**UNIVERSITATEA DIN PITEȘTI**

**FACULTATEA DE ȘTIINȚE,**

**EDUCAȚIE FIZICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**Coordonator științific**

**Conf. Univ. Dr. Doru Anastasiu Popescu**

**Absolvent**

**Cîrstea Florin Daniel**

**PITEȘTI**

**2018**

**UNIVERSITATEA DIN PITEȘTI**

**FACULTATEA DE ȘTIINȚE,**

**EDUCAȚIE FIZICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

**APLICAȚIE WEB**

**PENTRU E-LEARNING**

**Coordonator științific**

**Conf. Univ. Dr. Doru Anastasiu Popescu**

**Absolvent**

**Cîrstea Florin Daniel**

**PITEȘTI**

**2018**

# **CUPRINS**

[**INTRODUCERE 4**](#_Toc505016106)

# **CAPITOLUL 1. FUNDAMENTE ÎN DEZVOLTAREA APLICAȚIILOR WEB CU PYTHON………………………………………………………………..5**

* 1. **Bazele programării web………………………..……………...…….5**
     1. Istoria web-ului...................................................................5
     2. Serverul web.......................................................................6
  2. **Limbajul de programare Python……………...…………………….6**

**1.3 Framework-ul Django………….....…………………………………7**

* + 1. Ce este Django ?.................................................................7
    2. Procesul request-response….…………………………….8
    3. Arhitectura Django...……………....…………………....10

**CAPITOLUL 2. TEHNOLOGII FOLOSITE ÎN DEZVOLTAREA APLICAȚIEI..........................................................................................................12**

**2.1 Tehnologii Back-end.........................................................................12**

2.1.1 MySQL............................................................................12

**CAPITOLUL 3. PROIECTAREA APLICAȚIEI...............................................15**

**3.1 Configurarea mediului de lucru.......................................................15**

2.1.1 Crearea mediului virtual...................................................15

**Introducere**

Odată cu progresul în domeniul tehnologic, informatizarea companiilor și instituțiilor a avut o creștere semnificativă. Acest lucru a schimbat lumea, felul în care comunicăm și modul în care ne desfășurăm activitățile în viața de zi cu zi. S-au deschis foarte multe oportunități de dezvoltare, dar a apărut și nevoia constantă de simplificare a anumitor procese, prin crearea de aplicații și site-uri web.

Prin intermediul acestei lucrări de licență, intitulată ”Aplicație web pentru e-learning”, mi-am propus crearea unei platforme ce conține informații despre cursurile profesorilor și situațiile studenților din cadul unei instituții superioare de învățământ. Prin devoltarea acestei aplicații, personalul unei universități va avea posibilitatea introducerii datelor despre facultăți, departament, programe de studii, cursuri și studenți. Profesorii vor putea să-și adauge cursurile si lecțiile respective, dar și să noteze studenții și să le permită accesul la toate aceste informații. Astfel, aplicația reprezintă o metodă eficientă și ușoară în managementul cursurilor.

Aplicația este eficientă și securizată, datorită limbajului de programare folosit în dezvoltarea acesteia, și anume Python. Acest limbaj a avansat rapid în ultimii ani, fiind un limbaj dinamic, securizat și relativ ușor de folosit și implementat. Lucrul acesta reprezintă un mare plus, Python fiind folosit foarte larg atât în dezvoltarea aplicațiilor web, cât și în machine learning și data science.

În devoltarea aplicației web, am ales să folosesc Django, ce reprezintă un framework pentru Python și permite crearea de site-uri web foarte securizate, într-un timp cat mai scurt. Motivul pentru care am ales această metodă a fost învățarea și stăpânirea unui nou limbaj de programare, un limbaj foarte interesant și ce se află in dezvoltare continuă.

