UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Aplicație web pentru alegerea colegilor de cameră la cămin

propusă de

Silviu-Constantin Drăgan

Sesiunea: *iulie, 2018*

Coordonator științific

Lect.Dr. Cristian Frăsinaru

UNIVERSITATEA "ALEXANDRU IOAN CUZA" DIN IAȘI

FACULTATEA DE INFORMATICĂ

Aplicație web pentru alegerea colegilor de cameră la cămin

Silviu-Constantin Drăgan

Sesiunea: *iulie, 2018*

Coordonator științific

Lect.Dr. Cristian Frăsinaru

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele _____

Data _____ Semnătura _____

DECLARAŢIE privind originalitatea conţinutului lucrării de licenţă

Subsemntatul(a) ……

domiciliul în ……..

născut(ă) la data de ……., identificat prin CNP ……., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza" din Iaşi, Facultatea de …….. specializarea ……., promoţia ……., declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: _____

_____

_____elaborată sub îndrumarea dl. / d-na _____, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, …… Semnătură student ……

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Titlul complet al lucrării*", codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași,

Absolvent *Drăgan Silviu-Constantin*

_____

(semnătura în original)

Cuprins

INTRODUCERE **6**

I. DESCRIEREA APLICAȚIEI **8**

I.1. Modul User **8**

I.1.1. Logare 8

I.1.2. Recuperare parolă 8

I.1.3. Prezentare funcționalități 8

I.2. Modul administrator **11**

II. TEHNOLOGII FOLOSITE **13**

II.1. Python **13**

II.2. Django framework **14**

II.2.1. Introducere 14

II.2.2. Securitate 14

II.2.3. ORM 14

II.2.4. Templetizare 16

II.2.5. Crearea unui proiect 17

II.3. HTML, CSS & JavaScript **18**

II.4. MySQL **19**

III. ARHITECTURA APLICAȚIEI **20**

III.1. MVC **20**

III.1.1. Model 20

III.1.2. View 23

III.1.3. Controller 24

III.2. MTV **25**

IV. ALGORITMUL STABLE MATCHING ROOMMATES **27**

V. IMPLEMENTARE **31**

VI. MANUAL DE UTILIZARE **38**

CONCLUZII **39**

BIBLIOGRAFIE **40**

Introducere

În zilele noastre se pune accentul din ce în ce mai mult pe automatizarea lucrurilor ce sunt repetitive cu scopul de a ușura munca utilizatorilor prin intermediul tehnologiei. Alegerea colegului sau colegilor de cameră este un lucru esențial pentru studenții ce optează pentru cazarea la unul din căminele universității întrucât aceștia vor fi nevoiți să împartă o mare parte din spațiul lor personal cu alți studenți timp de un an universitar, deci se dorește ca alegerea acestor persoane să fie una cât mai potrivită.

În momentul de față, alegerea colegilor de cameră este posibilă atunci când studentul, după ce află că a primit un loc la cămin, vine la facultate pentru a lua dispoziția de cazare împreună cu colegii cu care a căzut de acord să împartă aceeași cameră și completează același tabel ce reprezintă locurile disponibile dintr-o cameră. Acest proces este unul destul de anevoios deoarece, spre exemplu, dacă 5 studenți doresc să stea în aceeași cameră vor trebui să fie printre primii care reușesc să își ia dispoziția de cazare pentru a putea să găsească o cameră cu toate cele 5 locuri vacante. În plus, cum acest lucru se întâmplă înainte de începerea anului universitar este și mai dificilă întâlnirea acestor grupuri când fiecare persoană trebuie să vină dintr-o anumită regiune a țării. Rezumând cele spuse anterior dezavantajele folosirii sistemului deja existent sunt următoarele:

- Necesitatea de a fi cât mai devreme la facultate astfel încât să fii sigur că ai colegii pe care îi dorești; bineînțeles că anumite persoane care nu vor putea ajunge de dimineață din diverse motive vor alege din camerele rămase
- În cazul în care un student nu știe pe nimeni el nu are o modalitate prin care să poată afla anumite informații despre celelalte persoane, prin urmare va fi în cameră cu niște persoane de care nu știe nimic și este șansa să existe divergențe
- Necesitatea de a completa un formular fizic, astfel pe lângă faptul că se pot deteriora foarte rapid și foarte ușor hârtiile, acestea se pot completa greșit din neatenție

Astfel pentru a elimina aceste dezavantaje am decis crearea unei aplicații web care să ușureze atât munca studenților, cât și a personalului facultăților.

