul 2.2.4 Fisierul \_\_init\_\_.py, si acesta este un fisier gol. Singurul rol al acestui fisier este acela de a-l spune interpretului de Python ca acel director trebuie tratat ca un proiect.

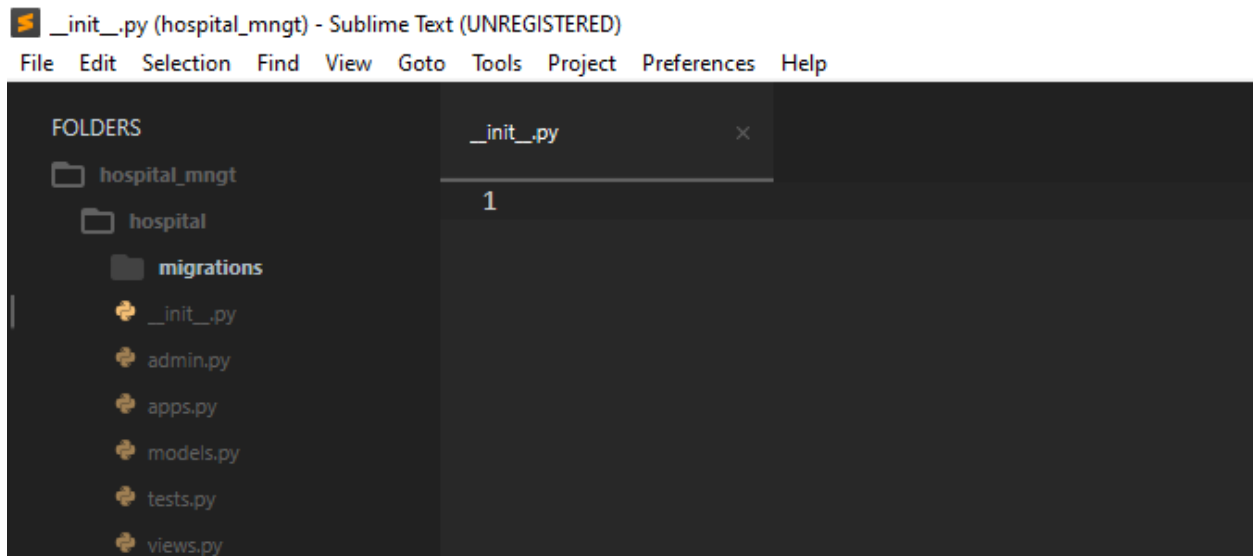


figura 13 Fisierul hospital.\_\_init\_\_.py

### 2.3.2. Fisierul admin.py

Fisierul *admin.py* este folosit pentru inregistrarea modelelor Django in administrarea aplicatiei. Este folosit pentru afisarea modelelor in panoul de administrare. Acest fisier are trei intrebuintari majore si anume:

- Inregistrarea modelelor;
- Crearea de superusers;
- Autentificarea si utilizarea aplicatiei;

### 2.3.3. Fisierul *apps.py*

*Apps.py* este un fisier folosit pentru a ajuta utilizatorul sa includa configuratiile aplicatiei. Cu toate acestea, includerea configurarilor de catre utilizator nu este un obicei foarte comun deoarece in marea majoritate a cazurilor configuratiile standard sunt suficiente.

### 2.3.4. Fisierul *models.py*

Fisierul *models.py* reprezinta modelele de aplicatii web sub forma de clase. Este considerat a fi cel mai important aspect al structurii fisierelor aplicatiei. Modelele defines structura bazei de date si descrie design-ul real, relatiile dintre seturile de date si constrangerile atributelor acestora.

### 2.3.5. Fisierul *views.py*

Un alt fisier foarte important atunci cand vorbim despre structura aplicatiei Django este fisierul *views.py*. Acest fisier ofera o interfata prin care un utilizator interactioneaza cu o aplicatie web Django. Ca si in cazul fisierului *models.py*, *views.py* contine toate view-urile sub forma de clase.

### 2.3.6. Fisierul urls.py

*Urls.py* functioneaza in mod asemanator cu fisierul *urls.py* discutat anterior in capitolul [2.2.6. Fisierul urls.py](#) avand ca scop principal de a lega solicitarile *URL* ale utilizatorului catre paginile corespunzatoare pe care acesta le indica.

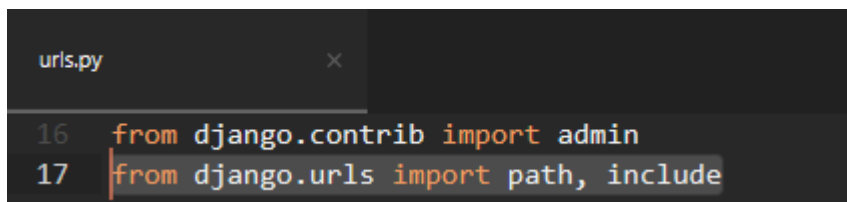
### 2.3.7. Fisierul test.py

Acest fisier permite utilizatorului sa scrie cod de testare pentru aplicatiile web. Este folosit pentru testarea aplicatiei.

### 3. Dezvoltarea aplicatiei

#### 3.1. Adaugarea url-ului aplicatiei

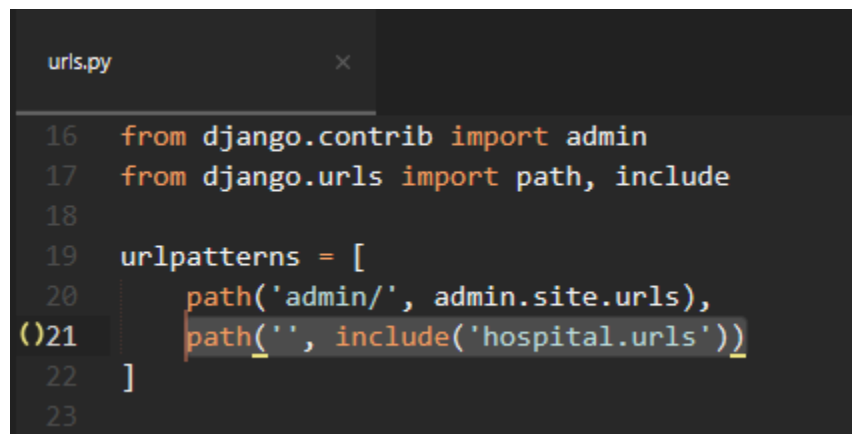
Odata creata aplicatia, primul lucru pe care trebuie sa il facem este sa modificam fisierul de `urls.py` din interiorul proiectului si sa legam url-urile proiectului cu cele ale aplicatiei. Pentru aceasta, se va deschide fisierul `urls.py` din interiorul proiectului si se va adauga un element nou in lista de `urlpatterns`. Practic, ii vom spune utilitarului Django ca in momentul accesarii url-ului respectiv, utilizatorul sa fie redirectionat catre aplicatia creata de noi. Pentru aceasta, avem nevoie intai sa importam o functie noua numita `"include"`.



```
urls.py
16 from django.contrib import admin
17 from django.urls import path, include
```

figura 14 includerea functiei "include"

Apoi, in lista de `urlpatterns` vom adauga url-ul pe care dorim sa il redirectionam si calea unde dorim sa redirectionam utilizatorul.



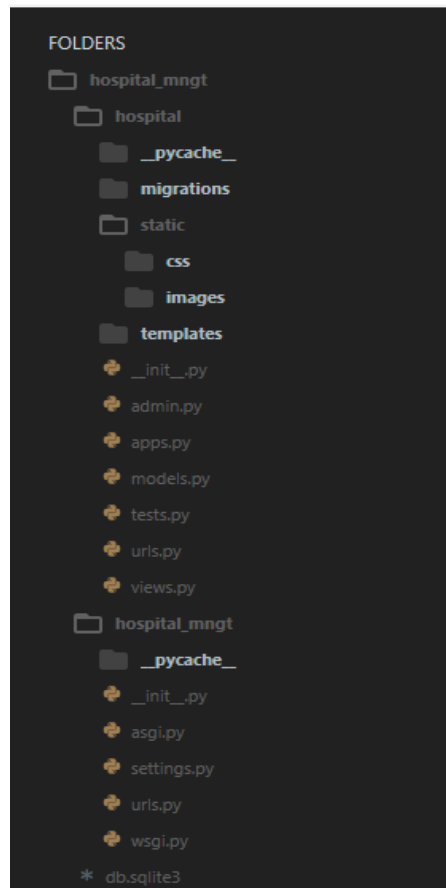
```
urls.py
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('hospital.urls'))
22 ]
23
```

figura 15 Adaugarea unui nou url in proiect

În cazul nostru, când utilizatorul accesează url-ul aplicației noastre, acesta va fi redirectionat către url-urile din interiorul aplicației create de noi, pe care le vom implementa ulterior.

### 3.2. Fișierele statice

Următorul pas este să creăm două noi directoare în interiorul aplicației noastre. Acestea sunt directorul *static* care va conține toate fișierele statice precum imaginile pe care le vom afișa în aplicația noastră web și fișierele ce țin de *css*, adică limbajul pe care îl vom folosi pentru a stiliza documentele *HTML*, și încă un director numit *templates* unde vom crea toate fișierele *HTML*, practice, în acest director vom crea paginile pe care dorim să le afișăm în aplicația noastră web. Directorul *static* este la rândul său compus din alte două directoare, unul *css* care va conține fișierele de stilizare discutate anterior, și unul *images* care va conține imaginile aplicației noastre. La sfârșit noua structură a aplicației noastre va arăta ca în figura de mai jos.