**Capitolul 1. Fundamente în dezvoltarea aplicațiilor web cu Python**

* 1. **Bazele programării web**
     1. **Istoria web-ului**

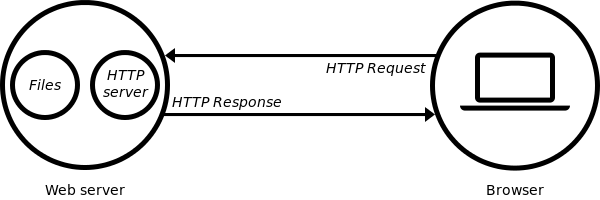
Web-ul ne ține conectați pe toți la o scală globală și face parte din viața noastră de zi cu zi. Începând din anul 1991, când prima pagină web a fost lansată de către Berners Lee și până în ziua de azi, au avut loc mari progrese în domeniu. De asemenea, acesta a inventat și două protocoluri web și anume HTTP și HTML dar și primul browser, numit „the World Wide Web”. În anii ce au urmat, au apărut și alte browsere ca: Line Mode Browser (1992), Mosaic Browser (1993), Opera 1.10 (1995) și Internet Explorer (1995).

Începând cu anul 1995, Javascript și CSS au fost introduse. Javascript permitea site-urilor web să gândescă, în timp ce CSS permitea modificarea layout-ului acestora (culori, margini, fonturi etc.). Totuși, CSS nu era compatibil la acea vreme cu browserele existente. Acest lucru a durat până în anul 2000, când a fost lansat Internet Explorer 5 , ce putea procesa CSS-ul în proporție de 99%.

Pe lângă HTML, CSS și Javascript, au apărut mai târziu mult mai multe tehnologii ce au schimbat total web-ul.

* + 1. **Serverul web**

Un server web reprezintă un computer ce stochează componentele unui site web. Serverul propriu-zis are în componența sa un server HTTP ce înțelege adresele web (URL-urile) și protocolul HTTP (prin care browser-ul afișează paginile).



Când browser-ul încearcă să afișeze o pagină web, are loc un request către pagina respectivă prin protocolul HTTP. Când requestul ajunge la serverul web, serverul HTTP acceptă requestul și trimite un răspuns cu pagina înapoi către browser. Dacă pagina nu este gasită, serverul va trimite un răspuns cu eroarea 404 (pagina nu se găsește pe server).

* 1. **Limbajul de programare Python**

Python este un limbaj de programare de nivel înalt și dinamic, folosit în dezvoltarea unei mari varietăți de aplicații. Acesta a fost creat de către Guido van Rossum și lansat în 1991, punând accent pe lizibilitate și o sintaxă ce permite exprimarea anumitor concepte, în linii mai puține de cod față de alte limbaje.

Limbajul de programare a fost făcut astfel încât să fie foarte extensibil, decât să aibe toate funcționalitățile construite înăuntrul său. Lucrul acesta l-a foarte popular pentru că permite extensia aplicațiilor prin adăugare de interfețe programabile. Python este de asemenea și un limbaj interpretat, ceea ce înseamnă că limbajul nu este compilat înaintea execuției, ci în timpul acesteia.

În ziua de azi, acesta are foarte multe întrebuințări. Poate fi folosit la procesarea imaginilor și a datelor, machine learning, aplicații web, aplicații desktop și cam la tot ceea ce poți pune într-un computer. Foarte des este comparat cu Ruby, deoarce împart o sintaxă asemănătoare și sunt amândouă limbaje interpretate. Totuși, datorită lizabilității sale, Python este mai folosit în domeniul web. Comparat cu alte limbaje ca Java sau PHP, acesta este un limbaj orientat pe obiect și nu numai, ce excelează prin proprietatea sa de a face mult mai mult cu linii de cod mai puține.

Fiind ușor de citit și înțeles de majoritatea programatorilor, acesta oferă o dezvoltare mult mai rapidă a aplicațiilor.

