Aplicatie de manageriere a unui spital

# 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 spitaul, optiuniile de *inregistrare, autentifacare* 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.

# Dezvoltarea aplicatiei

* 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](https://www.sublimetext.com/download) 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 fereastra 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.

Text

Description automatically generated

figura Verificare instalare Python

Daca mesajul este unul de eroare atunci Python se poate instala de pe site-ul oficial al acestuia accesand acest [link](https://www.python.org/downloads/).

Un ultim pas mai este necesar inainte de a putea incepea 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.

Text

Description automatically generated with medium confidence

figura Verificare instalare versiune Django

## 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”.*

*Text

Description automatically generated*

figura Crearea unui nou Proiect Django

Se poate observa ca in directorul in care am proiectul s-a creat un director nou cu numele proeictului 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.

Text

Description automatically generated

figura 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, salvarimigrari dar si multe altele.

Text

Description automatically generated

figura 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 acelui director este un pachet si asadar prezenta lui face ca directorul in care se afla sa fie tratat ca un proiect.

Graphical user interface, text, application

Description automatically generated

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

### 2.2.5. Fisierul settings.py

Acesta este cel mai important fisier, deorece 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.

Text

Description automatically generated

figura 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.

A screenshot of a computer

Description automatically generated with medium confidence

figura 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.

A screenshot of a computer

Description automatically generated

figura 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.*

A screenshot of a computer

Description automatically generated

figura Fisierul asgi.py

## 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 mail 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.

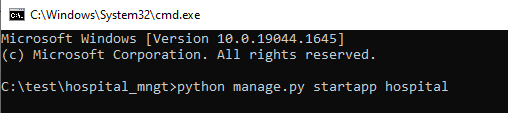


figura 11 Crearea unei aplicatii Django noi

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 creeri 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.

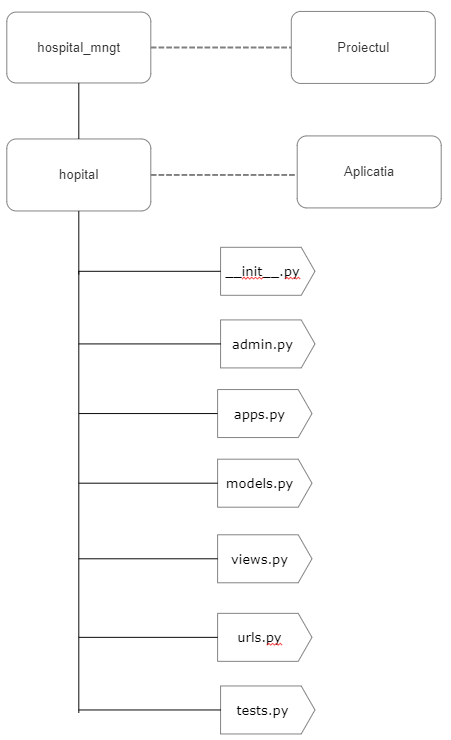


figura 12 Structura unei aplicatii Django

## 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 aceste este un fisier gol. Singurul rol al acestui fisier este acela de a-I spune interpretului de Python ca acel director trebuie tratat ca un proiect.

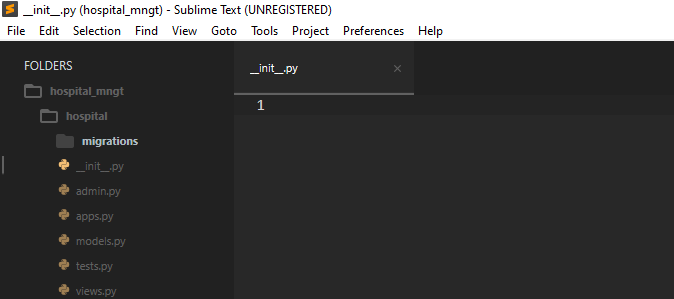


figura 13 Fisierul hospital.\_\_init\_\_.py

### 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 strctura 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 asemantor cu fisierul *urls.py* discutat anterior in capitolul [*2.2.6. Fisierul urls.py*](#_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.