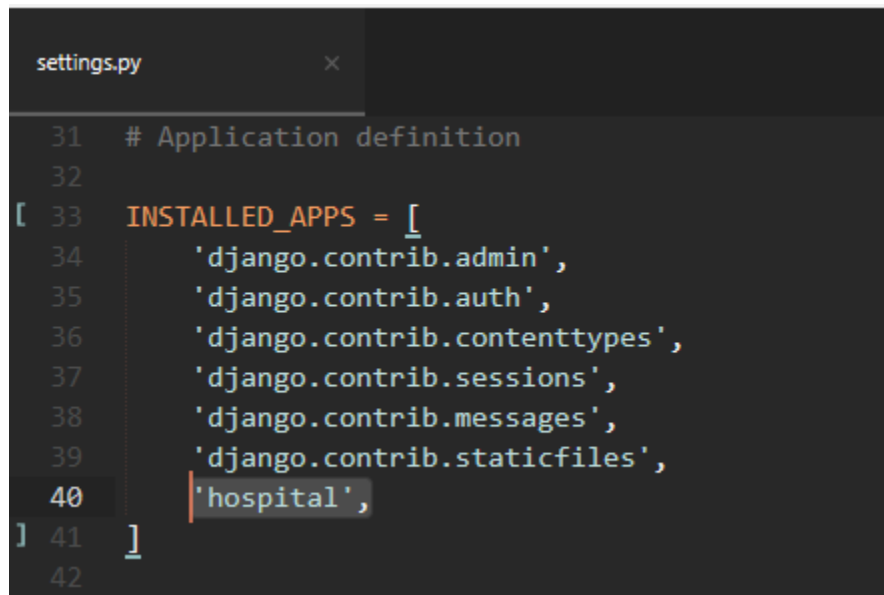


*figura 16 Noua structura a proiectului*

Totusi, pentru a putea vedea fisierele statice in browser mai este nevoie de cateva setari aditionale. Aceste setari se vor face in fisierul *settings.py* din interiorul proiectului nostru. Prima setare de care avem nevoie este sa adaugam aplicatia noastra in lista de aplicatii instalate ale proiectului. Deoarece in Django un proiect poate avea mai multe aplicatii, trebuie sa ii spunem lui Django ce aplicatii foloseste proiectul nostru. Desi aceasta setare nu are o legatura directa cu fisierele statice, fara ea tot ce vom dezvolta in interiorul aplicatiei noastre nu va fi luat in considerare, inclusive fisierele statice.



Adaugarea unei aplicatii noi in lista de aplicatii instalate se face ca in figura de mai jos, unde in lista de *INSTALLED\_APPS*, am adaugat aplicatia *hospital*.



```
settings.py
31 # Application definition
32
33 INSTALLED_APPS = [
34     'django.contrib.admin',
35     'django.contrib.auth',
36     'django.contrib.contenttypes',
37     'django.contrib.sessions',
38     'django.contrib.messages',
39     'django.contrib.staticfiles',
40     'hospital',
41 ]
42
```

figura 17 Adaugarea unei aplicatii noi in aplicatiile instalate

Urmatoarele doua setari au legatura directa cu fisierele statice, si anume aduagarea campurilor *STATIC\_URL* si *STATICFILES\_DIRS* pe care le voi prezenta in subcapitolele care urmeaza.

### 3.2.1. Campul *STATIC\_URL*

Cu ajutorul acestui camp vom defini URL-ul pe care il vom folosi cand vom face referire la fisierele statice situate in *STATIC\_ROOT* despre care vom discuta pe scurt in subcapitolul 3.2.3. *STATIC\_ROOT*. In mod implicit, acest camp este setat cu valoarea *None*, insemnand ca in mod implicit nu avem setat niciun url pentru aceste fisiere. Daca acest camp este diferit de *None*, valoarea url-ului pe care o vom da trebuie sa se termine cu *"/*.

Setarea acestui camp se face ca in figura de mai jos.

```
settings.py x
106
107 LANGUAGE_CODE = 'en-us'
108
109 TIME_ZONE = 'UTC'
110
111 USE_I18N = True
112
113 USE_TZ = True
114
115
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.0/howto/static-files/
118
119 STATIC_URL = '/static/'
```

figura 18 Setarea campului STATIC\_URL

### 3.2.2. Campul STATICFILES\_DIRS

În mod implicit, acest camp este o lista goală. Aceasta setare ne ajută să definim locația fișierelor noastre statice. Setarea acestui camp se face ca în figura de mai jos.

```
settings.py x
109 TIME_ZONE = 'UTC'
110
111 USE_I18N = True
112
113 USE_TZ = True
114
115
116 # Static files (CSS, JavaScript, Images)
117 # https://docs.djangoproject.com/en/4.0/howto/static-files/
118
119 STATIC_URL = '/static/'
120 STATICFILES_DIRS = [BASE_DIR, 'hospital/static']
```

figura 19 Setarea campului STATICFILES\_DIRS

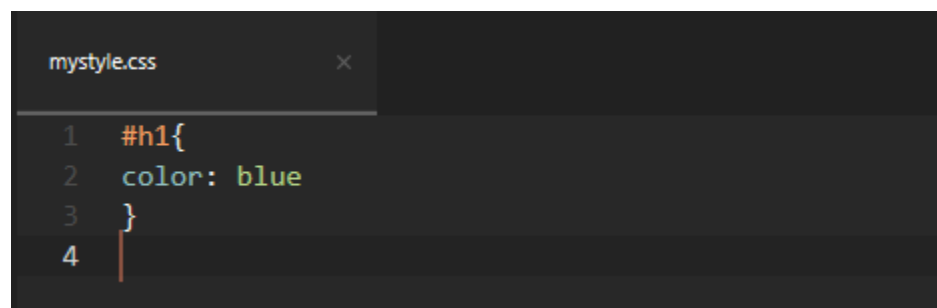
Aceasta setare semnaleaza ca fisierele statice ale aplicatiei sunt situate in directorul proiectului nostru Django (*BASE\_DIR*) in directorul *static* din directorul aplicatiei noastre *hospital* (*hospital/static*).

### 3.2.3. `STATIC_ROOT`

In mod implicit, acest camp este setat cu valoarea *“None”*. Acesta reprezinta calea absoluta a directorului de unde *collectstatic* va colecta fisierele statice pentru implementarea aplicatiei. *collectstatic* este o comanda de administrare a aplicatiei care colecteaza fisierele statice in *STATIC\_ROOT*.

### 3.2.4. Directorul static

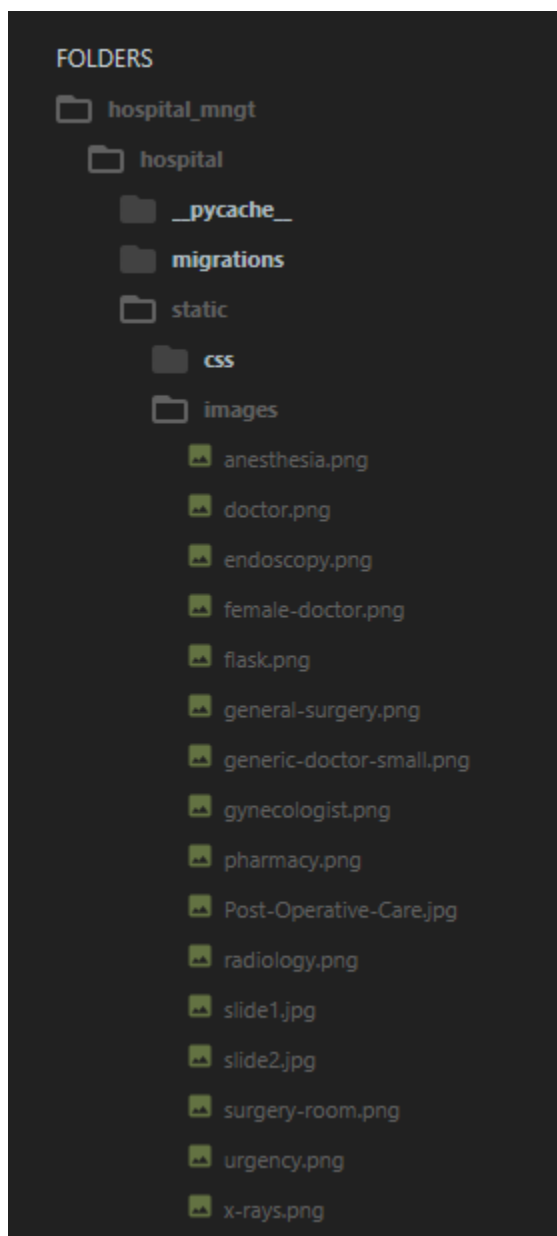
Asa cum am mentionat si anterior, acest director este la randul sau compus din alte doua directoare *“css”* si *“images”*. In directorul *“css”* am creat fisierul de stilizare *“mystyle.css”*. Acest fisier este un fisier simplu care doar seteaza culoarea header-ului 1 documentului *HTML* cu albastru. Acesta va fi folosit ulterior cand voi discuta despre fisierele *HTML*.



```
mystyle.css
1  #h1{
2    color: blue
3  }
4
```

figura 20 Fisierul *mystyle.css*

Directorul “*images*” contine imaginile pe care dorim sa le afisam in aplicatia noastra web. Aceste imagini sunt gratuite si pot fi folosite in interiorul aplicatiei noastre in mod legal. Sursa acestor imagini se poate gasi in capitolul [5. Referinte](#).



*figura 21 Directorul images*

### 3.3. Migrările și crearea de superusers

Migrările reprezintă modalitatea Django de a propaga modificările pe care le facem asupra modelelor noastre (adaugarea unui câmp, ștergerea unui model s.a.m.d) în schema bazei de date.

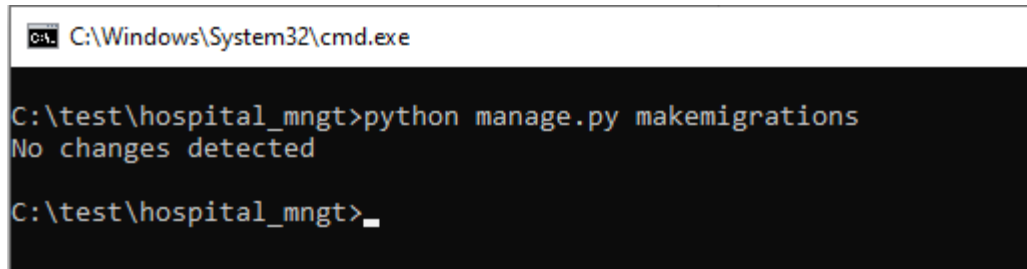
Există mai multe comenzi pentru a interacționa cu migrările și pentru a gestiona schema bazei de date. Acestea sunt:

- `migrate`: este responsabil pentru aplicarea și anularea migrărilor;
- `makemigrations`: este responsabil pentru crearea de noi migrări bazate pe modificările făcute asupra modelelor aplicației;
- `sqlmigrate`: afișează instrucțiunile SQL pentru o migrare;
- `showmigrations`: listează migrațiile unui proiect și starea acestora.

Putem considera *migrate* ca pe un sistem de control al versiunilor pentru schema bazei de date iar *makemigrations* este responsabil pentru împachetarea modificărilor modelului nostru în fișiere de migrare individuale, iar *migrate* este responsabil pentru aplicarea acestora în baza de date.

Django, în mod implicit creează pentru noi și o pagină de admin de unde putem administra operațiile asupra modelelor noastre și ne oferă și o interfață grafică pentru aceasta. Cu toate acestea, în cadrul aplicației vom crea toate funcționalitățile de care dispune această pagină fără a interacționa prea mult cu pagina implicită de admin a Django, însă în cadrul aplicației vom lucra cu *superusers*, acei utilizatori care au drepturi de administrator, aceasta fiind o aplicație de administrare.