* 1. **Framework-ul Django**
     1. **Ce este Django ?**

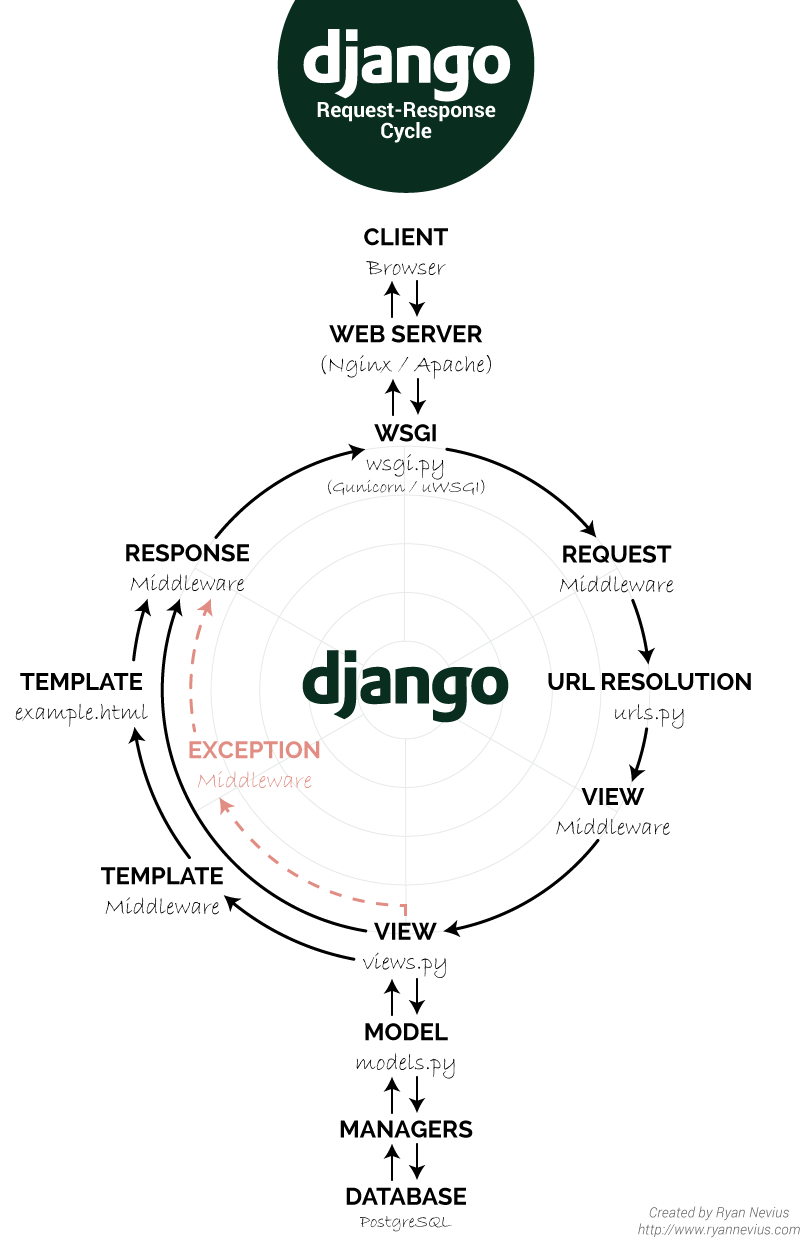
Când vorbim de crearea unei pagini web avem nevoie de ajutorul unui framework, în majoritatea cazurilor. Un framework reprezintă o infrastructură necesară pentru a programa rapid aplicații web fără a fi necesar să scriem codul de la zero. În cazul de față, Python oferă o bună varietate de framework-uri, printre care se numără: Pyramid, Tornado, Web2py și bineînțeles Django.

Django reprezintă de fapt o colecție open-source de librării Python, de nivel înalt, în stil **MVC** (Model View Controller)**,** ce pune accent pe eficiență. Este denumit de altfel, ca și un framework pentru „profesioniști cu termene limită”. Principul de bază al acestui framework este ”Don’t repeat yourself” (DRY), adică să nu te repeți. Ceea ce l-a făcut popular însă, este **modularitatea** sa ce permite dezvoltatorului să aleagă funcționalitățile pe care le dorește incluse și **transparența**, ceea ce îl face mai bun decât anumite framework-uri.

Caracteristicile sale majore sunt:

1. Este foarte rapid, a fost construit astfel încât aplicațiile să fie dezvoltate într-un timp cât mai scurt.
2. Are „Baterii incluse”, Django având grijă în același timp de autentificare, conținut, hărți, feed-uri RSS și multe altele.
3. Django este foarte securizat, mai ales împotriva injecțiilor SQL, cross-site scripting, cross-site request forgery și clickjacking.
4. Este scalabil, folosind cât mai puține resurse hardware.
5. Framework-ul este de asemenea și incredibil de versatil, fiind folosit de multe companii în diferite tipuri de aplicații.
   * 1. **Procesul request-response**

Primul lucru după crearea unui proiect în Django este construirea unui URL (urls.py) unde vederile vor fi afișate. De asemenea, trebuie create și vederile ce vor afișa template-urile respective pe pagina web. Acest lucru se realizează printr-un **request** și un **response** (Figura 1.2.2.1).