Utilizarea acestei aplicații ar avea următorul impact asupra studenților:

- Aceștia, pe baza unui cont creat cu ajutorul numărului matricol, vor putea să selecteze preferințele pentru colegii de cameră, de la opțiunea cea mai bună pentru el până la opțiunea cea mai rea
- Există posibilitatea ca un student nou-venit să nu cunoască pe nimeni, în acest caz el poate vedea recenziile ce au fost făcute după ce studenții au stat în aceeași cameră și astfel să găsească colegii potriviți
- Toate informațiile vor fi stocate într-o bază de date, nu va mai fi astfel nevoie de completarea unor hârtii în format fizic, deci asta ar putea însemna și nevoia de un personal mai mic din partea administrației facultății atunci când vine vorba de repartizarea studenților la cămin

Problema repartizării studenților în funcție de preferințele acestora va fi rezolvată cu ajutorul algoritmului de stable matching roommates, extins astfel încât să ofere o soluție indiferent dacă camera are 2, 3, 4 sau 5 locuri disponibile.

În cele ce urmează voi prezenta aplicația web în cadrul următoarelor capitole:

1) Descrierea aplicației – vor fi prezentate caracteristicile aplicației, atât din perspectiva studentului cât și din perspectiva administratorului
2) Tehnologii utilizate – vor fi prezentate tehnologiile folosite în vederea realizării acestei aplicații web
3) Arhitectura aplicației – detalii tehnice despre cum funcționează aplicația
4) Algoritmul stable matching roommates – explicarea algoritmului printr-un exemplu și prin prezentarea codului sursă
5) Implementare – explicarea anumitor funcționalități și prezentarea anumitor secvențe relevante de cod
6) Manual de utilizare – configurarea ce trebuie făcută pentru a putea rula pe o mașină aplicația

I. Descrierea aplicației

I.1. Modul User

I.1.1. Logare

Când studentul se loghează pentru prima dată va trebui să acceseze „Resetare parolă” și să urmeze instrucțiunile de acolo întrucât acesta are un cont deja creat de administrator, dar va trebui acum să își seteze o parolă. După ce parola a fost aleasă userul poate să se logheze introducând credențialele precum în figura de mai jos.