Pentru a crea un *superuser* trebuie mai întâi să rulăm comanda “*makemigrations*” ca în figura de mai jos.



```
C:\Windows\System32\cmd.exe

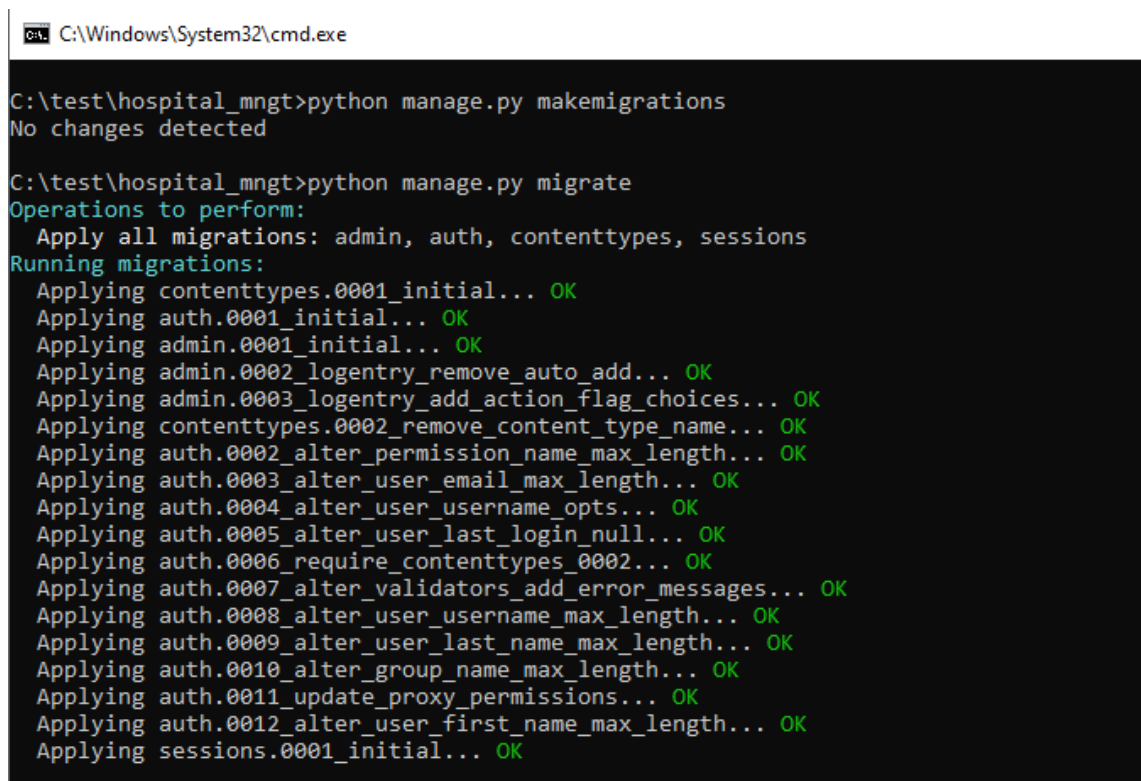
C:\test\hospital_mngt>python manage.py makemigrations
No changes detected

C:\test\hospital_mngt>_
```

figura 22 Executarea comenzii makemigrations

În urma executării acestei comenzi observăm că primim răspunsul că nu au fost detectate schimbări deoarece nu am făcut încă modificări în schema de bazei și nici nu am creat încă modelele aplicației noastre.

Următorul pas este să executăm comanda *“migrate”* care va aplica modificările noastre în schema bazei de date. Chiar dacă nu am făcut încă modificări în schema bazei de date, aceasta va aplica anumite setări implicite și va crea și tabela aferentă *superuser-ilor* de care avem nevoie pentru a crea un utilizator cu drepturi de administrator. Comanda se execută ca în figura de mai jos.



```
C:\Windows\System32\cmd.exe

C:\test\hospital_mngt>python manage.py makemigrations
No changes detected

C:\test\hospital_mngt>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying sessions.0001_initial... OK
```

figura 23 Executarea comenzii migrate

Acum putem crea un *superuser* executand comanda “*python manage.py createsuperuser*” ca in figura de mai jos.

```
C:\test\hospital_mngt>python manage.py createsuperuser
Username (leave blank to use 'bogdanhasan'): admin
Email address: admin@yahoo.com
Password:
Password (again):
The password is too similar to the username.
This password is too short. It must contain at least 8 characters.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.

C:\test\hospital_mngt>
```

figura 24 Crearea unui superuser

Pentru aceasta aplicatie am ales ca utilizatorul sa fie *admin* cu parola *admin*. Se poate observa ca Django ne antentioneaza ca parola este prea slaba si similara cu numele utilizatorului, insa putem ocoli aceste avertismente, insa pentru aplicatiile care ruleaza in medii de productie este recomandat sa alegem parole mai greu de ghicit.

### 3.4. Pagina initiala Django de administrare

Acum ca avem un *superuser* create, putem porni server-ul si naviga catre pagina de administrator. Pentru a porni server-ul se executa comanda ca mai jos.

```
C:\Windows\System32\cmd.exe - python manage.py runserver

C:\test\hospital_mngt>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

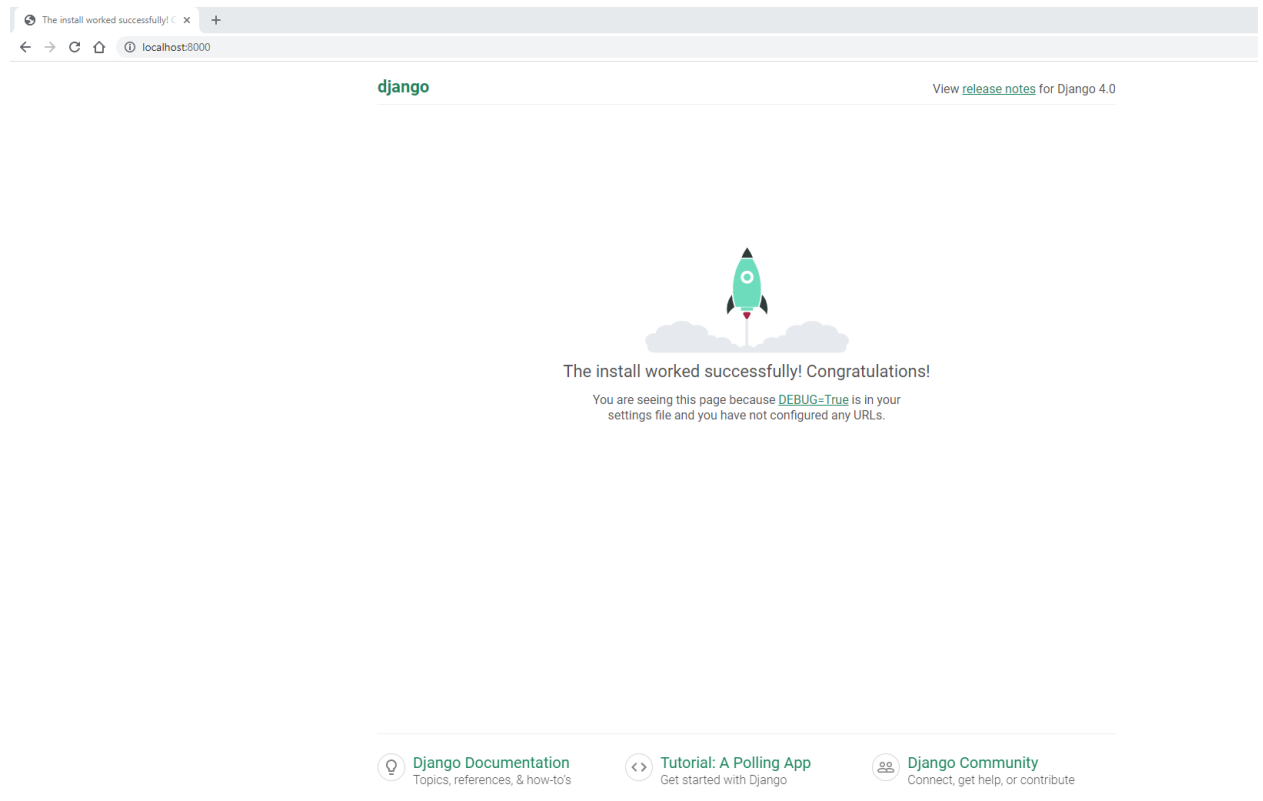
System check identified no issues (0 silenced).
May 03, 2022 - 19:46:35
Django version 4.0, using settings 'hospital_mngt.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

figura 25 Pornirea server-ului

In aceasta fereastră putem observa ca server-ul a pornit la adresa “127.0.0.1:8000” reprezentand adresa la care poate fi gasita aplicatia(127.0.0.1 sau localhost) si port-ul la care asculta aplicatia noastra (8000). De asemenea, in

cazul in care apar erori de cod, mesajele de eroare vor fi afisate in aceasta fereastra.

Odata pornit server-ul, putem accesa aplicatia din browser ca in figura de mai jos.



*figura 26 Accesarea aplicatiei*

Putem observa si mesajul standard care ne confirma ca instalarea aplicatiei a fost cu succes, insa nu avem inca paginile noastre create asa ca ne este afisat un mesaj standard.

Pentru a accesa pagina de admin discutata anterior, la adresa url-ului se va adauga `"/admin"` conform cu fisierul `urls.py` din cadrul aplicatiei.