****

*Figura 1.2.2.1 Ciclul request – response.*

Pentru conectarea unui server web (Apache, Nginx) la framework, Django se folosește de **WSGI** (wsgi.py). Acesta este un tool prin care serverul execută aplicația, apoi aplicația procesează request-ul, după care aceasta oferă serverului înapoi un răspuns. În Django, există și **middleware**-uri care oferă anumite funcționalități ce acționează ca niște filtre pentru datele schimbate între server și aplicație. Un exemplu de middleware sunt cele de tip **CSRF**, ce sunt folosite în formulare și împiedică atacurile cross-site request forgery.

Informațiile din modele (models.py) sunt transformate în limbaj SQL și pot fi furnizate prin vedere.

Vederile (views.py) sunt de fapt funcții ce trebuie să îndeplinescă 3 condiții și anume: să poată fi accesate în urls.py, trebuie să accepte un HttpRequest ca și parametru, și trebuie să returneze un obiect de tip HttpResponse. Astfel, handler-ul WSGI va știi ce funcție să cheme și va trimite datele către serverul web, realizându-se un proces request-response.

* + 1. **Arhitectura Django**

Toți dezvoltatorii de aplicații web cunosc conceptul de MVC (Model View Controller), fiind una din caracteristile de bază a celor mai bune framework-uri de programare web. De asemenea, Django se numără printre ele, doar că în cazul acesta MVC-ul este puțin diferit. Anumiți programatori consideră că de fapt Django folosește conceptul de MTV (Model Template View).

Un MVC este alcătuit din: **model (M)** ce reprezintă modelul datelor, acesta oferă și un layer abstract ce permite utilizarea aceluiași model cu baze de date diferite, **view (V)** ce este o reprezentare a datelor și în principiu ceea ce browser-ul afișează și **controller (C)** ce este folosit în prelucrarea datelor ce vor fi afișate, acesta controlând fluxul de informații dintre model și view.

Django folosește propria logică în implementarea MVC-ului, unde controller-ul este înlocuit de către framework însuși. MVT-ul aduce noi concepte pe lângă model și anume conceptele de: **template (T)** ce conține decizii în felul în care anumite lucruri sunt afișate în pagină și **view (V)** care reprezintă un „pod” între model și template-uri, asigurând prelucrarea corectă a datelor.

**Capitolul 2. Tehnologii folosite în dezvoltarea aplicației**

* 1. **Tehnologii Back-end**

Partea de Back-end sau server-side a unei aplicații reprezintă modul în care aceasta funcționează. În mare parte Back-end-ul constă în baze de date și servere, lucruri ce nu pot fi văzute de utilizatorii normali ai unui website.

Tehnologiile de tip Back-end sunt folosite în crearea de site-uri dinamice, adică site-uri ale căror date sunt alterate în timp real. De asemenea un site dinamic are nevoie de o bază de date în care informațiile gen imagini, documente sau profiluri de utilizatori vor fi stocate.

* + 1. **MySQL**

O bază de date reprezintă o colecție de date organizate. Când vine vorba de aplicație, avem nevoie de o astfel de bază de date pentru stocarea informațiilor.

Limbajul folosit într-o bază de date în majoritatea cazurilor este SQL (Structured Query Language) ce permite manipularea acesteia. Acesta a fost dezvoltat la începutul anilor ’70 de către Donald Chamberlin și Raymond Boyce la IBM, prima versiune numidu-se simplu *Sequel*. SQL este definit de către standardul ANSI/ISO încă din anul 1986. De atunci, standardul a fost tot schimbat, dar SQL nu a devenit niciodată portabil în totalitate, fiind necesară aducerea de modificări în bazele de date folosite. Limbajul este în mare parte alcătuit din următoarele elemente:

1. Clauze, ce reprezintă componentele unei instrucțiuni.
2. Predicate, ce specifică condițiile ce pot fi evaluate.
3. Expresii, ce produc valori sau tabele de date.
4. Interogări, ce colecteză datele cerute din baza de date.
5. Instrucțiuni, ce produc un efect asupra datelor.