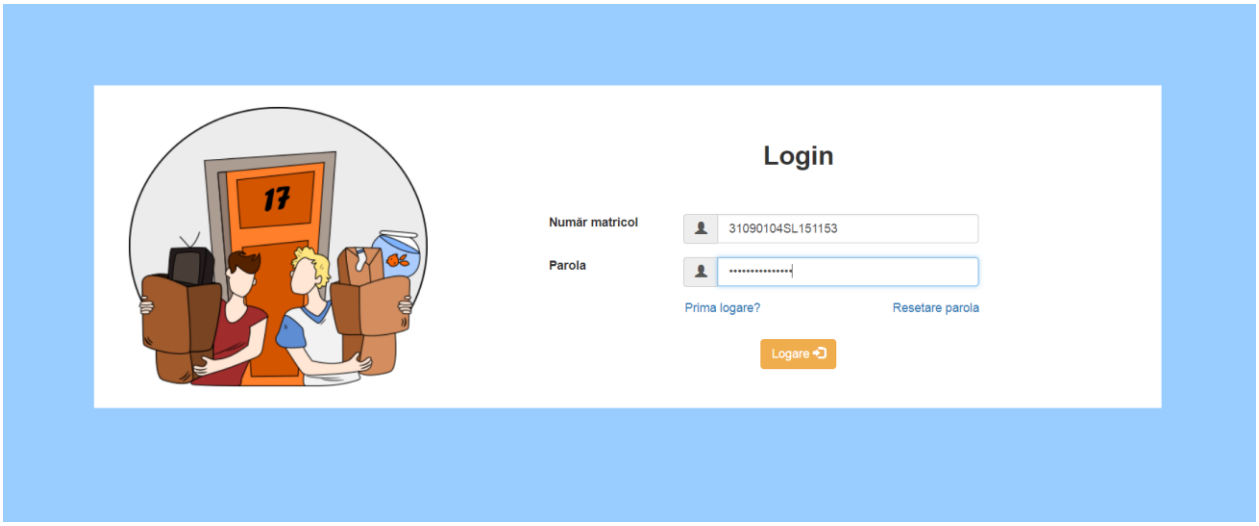


Figura I.1

I.1.2. Recuperare parolă

În cazul în care studentul și-a uitat parola sau vrea să o schimbe poate opta pentru opțiunea „Resetare parolă” unde va trebui să introducă adresa de email primită de la facultate, apoi va fi redirecționat către formular unde va introduce noua parolă și codul de securitate pe care l-a primit prin email.

I.1.3. Prezentare funcționalități

După ce s-a logat studentul este redirecționat către pagina principală unde se află detalii despre căminele Universității Alexandru Ioan Cuza, având posibilitatea astfel de a vedea unde sunt amplasate, ce dotări au, care este capacitatea camerelor și la ce preț.

Din bara de meniu are posibilitatea de a alege să își vizualizeze profilul, loc în care va alege și preferințele pentru colegii de cameră, să vadă recenziile făcute de studenții din anii precedenți sau să părăsească pagina prin comanda de deconectare.

Figura I.2

Pe pagina de profil a utilizatorului acesta are în partea stângă câteva informații despre el, numele, numărul matricol, grupa și anul, după cum se poate observa în Figura 3. De asemenea, acesta își poate adăuga o poză și este chiar recomandat pentru a putea fi recunoscut cu o mai mare ușurință.

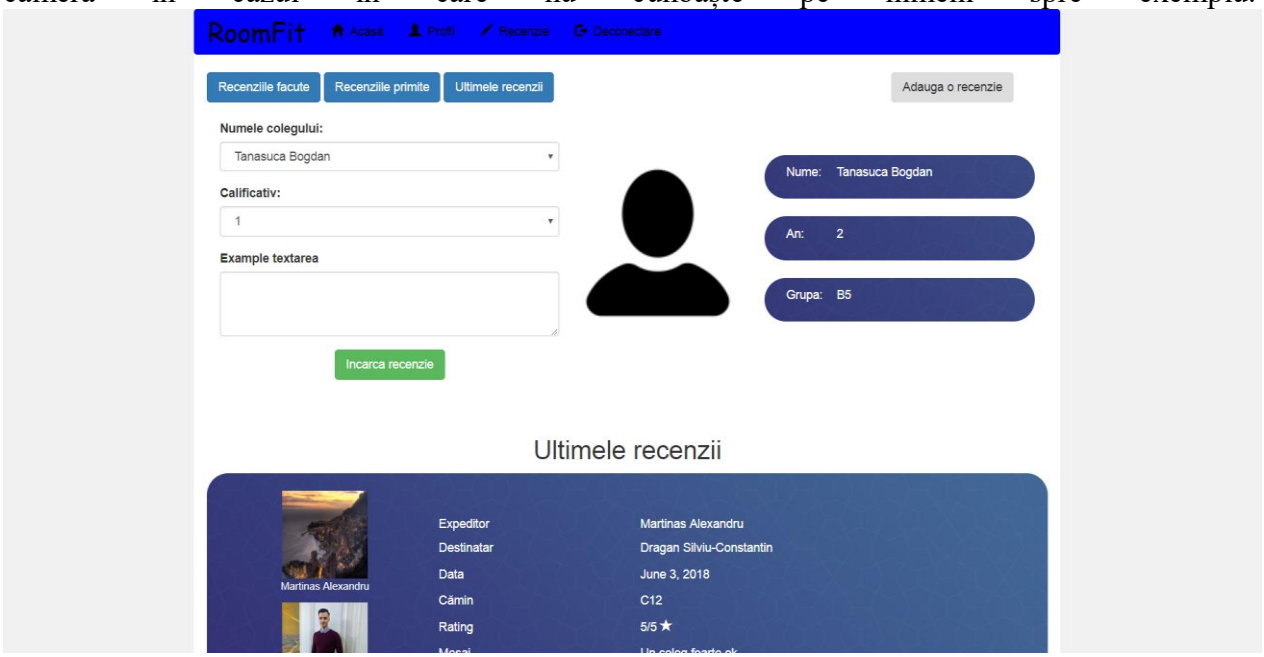
Figura I.3

Sunt 3 stări în care pagina de profil poate fi:

- Momentul în care utilizatorul are posibilitatea să își introducă preferințele; în lista respectivă vor fi doar studenții de la aceeași facultate cu el care au fost repartizați în același cămin. Utilizatorul va putea alege sau nu una sau mai multe persoane cu care își dorește să fie în cameră, introducerea preferințelor fiind în ordine, începând de la persoana cea mai dorită.
- Momentul după ce preferințele au fost introduse; se așteaptă ca administratorul să facă repartizarea studenților în camerele de cămin.
- Momentul în care repartizarea studenților a fost realizată, iar acum sunt afișate informațiile în cauză, numărul camerei în care va sta și studenții ce îi vor fi colegi.

De asemenea, tot pe această pagină este o secțiune de anunțuri unde sunt afișate mesajele din partea administratorului, aceste mesaje putând avea o dată limită pentru a fi realizate, spre exemplu introducere preferințelor până la o anumită dată urmând ca apoi administratorul să facă repartizarea studenților în funcție de acestea.

După cum se poate observa în Figura 4 prezentată mai jos, în cadrul paginii de recenzii un student poate face o recenzie doar persoanelor care i-au fost colegi de cameră putând acorda între 1 și 5 stele și având posibilitatea scrierii unei mici descrieri. De asemenea, acesta poate vedea recenziile făcute în anii precendenți cu scopul de a putea vedea care studenți i-ar fi colegi buni de cameră în cazul în care nu cunoaște pe nimeni spre exemplu.



[1]

Figura I.4

[1] Imagine cu user: https://www.shareicon.net/man-user-customer-847810

I.2. Modul administrator

La pagina de administrator se ajunge tot prin intermediul paginii de logare utilizând credențialele specifice acestei pagini, username-ul fiind „admin” și parola ”istrator2018”.