```

urls.py
16 from django.contrib import admin
17 from django.urls import path, include
18
19 urlpatterns = [
20     path('admin/', admin.site.urls),
21     path('', include('hospital.urls'))
22 ]
23

```

figura 27 URL-ul admin din urls.py din cadrul proiectului

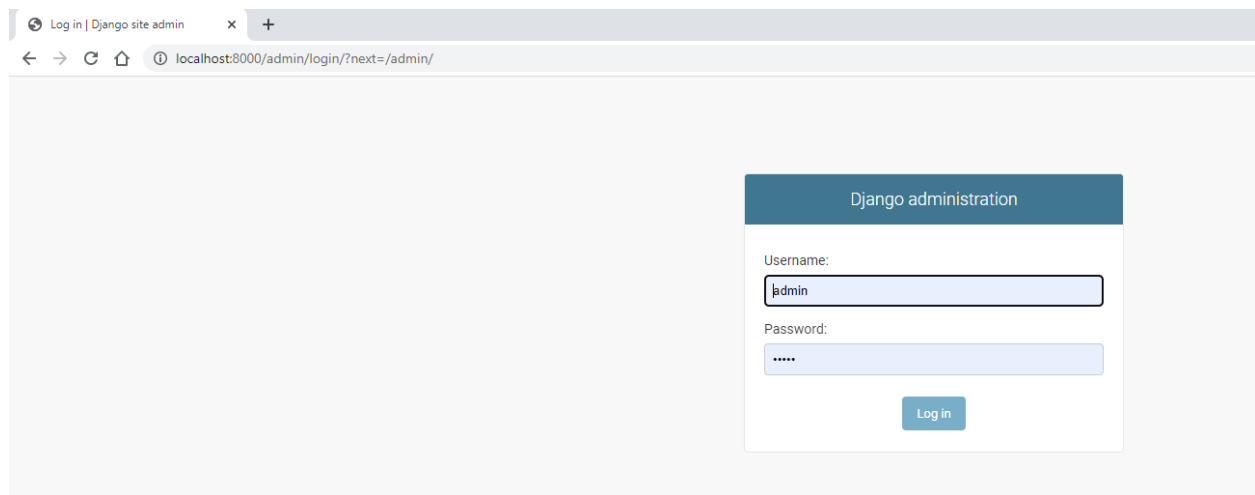


figura 28 URL-ul admin din browser

Se poate observa ca atunci cand accesam pagina de admin ni se va cere sa ne autentificam ca utilizatorul si parola unui *superuser*.

Odata accesat, din interiorul paginii putem face modificari asupra grupurilor si a utilizatorilor, insa nu vom lucra cu aceasta pagina, acesta fiind mai degraba o prezentare asupra functionalitatilor Django.

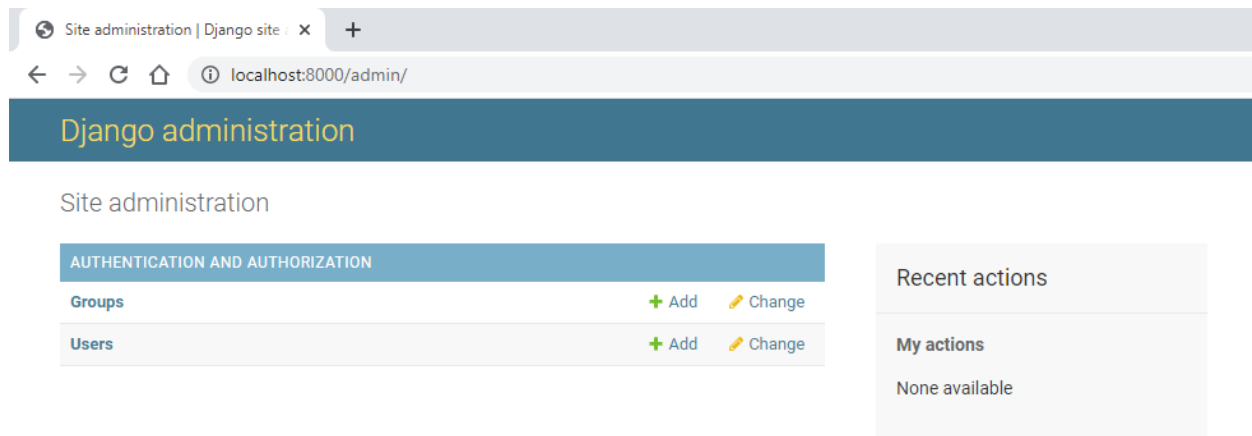


figura 29 Pagina de administrator Django

### 3.5. Directorul templates

Asa cum am mentionat anterior, acest director contine toate paginile HTML ale aplicatiei noastre. Folosim aceasta structura deoarece ne ajuta sa avem un proiect bine organizat si curat si ne ajuta atunci cand depanam diversele probleme care apar in dezvoltarea aplicatiei. In cazul in care primim o eroare vom sti exact care este fisierul care a cauzat eroarea, unde este situat acesta, si in functie de locatia acestuia vom sti si care este rolul acestuia.

Primul fisier *HTML* pe care il vom crea este fisierul *home.html* descris in subcapitolul urmator.

#### 3.5.1. Fisierul home.html

Acest fisier reprezinta pagina *HTML* pe care utilizatorul o va vedea prima oara cand va accesa aplicatia dezvoltata.

Incepem dezvoltarea acestei pagini prin incarcarea fisierelor statice (`{% load static %}`) si includerea altor pagini care au legatura directa cu pagina curenta (`{% include 'base.html' %}` si `{% include 'navigation.html' %}`), pagini pe care le voi descrie in subcapitolele ce urmeaza. Important de precizat aici sunt semnele de acolada si de procentaj "`{%%}`". Acestea ne ajuta sa scriem cod de Python in interiorul acestora.

Fisierul *home.html* este compus din mai multe sectiuni, prima sectiune fiind un diapozitiv de tip carusel cu trei imagini, fiecare imagine avand si un paragraf care descrie ce reprezinta fiecare din acele imagini, dar si doua butoane de inainte si inapoi pe care le vom folosi pentru a trece la urmatoarea imagine sau la imaginea precedenta.

Putem observa in inceputul sectiunii crearea diapozitivului cu trei imagini

```
<div id="carousel-slider" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
    <li data-target="#carousel-slider" data-slide-to="0" class="active"></li>
    <li data-target="#carousel-slider" data-slide-to="1"></li>
    <li data-target="#carousel-slider" data-slide-to="2"></li>
  </ol>
```

figura 30 Diapozitivul de tip carusel

Acesta este urmat de elementele diapozitivului, primul fiind pentru ingrijirea posoperatorie, unde este inserata o imagine reprezentativa "*Post-Operative-Care.jpg*" din directorul *images*, urmat de un paragraf cu definitia ingrijirii post operatorii.

```
<div class="carousel-inner">
  <div class="carousel-item active">
    
    <div class="carousel-caption text-white">
      <div class="background-1">
        <h2>Îngrijire postoperatorie</h2>
        <p>Îngrijirea postoperatorie este îngrijirea pe care o primiți după o intervenție chirurgicală. Îngrijirea de care aveți nevoie depinde de tipul de intervenție chirurgicală pe care o aveți, precum și de istoricul dumneavoastră de sănătate. Adesea include gestionarea durerii și îngrijirea rănilor.</p>
        <button class="btn btn-light"><a href="#departments" class="nav-link">Vezi toate serviciile</a></button>
      </div>
    </div>
  </div>
```

figura 31 Elementul Ingrijire postoperatorie

Se mai poate observa de asemenea si un buton “*Vezi toate serviciile*”, care este de fapt un link catre sectiunea de departamente din interiorul aceleasi pagini pe care o voi prezenta ulterior.

In mod similar sunt create si celelalte doua elemente ale diapozitivului, Interventia Chirurgicala si Unitatea de terapie intensiva neonatala.

```
<div class="carousel-item">
  
  <div class="carousel-caption text-white">
    <div class="background-1">
      <h2>Intervenția Chirurgicală</h2>
      <p>Chirurgia este o specialitate medicală care utilizează tehnici operative manuale și instrumentale asupra unei persoane pentru a investiga sau trata o afecțiune patologică, cum ar fi o boală sau o leziune, pentru a ajuta la îmbunătățirea funcția sau aspectul corporal sau pentru a repara zonele rupte nedorite.</p>
      <button class="btn btn-light"><a href="#departments" class="nav-link">Vezi toate serviciile</a></button>
    </div>
  </div>
</div>
```

*figura 32 Elementul Interventia chirurgicala*

```
<div class="carousel-item">
  
  <div class="carousel-caption text-white">
    <div class="background-1">
      <h2>Unitatea de terapie intensivă neonatală</h2>
      <p>O unitate de terapie intensivă neonatală, cunoscută și sub numele de creșă de terapie intensivă, este o terapie intensivă unitate specializată în îngrijirea nou-născuților bolnavi sau prematuri. Neonatal se referă la primul 28 de zile de viață. Îngrijirea neonatală, cunoscută sub numele de creșe specializate sau terapie intensivă, a fost în jur din anii 1960.</p>
      <button class="btn btn-light"><a href="#departments" class="nav-link">Vezi toate serviciile</a></button>
    </div>
  </div>
</div>
```

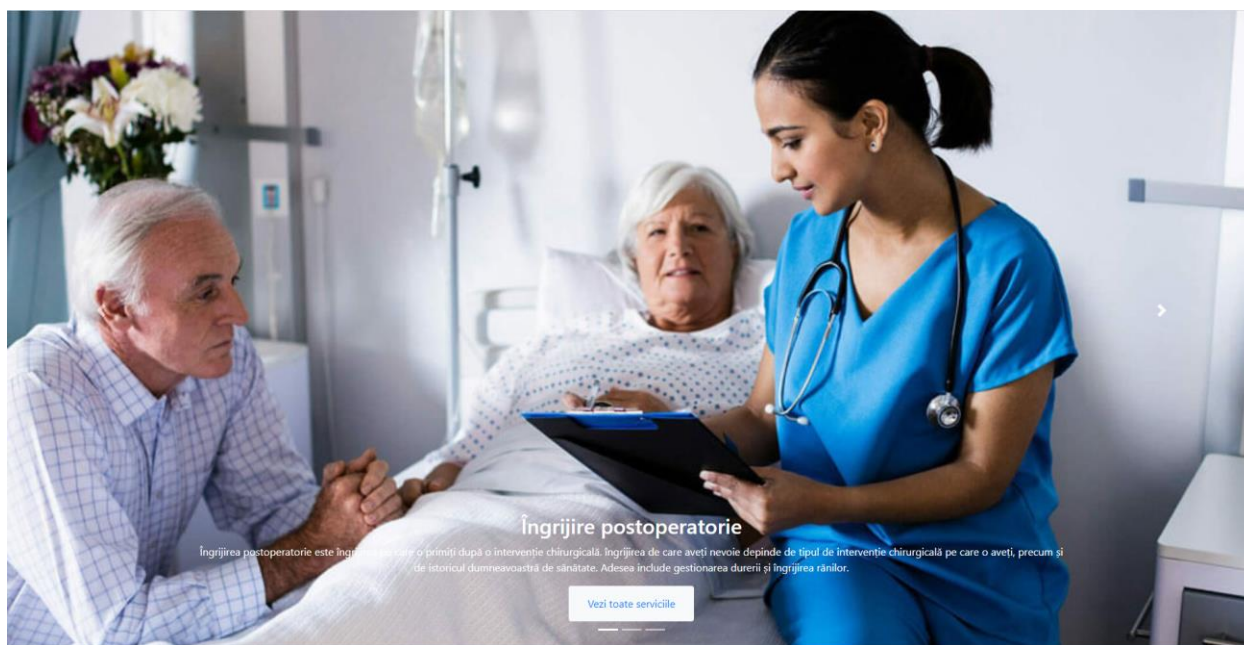
*figura 33 Unitatea de terapie intensiva neonatala*

Sectiunea se incheie cu cele doua butoane de “Inainte” si “Inapoi”.

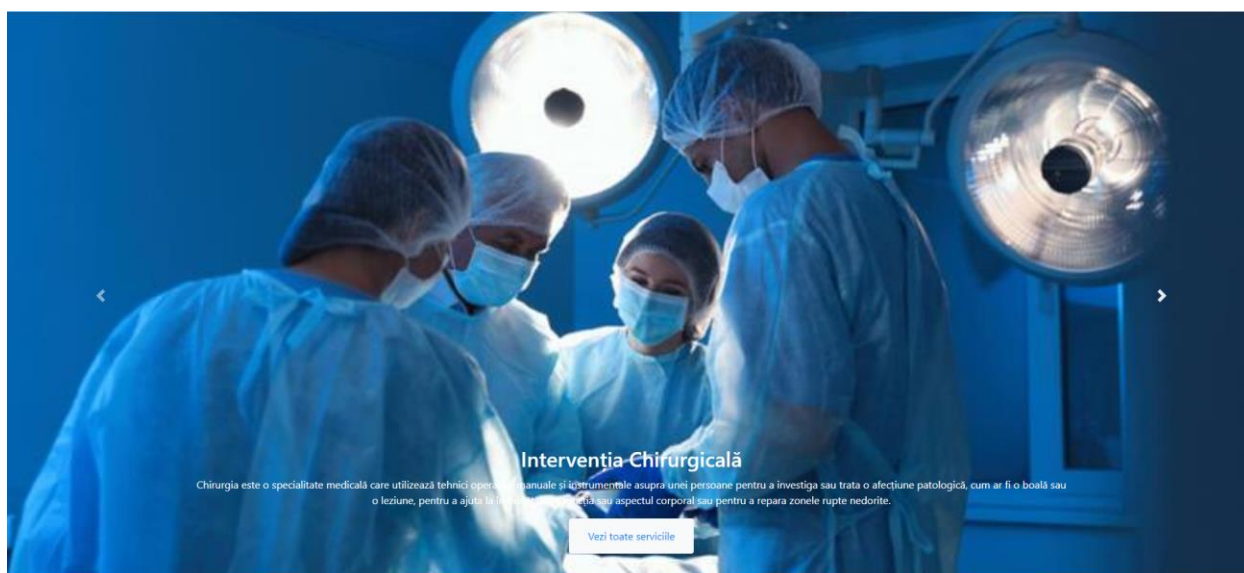
```
<a class="carousel-control-prev" href="#carousel-slider" role="button" data-slide="prev">
  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
  <span class="sr-only">Înapoi</span>
</a>
<a class="carousel-control-next" href="#carousel-slider" role="button" data-slide="next">
  <span class="carousel-control-next-icon" aria-hidden="true"></span>
  <span class="sr-only">Înainte</span>
</a>
```

*figura 34 Butoanele Inainte si Inapoi*

Rezultatul final este prezentat in figurile ce urmeaza.



*figura 35 Ingrijire postoperatorie*



*figura 36 Interventie Chirurgicala*