1. Dezvoltarea 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”.

Graphical user interface, text

Description automatically generated

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

A screenshot of a computer

Description automatically generated with medium confidence

In cazul nostru, cand utilizatorul acceseaza url-ul aplicatiei noastre, acesta va fi redirectionat catre url-urile din interiorul aplcatiei create de noi, pe care le vom implementa ulterior.

Urmatorul pas este sa cream doua noi directoare in interiorul aplicatiei noastre. Acestea sunt directorul *static* care va contine toate fisierele statice precum imaginile pe care le vom afisa in aplicatia noastra web si fisierele ce tin de *css,* adica limbajul pe care il vom folosi pentru a stiliza documentele *HTML,* si inca un director numit *templates* unde vom crea toate fisierele *HTML,* practice, in acest director vom crea paginile pe care dorim sa le afisam in aplicatia noastra web. Directorul *static* este la randul sau compus din alte doua directoare, unul *css* care va contine fisierele de stilizare discutate anterior, si unul *images* care va contine imaginile aplicatiei noastre. La sfarsit noua structura a aplicatiei noastre va arata ca in figura de mai jos

1. Lista figurilor

[figura 1 Verificare instalare Python 2](#_Toc101453988)

[figura 2Verificare instalare versiune Django 3](#_Toc101453989)

[figura 3 Crearea unui nou Proiect Django 3](#_Toc101453990)

[figura 4 Listare structura proiect Django 4](#_Toc101453991)

[figura 5 Fisierul manage.py 5](#_Toc101453992)

[figura 6 Fisierul \_\_init.py\_\_ 6](#_Toc101453993)

[figura 7 Fisierul settings.py 7](#_Toc101453994)

[figura 8 Fisierul urls.py 7](#_Toc101453995)

[figura 9 Fisierul wsgi.py 8](#_Toc101453996)

[figura 10 Fisierul asgi.py 9](#_Toc101453997)

[figura 11 Crearea unei aplicatii Django noi 9](#_Toc101453998)

[figura 12 Structura unei aplicatii Django 10](#_Toc101453999)

[figura 13 Fisierul hospital.\_\_init\_\_.py 11](#_Toc101454000)

# 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/>

# Cuprins

[1. Descrierea aplicatiei 1](#_Toc102050030)

[2. Dezvoltarea aplicatiei 2](#_Toc102050031)

[2.1. Setarea mediului de lucru 2](#_Toc102050032)

[2.2. Crearea unui proiect Django nou 3](#_Toc102050033)

[2.2.3. Fisierul manage.py 5](#_Toc102050034)

[2.2.4. Fisierul \_\_init.py\_\_ 5](#_Toc102050035)

[2.2.5. Fisierul settings.py 6](#_Toc102050036)

[2.2.6. Fisierul urls.py 7](#_Toc102050037)

[2.2.7. Fisierul wsgi.py 8](#_Toc102050038)

[2.2.8. Fisierul asgi.py 8](#_Toc102050039)

[2.3. Crearea unei aplicatii Django noi 9](#_Toc102050040)

[2.3.1 Fisierul hospital.\_\_init\_\_.py 11](#_Toc102050041)

[2.3.2. Fisierul admin.py 11](#_Toc102050042)

[2.3.3. Fisierul apps.py 12](#_Toc102050043)

[2.3.4. Fisierul models.py 12](#_Toc102050044)

[2.3.5. Fisierul views.py 12](#_Toc102050045)

[2.3.6. Fisierul urls.py 13](#_Toc102050046)

[2.3.7. Fisierul test.py 13](#_Toc102050047)

[4. Referinte 15](#_Toc102050048)

[5. Cuprins 16](#_Toc102050049)