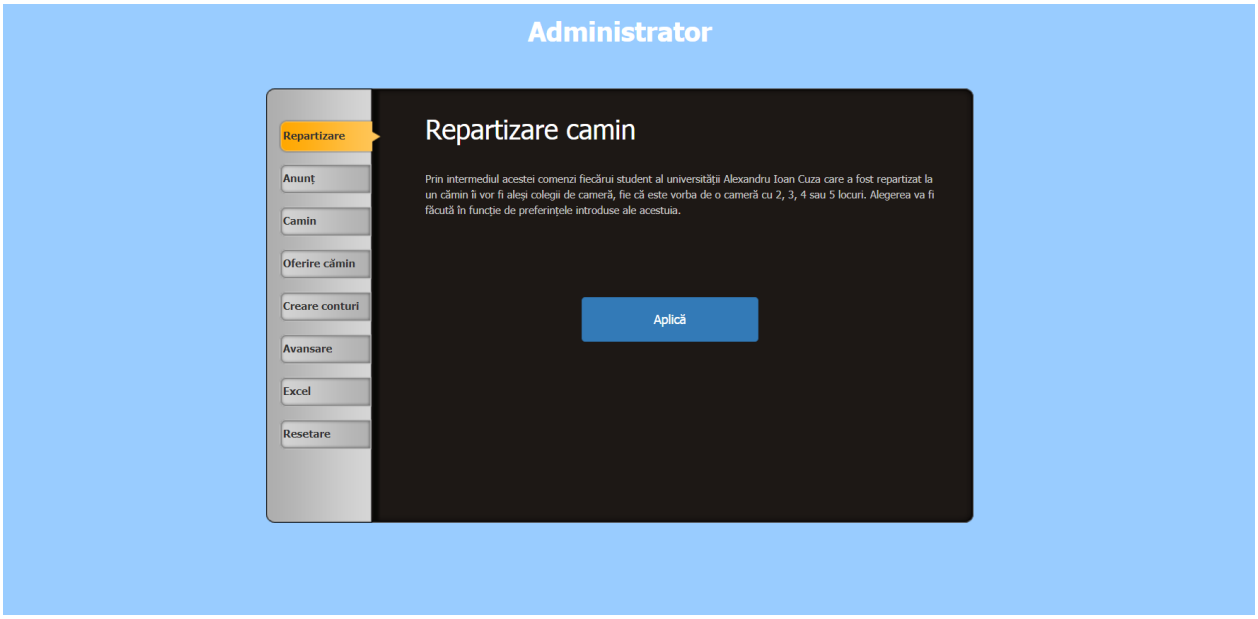


Figura I.5

Odată ajuns pe această pagină administratorul are la dispoziție o listă de acțiuni, după cum se poate observa în Figura 5 de mai sus, pe care le poate folosi, acestea sunt:

- Repartizare cămin: Prin intermediul acestei comenzi și cu ajutorul algoritmului de stable matching roommates este realizată repartizarea studenților în una din camerele căminului la care au fost distribuiți ținându-se cont de preferințele acestora pentru alegerea colegilor de cameră.
- Anunț: Cu ajutorul acestei comenzi administratorul poate să posteze anunțuri, anunțuri ce vor fi afișate în pagina de profil a utilizatorilor. Un anunț este format din titlu, mesaj și o dată limită(deadline) dacă este nevoie, altfel poate să nu fie completat acest câmp.
- Cămin: După ce repartizarea căminului a fost realizată aici se poate vedea pentru fiecare facultate ce camere a avut, în ce cămine și ce studenți stau în ele.
- Oferire cămin: Cu această opțiune se introduc în baza de date, prin intermediul unui fișier .CSV, informațiile referitoare la cine a primit un loc de cazare și în ce cămin. Fișierul trebuie

să conțină pe prima coloană numărul matricol al studentului, iar pe cea de a doua coloană căminul în care a fost repartizat.

- Creare conturi: Această comandă are rolul de a crea conturi pentru studenți, de exemplu la începutul anului universitar pentru studenții din primul an. Crearea conturilor se face prin încărcarea unui fișier .CSV care are următoarele coloane: an de studiu, nume, număr matricol, email, sex, grupă, facultate.
- Avansare: La începere unui nou an universitar prin intermediul acestei comenzi se incrementează anul de studiu al studenților.
- Excel: După ce s-a făcut repartizarea studenților în camere administratorul are opțiunea de a face un export a acestor date într-un Excel în care fiecare facultate are asignat un worksheet în care se află fiecare cameră din fiecare cămin care i-a fost distribuită și persoanele care ocupă acea cameră
- Resetare: În cazul în care apare un eveniment neașteptat, repartizarea nu este realizată cu succes sau se dorește realizarea din nou a repartizării din diferite motive, prin intermediul acestei comenzi sunt șterse datele care sunt generate de comanda „Repartizare cămin”, astfel fiind totul pregătit pentru o nouă redistribuire

II. Tehnologii folosite

II.1. Python

Python este un limbaj de programare orientat obiect, de nivel înalt și interpretabil, având o paletă foarte diversificată de domenii de utilizare, cum ar fi dezvoltare web, machine learning, jocuri video, automatizări și multe altele.

Unul din cele mai mari avantaje ale acestuia este faptul că scrierea algoritmilor se realizează într-un număr de linii considerabil mai mic decât alte limbaje de top cum ar fi C/C++ sau Java, acest lucru conducând la o dezvoltare mai rapidă a produselor software. În același timp codul este ușor de citit de către programator asemănându-se destul de mult cu limba engleză (vezi exemplul de mai jos), prin urmare acest lucru îmbunătățește timpul de înțelegere al unui algoritm.

Exemplu:
```
if student_name in students_list:
    # do_something
```

În timp acest limbaj de programare a devenit din ce în ce mai popular și fiind open-source vine cu o multitudine de biblioteci și framework-uri datorită persoanelor care ajută la îmbunătățirea lui.

Codul scris în acest limbaj va fi mereu frumos organizat datorită unui set de reguli, cunoscut drept PEP 8, iar câteva reguli sunt următoarele:

- Indentarea: în locul clasicelor acolade din alte limbaje de programe pentru a delimita blocurile de cod aici se folosește indentarea, câte 4 spații sau un TAB pentru fiecare nivel de indentare.
- Lungimea maxină a unei linii este de 79 de caractere, după depășirea acestui număr linia respectivă este subliniată și ți se recomandă scrierea pe următorul rând.
- Între două clase se lasă două rânduri libere, iar între două metode ale unei clase un rând.