```

<div class="container-fluid" id="departments">
  <div class="row">
    <div class="col-md-12">
      <h2 class="text-center text-primary">Departamente</h2>
      <br>
      <h1 class="text-center">Servicii Medicale</h1>
    </div>
  </div>
</div>

```

figura 42 Identificatorul departments

Acesta este urmat de sase blocuri cu imagini reprezentative departamentului respectiv. Toate blocurile sunt create in mod similar, diferenta facand doar imaginea inserata.

```

<div class="container-fluid">
  <div class="row owl-carousel owl-theme">
    <div class="card shadow text-center items" style="height: 159px;">
      <div class="card-img">
        
      </div>
      <div class="card-body">
        <p class="card-text">Ginecolog si Obstetrician</p>
      </div>
    </div>
  </div>

```

figura 43 Blocul de imagini al departamentelor

Rezultatul final este prezentat in figura urmatoare.



figura 44 Sectiunea servicii medicale

Urmeaza o sectiune asemanatoare celei prezentate anterior, numita sectiunea de "Alte servicii".



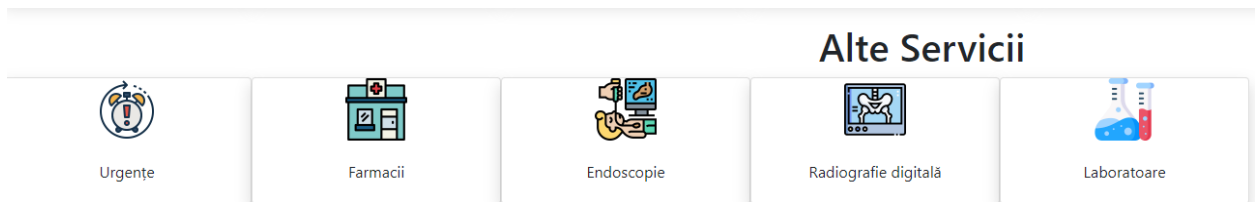


figura 45 Secțiunea alte servicii

În final, vom încheia cu secțiunea în care se prezintă doctorii. De asemenea este prezent și aici un identificator “doctors” la care butonul “Găsește doctori” face referire.

```
<div class="container-fluid" id="doctors">
  <div class="row">
    <div class="col-md-12">
      <h4 class="text-center text-primary">Întâlnește-ne echipa</h4>
      <h1 class="text-center">Doctorii cu care ne mândrim</h1>
    </div>
  </div>
</div>
```

figura 46 Identificatorul doctors

Voi continua cu câte un bloc pentru fiecare doctor cu o scurtă prezentare a specializării acestuia și cu câte o imagine reprezentativă. Fiecare bloc ce descrie un doctor este implementat în mod similar.

```
<div class="row owl-carousel owl-theme">
  <div class="card shadow text-center items">
    <div class="card-body profile-card">
      
      <h4 class="card-title" style="font-size: 18px;">Dr. Maria Grigorescu</h4>
      <h6 class="card-subtitle">Chirurg Consultant Senior</h6>
      <p class="card-text">MBBS, MS(TU), FMAS, DMAS (WLH)</p>
      <a href="#" class="card-link"><i class="fa fa-facebook"></i></a>
      <a href="#" class="card-link"><i class="fa fa-instagram"></i></a>
      <a href="#" class="card-link"><i class="fa fa-linkedin"></i></a>
    </div>
  </div>
</div>
```

figura 47 Blocul doctorilor

Rezultatul final este prezentat mai jos.



figura 48 Secțiunea doctorilor

### 3.5.2. Pagina about.html

Aceasta este pagina “Despre noi” care conține o scurtă prezentare a spitalului.

```
{% include 'base.html' %}
{% include 'navigation.html' %}

{% block content %}
<h1 class="text-center" style="margin-top: 50px;">Despre noi</h1>
<div class="container" style="margin-top: 100px;">
  <div class="row">
    <div class="col-md-6">
      <h1 class="strong">Cine suntem<br>Și ce facem</h1>
      <p class="lead">Acesta este portalul lider mondial pentru<br>managerierea spitalelor </p>
    </div>
    <div class="col-md-6">
      <p>Cu ajutorul serviciilor noastre, vă puteți asigura că veți beneficia de servicii medicale de calitate în confortul casei dumneavoastră.</p>
      <p>Oferim specialiști, însoțitori și kinetoterapeuți pentru vizite la domiciliu care au trecut de principiile noastre aprofundate de îngrijire și au avut experiențele și învățarea restaurativă verificate de specialiști seniori.</p>
    </div>
  </div>
  <div class="row">
    <div class="site-heading text-center">
      <h3>Echipa noastră</h3>
      <p>Specialiștilor noștri sunt persoane din organisme de acreditare terapeutică universală. Datorită utilizării administrațiilor noastre, pacienții noștri pot rămâne în casele lor mai mult timp, pot pune deoparte bani în plus și pot avea sentimente autentice de liniște.</p>
    </div>
  </div>
</div>
</div>
{% endblock content %}
{% include 'footer.html' %}
```

figura 49 about.html

## Despre noi

### Cine suntem Și ce facem

Acesta este portalul lider mondial pentru  
managerierea spitalelor

Cu ajutorul serviciilor noastre, vă puteți asigura că veți beneficia de servicii  
medicale de calitate în confortul casei dumneavoastră.

Oferim specialiști, însoțitori și kinetoterapeuți pentru vizite la domiciliu care  
au trecut de principiile noastre aprofundate de înrolare și au avut  
experiențele și învățarea restaurativă verificate de specialiști seniori.

### Echipa noastră

specialiștilor noștri sunt persoane din organisme de acreditare terapeutică universală. Datorită utilizării administrațiilor noastre, pacienții noștri pot rămâne în casele  
lor mai mult timp, pot pune deoparte bani în plus și pot avea sentimente autentice de liniște.


figura 50 Pagina despre noi


### 3.5.3. Pagina contact.html

Aceasta este pagina prin care utilizatorul poate vedea metodele prin care poate lua legătura cu spitalul. Aceasta este formata din patru blocuri, unul pentru contactul prin telefon, unul pentru contactul prin e-mail, pentru adresa sediului si inca un bloc pentru programul spitalului. Ca si in cazul paginii [home.html](#) blocul pentru contact prin telefon va deschide o fereastră de tip pop-up cu optiunea de a apela la numarul respectiv, blocul "Trimite-ne un e-mail" va deschide o fereastră de tip pop-up cu posibilitatea de a trimite un e-mail la adresa respective, iar blocul cu adresa va redirectiona catre o pagina de *Google Maps* cu adresa spitalului. De notat la aceasta pagina este stilizarea de la inceputul acesteia care face referire si la fisierul [mystyle.css](#) si care creeaza un effect de planare atunci cand trecem cu mouse-ul peste unul din blocuri.

### la legătura cu noi

 **Contactează-ne prin  
telefon**  
[+40 000 000 000](#)

 **Trimite-ne un e-mail**  
[abc@gmail.com](#)

 **Adresa sediului**  
[Bucuresti](#)

 **Programul Spitalului**  
[Deschis Nonstop](#)

figura 51 Efectul de planare al blocurilor din contact.html

```

{% include 'base.html' %}
{% include 'navigation.html' %}
<style>
    .mydivstyle:hover {
        box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.3);
        transform: translateY(-20px);
    }
</style>

{% block content %}
<div class="container" style="margin-top: 50px;">
    <h1 class="text-center">Ia legătura cu noi</h1>
    <div class="row mt-5 text-center">
        <div class="col-lg-3 col-sm-6 mydivstyle">
            <h4><i class="fas fa-phone-square"> Contactează-ne prin telefon</i></h4>
            <p class="text-danger font-weight-bold mt-3"><a href="tel:+40 000 000 000">+40 000 000 000</a></p>
        </div>
        <div class="col-lg-3 col-sm-6 mydivstyle">
            <h4><i class="fa fa-envelope"></i> Trimite-ne un e-mail</h4>
            <p class="text-danger font-weight-bold mt-3"><a href="mailto:abc@gmail.com">abc@gmail.com</a></p>
        </div>
        <div class="col-lg-3 col-sm-6 mydivstyle">
            <h4><i class="fa fa-address-card"></i> Adresa sediului </h4>
            <p class="text-danger font-weight-bold mt-3"><a href="http://maps.google.com/maps?q='Bucuresti'">Bucuresti</a></p>
        </div>
        <div class="col-lg-3 col-sm-6 mydivstyle">
            <h4><i class="fa fa-clock"></i> Programul Spitalului </h4>
            <p class="font-weight-bold mt-3">Deschis Nonstop</p>
        </div>
    </div>
</div>
{% endblock content %}
{% include 'footer.html' %}

```

figura 52 Pagina contact.html

### 3.5.4. Pagina login.html

Urmatoarea pagina despre care voi vorbi este pagina *login.html*. In aceasta pagina utilizatorul se poate autentifica in aplicatie pentru a putea beneficia de functiile de administrator ale aplicatiei. Incep dezvoltarea paginii prin doua clauze de "if". Acestea sunt necesare pentru a verifica daca utilizatorul a introdus datele de autentificare corect.