Pentru a lucra cu o bază de date, se folosesc diferite sisteme de management a bazelor de date ca: Oracle, PostgreSQL, SQLite și MySQL, care este cel mai popular și open-source.



*Figura 2.1.1.1 MySQL*

Bazele de date MySQL sunt relaționale, asta însemnând că datele sunt stocate în tabele separate. De asemenea utilizatorul poate decide relațiile între câmpurile de date (one-to-one, one-to-many, many-to-many etc.). MySQL este și open-source ceea ce permite folosirea gratuită de către oricine a resurselor oferite de acesta.

Am ales să folosesc MySQL în dezvoltarea aplicației datorită faptului că este foarte eficient și integrabil pe orice platformă, ușor de folosit și securizat. Acesta consumă resurse puține și rulează foarte bine în background cu alte aplicații, acest lucru făcându-l o opțiune perfectă pentru stocarea datelor aplicației. Instalarea serverului MySQL pe mașina locală, se realizează ușor prin aplicația MySQL Installer, iar prin crearea unui utilizator și a unei baze de date, MySQL este gata de folosit în Back-end-ul aplicației web. Acest proces este trivial, fiind necesar doar terminalul (sau consola) sistemului de operare.

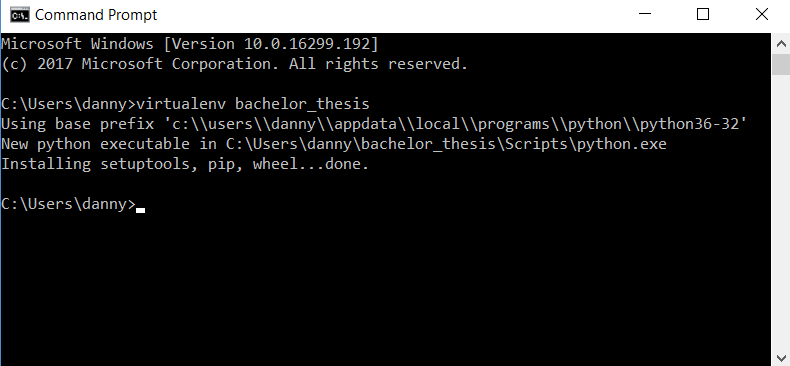
Prin utilizarea MySQL ca și sistem de management al datelor, am fost capabil să mențin datele într-un singur loc și să le accesez foarte ușor și fără probleme în procesul proiectării aplicației.

**Capitolul 3. Proiectarea aplicației**

* 1. **Configurarea mediului de lucru**
     1. **Crearea mediului virtual**

În dezvoltarea aplicațiilor cu Django, un mediu virtual este esențial în păstrarea securizată a proiectului și a tuturor datelor din componența sa. Acest lucru ajută foarte mult și la lansarea aplicației pe o platformă, fiind considerat o practică bună.

Pe parcursul dezvoltării acestei aplicații, am folosit Django 2.0.1 și Python 3.6.4, ceea ce înseamnă că **pip** va avea versiunea 9.0.1. Pip este un pachet în Python, folosit la instalarea sau dezinstalarea altor pachete. În continuare am folosit pip pentru a instala mediul virtual al aplicației. Pentru crearea unui mediu virtual totul se realizează printr-o singură comandă:



*Figura 3.1.1.1 Crearea mediului virtual*

**BIBLIOGRAFIE**

1. <https://www.thoughtco.com/what-is-python-2813564>
2. <https://www.upwork.com/hiring/development/django-programming>
3. <https://medium.com/zeitcode/django-middlewares-and-the-request-response-cycle-fcbf8efb903f>
4. <https://www.djangoproject.com>
5. [https://developer.mozilla.org/enUS/docs/Learn/Common\_questions/What\_is\_a\_web\_server](https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server)
6. <https://djangobook.com/model-view-controller-design-pattern/>
7. <https://en.wikipedia.org>
8. <http://ryannevius.com>
9. <https://dev.mysql.com/doc/refman/5.7/en/>