Cele mai cunoscute librării sunt Astropy, Biopython, Pandas, NumPy și Matplotlib, iar cele mai cunoscute framework-uri sunt Django, Web2py, TurboGears, Pyramid și Flask(microframework).

Pentru realizarea acestei lucrări de licență am folosit varianta 3.6.1 de Python.

II.2. Django framework

II.2.1. Introducere

În cadrul proiectului, aplicația a fost dezvoltată cu ajutorul framework-ului de Python numit Django. Un framework este un set de componente ce vin în ajutorul programatorului pentru a îl ajuta să realizeze o aplicație web mai ușor și mai rapid oferindu-i anumite funcționalități de bază pe care dezvoltatorul aplicației le ia ca atare, nemaifiind nevoit să le implementeze acesta, doar să le modifice acolo unde este cazul.

Printre site-urile care folosesc acest framework se numără Pinterest, Prezi, Washington Post și Instagram.

II.2.2. Securitate

În ceea ce privește securitatea Django vine cu o multitudine de caracteristici, cum ar fi:

- Protecție, folosind templetizarea Django, împotriva atacului de tipul Cross Site Scripting(XSS), fiind unul din cele mai întâlnite atacuri care este caracterizat prin injectarea de cod malițios în interiorul paginilor web.
- Protecție împotriva atacului Cross Site Request Forgery(CSRF) care se realizează prin furtul unei sesiuni ale unui user astfel putând realiza orice fel de operațiune specifică site-ului respectiv ca și cum ar fi făcută de user-ul respectiv.
- SQL injection este atacul prin care pot fi modificate/șterse informații din baza de date sau pot fi furate(scurgeri de informații), realizându-se prin introducerea de cod SQL într-o secțiune de input din pagina web, dar prin utilizarea ORM-ului din Django acest atac este evitat.
- Protecție împotriva atacului Clickjacking care este atunci când un site malițios are alt site într-un frame, astfel accesând acel site îți vor fi furate informații.

II.2.3. ORM

Object-relation mapping(ORM) este o tehnică de programare cu ajutorul căreia programatorul poate manipula mai ușor informația unei baze de date nemaifiind nevoie să scrie interogări SQL.

În continuare voi prezenta diferențele dintre manipularea informației folosind Django ORM și a folosirii interogărilor SQL asupra operațiilor CRUD pe baza de date folosită pentru realizarea aplicației web.

CREATE - realizarea unui anunț

ORM:

```
anunt = Anunt(titlu=”Repartizare cămin”, mesaj=”Repartizarea va fi făcută pe 25.09.2018.”)
anunt.save()
```

Interogare SQL:

```
c = conn.cursor()
c.execute(”INSERT INTO stable_anunt(titlu, mesaj) VALUES(“Repartizare cămin”,”Repartizarea va fi făcută pe 25.09.2018.”)
c.close()
```

unde conn este conexiunea la baza de date

READ – selectarea informațiilor unui student

ORM:

```
info_student = Student.objects.get(numar_matricol=”31090104SL151135”)
```

Interogare SQL:

```
c = conn.cursor()
c.execute(“SELECT * FROM stable_student where numar_matricol=’31090104SL151135’”)
info_student = c.fetchone()
c.close()
```

După cum se poate vedea folosirea ORM-ului este mult mai elegantă și în același timp aplicația este mai sigură, nefiind vulnerabilă la SQL injection spre exemplu.