```

1  {% include 'base.html' %}
2  {% include 'navigation.html' %}
3  {% block content %}
4
5  {% if error == "no" %}
6  <script>
7      alert('Autentificare cu succes!');
8      window.location = "{% url 'dashboard' %}"
9  </script>
10 {% endif %}
11 {% if error == "yes" %}
12 <script>
13     alert('Autentificare invalidă! Mai încearcă o dată.');
```

figura 53 Clauzele "if" din pagina login.html

Fiecare din aceasta clauza va genera un mesaj de alerta in functie daca datele de autentificare introduse au fost corecte, caz in care in pagina va aparea un mesaj "Autentificare cu succes", sau incorecte, caz in care se va afisa mesajul "Autentificare invalida! Mai incearca o data.". Important de notat in aceasta pagina este faptul ca cele doua clauze "if" se folosesc de o variabila "error" pentru a putea verifica daca datele introduse sunt corecte sau nu. Mai multe despre aceasta variabila si despre modul in care este asignata valoarea acesteia voi discuta in capitolul *views.py*.

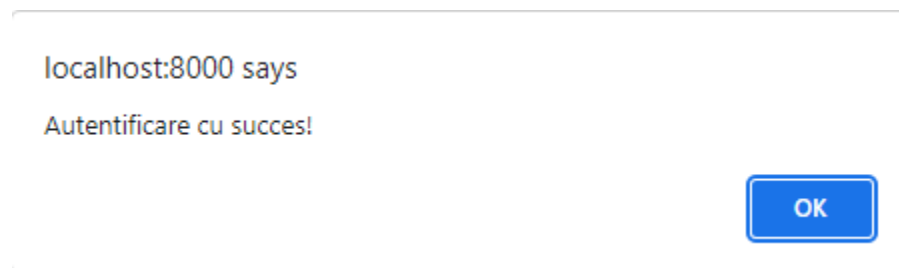


figura 54 Autentificare cu succes

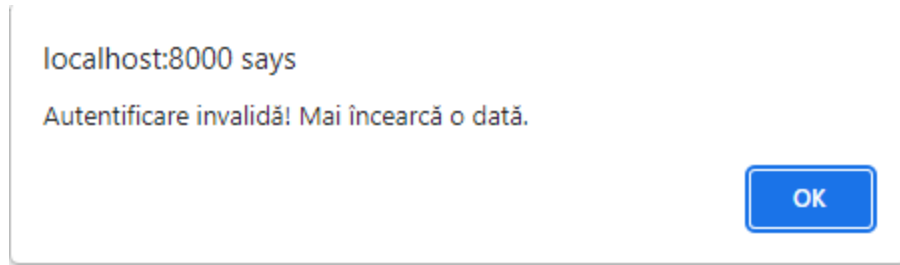


figura 55 Autentificare invalida

In continuare, voi crea campurile necesare pentru utilizator si parola.

```
<div class="container">
  <h2 class="text-center">Autentificare Admin</h2>
  <form method="POST">
    {% csrf_token %}
    <label>Numele de utilizator</label>
    <input type="text" class="form-control" name="uname">
    <label>Parola</label>
    <input type="password" class="form-control" name="pwd">

    <input type="submit" value="Autentificare" class="form-control btn btn-primary">
  </form>
</div>
```

figura 56 Campurile pentru utilizator si parola

---

### Autentificare Admin

Numele de utilizator

Parola

Autentificare

figura 57 Pagina de autentificare

### 3.5.5. Pagina add\_docotor.html

Aceasta este pagina prin care administratorul poate adauga un doctor in aplicatie. Insa, pentru a putea face acest lucru trebuie mai intai sa creem modelul pentru doctori. In subcapitolele ce urmeaza voi crea modelele pentru doctori, pacienti si pentru programari.

### 3.5.5.1. Modelul doctorilor

Modelul doctorilor reprezinta datele de care avem nevoie pentru a crea un doctor in baza de date. Acest model este de fapt o clasa cu variabilele care definesc un obiect de tip doctor. Pentru aceasta, se va deschide fisierul *models.py* si vom crea clasa *Doctor* cu campurile de *name* pentru numele doctorului, acest camp fiind de tip Char cu o lungime maxima de 50 de caractere, *mobile*, camp de tip Integer si *special*, pentru specialitatea doctorului, acest camp fiind tot un camp de tip Char cu lungime maxima de 50 de caractere.

```
from django.db import models

# Create your models here.
class Doctor(models.Model):
    name = models.CharField(max_length=50)
    mobile = models.IntegerField()
    special = models.CharField(max_length=50)

    def __str__(self):
        return self.name
```

figura 58 Modelul Doctor

De mentionat este faptul ca la finalul clasei, am definit si o functie “\_\_str\_\_(self)”. Scopul acestei functii este de a intoarce campul care ni se pare relevant atunci cand dorim sa facem o interogare in baza de date. In cazul de fata, atunci cand dorim sa interogam un doctor din baza de date, acesta ne va intoarce numele doctorului respectiv.

Pentru a putea face interogari in baza de date, trebuie mai intai sa migram modificarile facute asupra modelelor create. Acest lucru se realizeaza prin executarea comenzile de *makemigrations* si *migrate* descrise in capitolul [3.3. Migrarile si crearea de superusers](#).

```
C:\Windows\System32\cmd.exe

C:\test\hospital_mngt>python manage.py makemigrations
Migrations for 'hospital':
  hospital\migrations\0001_initial.py
    - Create model Doctor

C:\test\hospital_mngt>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, hospital, sessions
Running migrations:
  Applying hospital.0001_initial... OK
```

figura 59 Migrarea modelelor

#### 3.5.5.2. Modelul pacientilor

Similar cu modelul doctorilor descries anterior, se va crea modelul *Patient*, cu campurile *name*, de tip Char cu lungime maxima de 50 de caractere, *gender*, tot de tip Char, cu lungime maxima de 10 caractere reprezentand sexul pacientului, *mobile*, de tip Integer, in acest caz, spre deosebire de campul cu acelasi nume din modelul *Doctor* poate fi *null*, si campul *address*, de tip Text, reprezentand adresa pacientului. Ca si in cazul doctorilor, acest model are implementata functia “*\_\_str\_\_(self)*” care va intoarce numele pacientului.

```
class Patient(models.Model):
    name = models.CharField(max_length=50)
    gender = models.CharField(max_length=10)
    mobile = models.IntegerField(null=True)
    address = models.TextField()

    def __str__(self):
        return self.name
```

figura 60 Modelul Patient



### 3.5.5.3. Modelul programarilor

Pentru crearea modelului programarilor, ma folosesc de celelalte doua modele create anterior, *Doctor* si *Patient*, avand in vedere faptul ca un pacient isi poate face programare la un doctor. In acest caz, in clasa *Appointment* voi crea doua campuri de tip *ForeignKey*, unul pentru doctor si unul pentru pacient. De notat la aceste doua campuri este faptul ca atunci cand unul din campuri este sters, se vor sterge toate obiectele care referentiaza obiectul sters. Acest lucru se realizeaza prin "*on\_delete=models.CASCADE*". Pe langa aceste doua campuri, o programare mai are nevoie si de o data si o ora la care se face programarea, acestea realizandu-se prin campurile *date* de tip *Date* si *time* de tip *Time*.

```
class Appointment(models.Model):
    doctor = models.ForeignKey(Doctor, on_delete=models.CASCADE)
    patient = models.ForeignKey(Patient, on_delete=models.CASCADE)
    date = models.DateField()
    time = models.TimeField()

    def __str__(self):
        return self.Doctor.name + "_" + self.Patient.name
```

figura 61 Modelul programarilor

Acum ca toate modelele sunt create voi discuta despre pagina *add\_doctor.html*. Ca si in cazul paginii *login.html* aceasta pagina incepe prin doua clauze "if" care verifica daca datele introduse sunt corecte, caz in care se va afisa mesajul "*Record salvat cu success!*", sau daca datele introduse sunt invalide, caz in care se va afisa mesajul "*Recordul nu a putut fi salvat! Va rugam mai incercati o data.*".

```

{% include 'base.html' %}
{% include 'navigation.html' %}

{% block content %}
{% if error == "no" %}
<script>
    alert('Record salvat cu succes!');
    window.location = "{% url 'view_doctor' %}"
</script>
{% endif %}
{% if error == "yes" %}
<script>
    alert('Recordul nu a putut fi salvat! Vă rugăm mai încercați o dată.');

```

*figura 62 Clauzele if pentru adaugarea doctorului*

In continuare, voi crea campurile specifice pentru numele doctorului, numarului de telefon si pentru specialitatea acestuia.

```

<div class="container">
    <h2 class="text-center">Adaugă un doctor</h2>
    <form method="POST">
        {% csrf_token %}
        <label>Numele Doctorului</label>
        <input type="text" class="form-control" name="name">
        <label>Telefonul Mobil</label>
        <input type="text" class="form-control" name="mobile">
        <label>Specialitatea</label>
        <input type="text" class="form-control" name="special">

        <input type="submit" value="Trimitere" class="form-control btn btn-primary">
    </form>
</div>

```

*figura 63 Campurile pentru adaugarea unui doctor*

## Adaugă un doctor

Numele Doctorului

Telefonul Mobil

Specialitatea

Trimite

*figura 64 Pagina add\_doctor.html*

### 3.5.6. Pagina add\_patient.html

Aceasta pagina a fost implementata similar celei descrise anterior, diferenta facand-o doar campurile specifice unui model de tip *Patient*.

```

1  {% include 'base.html' %}
2  {% include 'navigation.html' %}
3
4  {% block content %}
5  {% if error == "no" %}
6  <script>
7      alert('Record salvat cu succes!');
8      window.location = "{% url 'view_patient' %}"
9  </script>
10 {% endif %}
11 {% if error == "yes" %}
12 <script>
13     alert('Recordul nu a putut fi salvat! Vă rugăm mai încercați o data.');

```

figura 65 Pagina add\_patient.html

---

## Adaugă un pacient

Numele Pacientului

Sexul

Telefonul Mobil

Adresa

submit

figura 66 Afisarea paginii add\_patient.html

### 3.5.7. Pagina add\_appointment.html

Aceasta este pagina in care se pot face programari. De notat la aceasta pagina este faptul ca pentru doctori si pacienti se va face o itererare prin doctorii si pacientii din baza de date. Astfel, administratorul poate alege unul din pacientii din baza de date caruia sa l se faca o programare la unul din doctorii din baza de date.

```

{% include 'base.html' %}
{% include 'navigation.html' %}

{% block content %}
{% if error == "no" %}
<script>
    alert('Record salvat cu succes!');
    window.location = "{% url 'view_appointment' %}"
</script>
{% endif %}
{% if error == "yes" %}
<script>
    alert('Recordul nu a putut fi salvat! Vă rugam mai încercați o data.');
```

figura 67 Pagina add\_appointment.html

---

Fă o programare

Numele Doctorului

Maria Grigorescu

Numele Pacientului

Alexandra Vasilescu

Data

mm/dd/yyyy

Ora

--:-- --

submit

figura 68 Afisarea paginii `add_appointment.html`

### 3.5.8.      Paginile `view_doctor.html`, `view_patient.html` si `view_appointment.html`

Aceste trei pagini sunt create in mod similar si reprezinta paginile pentru afisarea doctorilor, pacientilor si respectiv programarilor. De notat la aceste pagini este faptul ca pentru fiecare pagina se face o iterare asupra datelor din baza de date, spre exemplu, toti doctorii vor fi salvati in variabila *doc* si apoi se face o iterare prin aceasta variabila *“for l in doc”*, apoi pentru fiecare record se va afisa id-ul respective *“i.id”*, aceasta valoarea fiind incrementata in mod automat de Django, apoi si vor afisa numele, telefonul mobil si specialitatea doctorului. *“i.name, i.mobile, i.special”*. Acelasi lucru se intampla si pentru programari, unde toate programarile sunt salvate in variabila *apt*, si pentru pacienti, *pat*, pentru fiecare afisandu-se campurile specifice modelului corespunzator. De notat mai este si faptul ca fiecare pagina ofera posibilitatea de a sterge un record. Odata apasat butonul de *Sterge* se va afisa un mesaj de confirmare *“Sunteti siguri ca vreti sa stergeti acest record?”*.

```

{% include 'base.html' %}
{% include 'navigation.html' %}
{% block content %}
<div>
  <h2>Afişare Doctori</h2>
  <table id="example" class="table table-hover">
    <thead>
      <tr>
        <th>Id-ul Doctorului</th>
        <th>Numele</th>
        <th>Telefonul Mobil</th>
        <th>Specialitatea</th>
        <th>Şterge</th>
      </tr>
    </thead>
    <tbody>
      {% for i in doc %}
      <tr>
        <td>{{i.id}}</td>
        <td>{{i.name}}</td>
        <td>{{i.mobile}}</td>
        <td>{{i.special}}</td>
        <td><a href="{% url 'delete_doctor' i.id %}"
          onclick="return confirm('Sunteţi siguri că vreţi să ştergeţi acest record?')" class="btn btn-danger">Şterge</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
{% endblock content %}
{% include 'footer.html' %}

```

figura 69 Pagina view\_doctor.html

```

{% include 'base.html' %}
{% include 'navigation.html' %}
{% block content %}
<div>
  <h2>Afişare Pacienţi</h2>
  <table id="example" class="table table-hover">
    <thead>
      <tr>
        <th>Id-ul Pacientului</th>
        <th>Numele</th>
        <th>Sexul</th>
        <th>Telefonul Mobil</th>
        <th>Adresa</th>
        <th>Şterge</th>
      </tr>
    </thead>
    <tbody>
      {% for i in pat %}
      <tr>
        <td>{{i.id}}</td>
        <td>{{i.name}}</td>
        <td>{{i.gender}}</td>
        <td>{{i.mobile}}</td>
        <td>{{i.address}}</td>
        <td><a href="{% url 'delete_patient' i.id %}"
          onclick="return confirm('Sunteţi siguri că vreţi să ştergeţi acest record?')" class="btn btn-danger">Şterge</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
{% endblock content %}
{% include 'footer.html' %}

```

figura 70 Pagina view\_patient.html



```
{% include 'base.html' %}
{% include 'navigation.html' %}
{% block content %}
<div>
  <h2>Afişare Programări</h2>
  <table id="example" class="table table-hover">
    <thead>
      <tr>
        <th>Id-ul Programării</th>
        <th>Numele Doctorului</th>
        <th>Numele Pacientului</th>
        <th>Data</th>
        <th>Ora</th>
        <th>Şterge</th>
      </tr>
    </thead>
    <tbody>
      {% for i in apt %}
      <tr>
        <td>{{i.id}}</td>
        <td>{{i.doctor.name}}</td>
        <td>{{i.patient.name}}</td>
        <td>{{i.date}}</td>
        <td>{{i.time}}</td>
        <td><a href="{% url 'delete_appointment' i.id %}"
          onclick="return confirm('Sunteţi siguri că vreţi să ştergeţi acest record?')" class="btn btn-danger">Şterge</a>
        </td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
</div>
{% endblock content %}
{% include 'footer.html' %}
```

figura 71 Pagina view\_appointment.html

## Afişare Doctori

[Copy](#) [Excel](#) [CSV](#) [PDF](#)

Search:

Id-ul Doctorului	Numele	Telefonul Mobil	Specialitatea	Şterge
8	Maria Grigorescu	40701122334	Chirurg Consultant Senior	<a href="#">Şterge</a>
9	Alexandru Georgescu	40701122335	Chirurg Consultant Senior	<a href="#">Şterge</a>
10	Ana-Maria Axinte	40701122336	Chirurg Consultant Senior	<a href="#">Şterge</a>
11	Sebastian Stan	40701122337	Ginecolog si Obstetrician	<a href="#">Şterge</a>
12	Mihaela Enescu	40701122338	Chirurg Ortoped	<a href="#">Şterge</a>
13	Ioana Niculescu	40701122339	Medic consultant	<a href="#">Şterge</a>

Showing 1 to 6 of 6 entries

Previous [1](#) Next

figura 72 Afişarea paginii view\_doctor.html

## Afişare Pacienţi

[Copy](#) [Excel](#) [CSV](#) [PDF](#)

Search:

Id-ul Pacientului	Numele	Sexul	Telefonul Mobil	Adresa	Şterge
3	Alexandra Vasilescu	Feminin	40700000000	Bucuresti, str. Marceselor nr. 9	<a href="#">Şterge</a>
4	Nicutor Emanuel	Masculin	40722000033	Bucuresti, str. Izvoarelor nr. 9	<a href="#">Şterge</a>
5	Georgiana Daniela Alexe	Feminin	40701122399	Bucuresti, str. Caiselor nr. 256 bl. td9, ap. 110	<a href="#">Şterge</a>

Showing 1 to 3 of 3 entries

Previous [1](#) Next

figura 73 Afişarea paginii view\_patient.html

Afișare Programări					
<div>Copy Excel CSV PDF</div> <div>Search: <input type="text"/></div>					
Id-ul Programării	Numele Doctorului	Numele Pacientului	Data	Ora	Șterge
2	Maria Grigorescu	Alexandra Vasilescu	July 30, 2022	9:40 p.m.	<button>Șterge</button>
3	Mihaela Enescu	Georgiana Daniela Alexe	July 30, 2022	10 a.m.	<button>Șterge</button>

Showing 1 to 2 of 2 entries

Previous **1** Next

figura 74 Afișarea paginii view\_appointment.html

Voi discuta despre modul in care au fost create variabilele *doc*, *pat* si *apt* in capitolul *views.py*, iar despre bara de search si generarea de fisiere *Excel*, *CSV* si *PDF* in capitolul [3.5.10. base.html](#).

3.5.9. Pagina navigation.html

3.5.10. Pagina base.html

3.5.11. Pagina footer.html

In incheierea fiecărei pagini, mai includem inca o pagina “*footer.html*” care reprezinta blocul cu drepturile de autor.

```
{% include 'footer.html' %}
```

figura 75 Includerea paginii footer.html

```
<nav class="navbar navbar-expand-sm bg-dark navbar-dark fixed-bottom"></class>
  <div class="footer-bottom">
    <div class="container">
      <div class="col-sm-12 text-center ">
        <p class="text-white">CopyRight © 2022 Aplicație dezvoltată de Ana</p>
      </div>
    </div>
  </div>
</nav>
```

figura 76 footer.html

Efectul acestei pagini este prezentat mai jos.

CopyRight © 2022 Aplicație dezvoltată de Ana

figura 77 Efectul paginii footer.html

### 3.5.12. Pagina index.html

Aceasta pagina este un bord in care este prezentata situatia doctorilor, pacientilor si a programarilor. Este pagina pe care administratorul o vede prima data cand se autentifica in pagina aplicatie. Aceasta are o stilizare ca atunci cand trecem cu mouse-ul peste unul din blocurile pentru doctor, pacient si respective programari sa planeze. De asemenea, fiecare bloc are cate un link aferent, pentru doctori, acesta va trimite catre pagina de afisare doctori, pentru pacienti va trimite catre pagina afisare pacienti si pentru programari va trimite catre pagina afisare programari.

```

<style>
    .mydivstyle:hover {
        box-shadow: 0 0 20px 0 rgba(0, 0, 0, 0.3);
        transform: translateY(-20px);
    }

    hr.rounded {
        border-top: 10px solid lightblue;
        border-radius: 5px;
    }
</style>

{% block content %}
<h1 class="text-center">Bordul Administratorului</h1>
<br><br><br><br><br>

<div class="container">
    <div class="row">
        <div class="col-sm-4 mydivstyle">
            <h2 style="color: red;">{{ d }}</h2>
            <a href="{% url 'view_doctor' %}">
                <h4 style="color: blue">Doctori</h4>
                <hr class="rounded">
            </a>
        </div>
        <div class="col-sm-4 mydivstyle">
            <h2 style="color: red;">{{ p }}</h2>
            <a href="{% url 'view_patient' %}">
                <h4 style="color: blue">Pacienți</h4>
                <hr class="rounded">
            </a>
        </div>
        <div class="col-sm-4 mydivstyle">
            <h2 style="color: red;">{{ a }}</h2>
            <a href="{% url 'view_appointment' %}">
                <h4 style="color: blue">Programări</h4>
                <hr class="rounded">
            </a>
        </div>
    </div>
</div>

```

figura 78 Pagina index.html

## Bordul Administratorului



figura 79 Afisarea paginii index.html

Modul in care au fost implementate contoarele pentru doctori, pacienti si programari este descris in capitolul [3.6. Adaugarea view-urilor](#).

### 3.6. Adaugarea view-urilor

### 3.7. Adaugarea url-urilor

Adaugarea url-urilor este necesara deoarece aceasta ii va spune lui Django ca odata introdus un url in browser, acesta va stii ce pagina trebuie afisata. Important de mentionat aici este faptul ca trebuie importate toate view-urile folosite. Astfel, din fisierul *views.py* prezentat in capitolul anterior vom importa toate functiile create. Apoi vom adauga in lista *urlpatterns* url-ul pe care dorim ca utilizatorul sa il acceseze pentru a afisa pagina respectiva. De asemenea mai este de notat ca pentru fiecare intrare in lista avem si un nume, acest lucru usureaza munca de dezvoltare fiind mai usor sa facem referire la o anumita pagina printr-un nume reprezentativ.

```

from django.urls import path
from .views import About, Home, Contact, Login, Logout_admin, Index, View_Doctor, Delete_Doctor,

urlpatterns = [
    path('', Home, name='home'),
    path('about/', About, name='about'),
    path('contact/', Contact, name='contact'),
    path('admin_login/', Login, name='admin_login'),
    path('admin_logout/', Logout_admin, name='admin_logout'),
    path('index/', Index, name='dashboard'),
    path('view_doctor/', View_Doctor, name='view_doctor'),
    path('add_doctor/', Add_Doctor, name='add_doctor'),
    path('delete_doctor(?P<int:pid>)/', Delete_Doctor, name='delete_doctor'),
    path('view_patient/', View_Patient, name='view_patient'),
    path('delete_patient(?P<int:pid>)/', Delete_Patient, name='delete_patient'),
    path('add_patient/', Add_Patient, name='add_patient'),
    path('view_appointment/', View_Appointment, name='view_appointment'),
    path('add_appointment', Add_Appointment, name='add_appointment'),
    path('delete_appointment(?P<int:pid>)/', Delete_Appointment, name='delete_appointment'),
]

```

*figura 80 Adaugarea url-urilor*

Important de notat mai este si faptul ca pentru url-urile de “delete\_patient”, “delete\_doctor” si “delete\_appointment” avem acest “(?P<int:pid>)/”, acesta fiind necesar deoarece aceste url-uri contin metode de tip POST, adica cereri ale utilizatorului de a trimite informatii catre server. In cazul aplicatiei noastre, prin apasarea butonului de stergere, ii cerem server-ului sa stearga intrarea cu id-ul respectiv.

## 4. Lista figurilor

figura 1 Verificare instalare Python .....	2
figura 2 Verificare instalare versiune Django.....	3
figura 3 Crearea unui nou Proiect Django.....	3
figura 4 Listare structura proiect Django .....	4
figura 5 Fisierul manage.py .....	5
figura 6 Fisierul __init.py__.....	6
figura 7 Fisierul settings.py .....	7
figura 8 Fisierul urls.py .....	7
figura 9 Fisierul wsgi.py.....	8
figura 10 Fisierul asgi.py.....	9
figura 11 Crearea unei aplicatii Django noi.....	9
figura 12 Structura unei aplicatii Django .....	10
figura 13 Fisierul hospital.__init__.py.....	11
figura 14 includerea functiei "include" .....	14
figura 15 Adaugarea unui nou url in proiect.....	14
figura 16 Noua structura a proiectului.....	16
figura 17 Adaugarea unei aplicatii noi in aplicatiile instalate .....	17
figura 18 Setarea campului STATIC_URL.....	18
figura 19 Setarea campului STATICFILES_DIRS .....	18
figura 20 Fisierul mystyle.css .....	19
figura 21 Directorul images.....	20
figura 22 Executarea comenzii makemigrations.....	22
figura 23 Executarea comenzii migrate .....	22
figura 24 Crearea unui superuser .....	23
figura 25 Pornirea server-ului .....	23
figura 26 Accesarea aplicatiei .....	24
figura 27 URL-ul admin din urls.py din cadrul proiectului .....	25
figura 28 URL-ul admin din browser .....	25
figura 29 Pagina de administrator Django .....	26
figura 30 Diapozitivul de tip carusel.....	27
figura 31 Elementul Ingrijire postoperatorie .....	27
figura 32 Elementul Interventia chirurgicala .....	28
figura 33 Unitatea de terapie intensiva neonatala .....	28
figura 34 Butoanele Inainte si Inapoi .....	28
figura 35 Ingrijire postoperatorie .....	29
figura 36 Interventie Chirurgicala .....	29
figura 37 Unitatea de terapie intensiva neonatala .....	30
figura 38 Sectiunea de introducere .....	30
figura 39 Butoanele din sectiunea de introducere .....	31

figura 40	Sectiunea de introducere .....	31
figura 41	Butonul contact de urgenta.....	31
figura 42	Identificatorul departments .....	32
figura 43	Blocul de imagini al departamentelor .....	32
figura 44	Sectiunea servicii medicale.....	32
figura 45	Sectiunea alte servicii .....	33
figura 46	Identificatorul doctors .....	33
figura 47	Blocul doctorilor .....	33
figura 48	Sectiunea doctorilor .....	34
figura 49	about.html .....	34
figura 50	Pagina despre noi .....	35
figura 51	Efectul de planare al blocurilor din contact.html .....	35
figura 52	Pagina contact.html .....	36
figura 53	Clauzele "if" din pagina login.html .....	37
figura 54	Autentificare cu succes.....	37
figura 55	Autentificare invalida.....	38
figura 56	Campurile pentru utilizator si parola.....	38
figura 57	Pagina de autentificare.....	38
figura 58	Modelul Doctor.....	39
figura 59	Migrarea modelelor .....	40
figura 60	Modelul Patient .....	40
figura 61	Modelul programarilor .....	41
figura 62	Clauzele if pentru adaugarea doctorului .....	42
figura 63	Campurile pentru adaugarea unui doctor .....	42
figura 64	Pagina add_doctor.html .....	43
figura 65	Pagina add_patient.html .....	44
figura 66	Afisarea paginii add_patient.html .....	44
figura 67	Pagina add_appointment.html.....	46
figura 68	Afisarea paginii add_appointment.html.....	47
figura 69	Pagina view_doctor.html.....	48
figura 70	Pagina view_patient.html.....	48
figura 71	Pagina view_appointment.html .....	49
figura 72	Afisarea paginii view_doctor.html.....	49
figura 73	Afisarea paginii view_patient.html.....	49
figura 74	Afisarea paginii view_appointment.html .....	50
figura 75	Includerea paginii footer.html.....	50
figura 76	footer.html.....	50
figura 77	Efectul paginii footer.html.....	50
figura 78	Pagina index.html .....	52
figura 79	Afisarea paginii index.html .....	53
figura 80	Adaugarea url-urilor .....	54



## 5. Referinte

<https://docs.djangoproject.com/en/4.0/>

<https://ro.wikipedia.org/wiki/Python>

<https://packagecontrol.io/>

<https://techvidvan.com/tutorials/django-project-structure-layout/>

<https://docs.djangoproject.com/en/4.0/ref/contrib/staticfiles/#module-django.contrib.staticfiles>

[https://docs.djangoproject.com/en/4.0/ref/settings/#std:setting-STATIC URL](https://docs.djangoproject.com/en/4.0/ref/settings/#std:setting-STATIC_URL)

<https://www.flaticon.com/>

<https://docs.djangoproject.com/en/4.0/topics/migrations/>

<https://docs.djangoproject.com/en/1.8/intro/tutorial02/>

## 6. Cuprins

1.	Descrierea aplicatiei.....	1
2.	Dezvoltarea aplicatiei.....	2
2.1.	Setarea mediului de lucru .....	2
2.2.	Crearea unui proiect Django nou.....	3
2.2.3.	Fisierul manage.py .....	5
2.2.4.	Fisierul __init.py__ .....	5
2.2.5.	Fisierul settings.py .....	6
2.2.6.	Fisierul urls.py .....	7
2.2.7.	Fisierul wsgi.py .....	8
2.2.8.	Fisierul asgi.py .....	8
2.3.	Crearea unei aplicatii Django noi .....	9
2.3.1	Fisierul hospital.__init__.py .....	11
2.3.2.	Fisierul admin.py .....	11
2.3.3.	Fisierul apps.py .....	12
2.3.4.	Fisierul models.py .....	12
2.3.5.	Fisierul views.py .....	12
2.3.6.	Fisierul urls.py .....	13
2.3.7.	Fisierul test.py .....	13
3.	Dezvoltarea aplicatiei.....	14
3.1.	Adaugarea url-ului aplicatiei .....	14
3.2.	Fisierele statice .....	15
3.2.1.	Campul STATIC_URL.....	17
3.2.2.	Campul STATICFILES_DIRS .....	18
3.2.3.	STATIC_ROOT .....	19
3.2.4.	Directorul static.....	19
3.3.	Migrarile si crearea de superusers.....	21
3.4.	Pagina initiala Django de administrare .....	23
3.5.	Directorul templates .....	26
3.5.1.	Fisierul home.html .....	26

3.5.2.	Pagina about.html .....	34
3.5.3.	Pagina contact.html .....	35
3.5.4.	Pagina login.html .....	36
3.5.5.	Pagina add_docotor.html .....	38
3.5.5.1.	Modelul doctorilor .....	39
3.5.5.2.	Modelul pacientilor.....	40
3.5.5.3.	Modelul programarilor.....	41
3.5.6.	Pagina add_patient.html.....	43
3.5.7.	Pagina add_appointment.html .....	44
3.5.8.	Paginile view_doctor.html, view_patient.html si view_appointment.html .....	47
3.5.9.	Pagina navigation.html .....	50
3.5.10.	Pagina base.html .....	50
3.5.11.	Pagina footer.html .....	50
3.5.12.	Pagina index.html.....	51
3.6.	Adaugarea view-urilor.....	53
3.7.	Adaugarea url-urilor.....	53
5.	Referinte .....	57
6.	Cuprins .....	58