UPDATE – setare True a câmpului repartizare_camin din tabela Repartizare atunci când un student este asignat unei camere de cămin

ORM:

```
info_repartizare_st = Repartizare.objects.get(numar_matricol=numar_matricol)
info_repartizare_st.repartizare_camera = True
info_repartizare_st.save()
```

Interogare SQL:

```
c = conn.cursor()
c.execute("UPDATE stable_repartizare set repartizare_camera=True where
numar_matricol=%s", numar_matricol)
conn.commit()
c.close()
```

II.2.4. Templetizare

Întrucât este un framework Django are nevoie de un mecanism prin care să poată genera HTML într-un mod dinamic, acest mecanism este Django template oferind un mini-limbaj cu ajutorul căruia programatorul poate transpune informația în pagină.

```
{% extends 'stable/base.html' %}
{% block title %} Profil {% endblock %}
{% load staticfiles %}

{% block profil %}
    <center><h4> Date personale </h4><br></center>
    {% if student.poza_profil|length > 3 %}
       <img src="{{ student.poza_profil.url }}" class="img-responsive"
alt="" height="280" width="280">
    {% else %}
        {% if student.sex == 'F' %}
            <img src="{% static 'stable/images/14.png' %}" class="img-
responsive" alt="" height="280" width="280">
        {% else %}
            <img src="{% static 'stable/images/13.png' %}" class="img-
responsive" alt="" height="280" width="280">
        {% endif %}
    {% endif %}
{% endblock %}
```

Exemplul de mai sus este un fragment din pagina de profil a user-ului. După cum se poate observa pe prima linie este o referință către pagina HTML base, această pagină conține bara de meniu a

aplicației și pentru a respecta principiul DRY am scris acest meniu într-un singur fișier apoi celelalte pagini ce au nevoie de el doar îl extind.

Motivul pentru care se folosește această templetizare este că ne permite să separăm codul HTML de partea de back-end astfel fiind o mai bună structurare.

II.2.5. Crearea unui proiect

Presupunem că pe mașina ce urmează să fie creat proiectul există deja instalate o versiune de Python și Django.

Să spunem că vrem să creăm un proiect nou cu numele Demo, pentru început deschidem Command Prompt și mergem acolo unde dorim să îl creăm și executăm comanda:

```
django-admin startproject Demo
```

Se poate observa că un director cu numele Demo a fost creat având următorul conținut:

- Demo
  - Demo
    - __init__.py
    - settings.py
    - urls.py
    - wsgi.py
  - manage.py

Acum trebuie să intrăm în directorul tocmai creat și să pornim serverul

```
cd Demo
python manage.py runserver
```

După cum putem observa în Figura 1 de mai jos după ce am pornit serverul dacă prin intermediul unui browser accesăm 127.0.0.1:8000 va apărea o pagină simplistă de la Django care are scopul să ne spună că proiectul a fost creat cu succes și este funcțional. Un website de regulă este format din mai multe componente, iar în Django aceste componente se numesc aplicații. Să spunem că dorim să creăm o componentă numită Logare, comanda pentru a o crea este următoarea și trebuie rulată din directorul Demo (cel în care am rămas de mai sus):

```
python manage.py startapp Logare
```

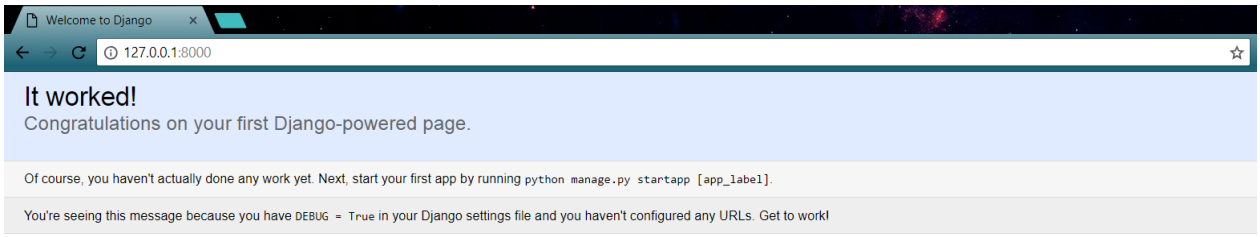


Figura II.1

Dacă ne uităm acum în directorul Demo putem observa că avem următoarea structură de fișiere:

- Demo
  - Demo
    - __init__.py
    - settings.py
    - urls.py
    - wsgi.py
  - Logare
    - Migrations
      - __init__.py
    - __init__.py
    - admin.py
    - apps.py
    - models.py
    - tests.py
    - views.py
  - db.sqlite3
  - manage.py

II.3. HTML, CSS & JavaScript

În ceea ce privesc paginile web, adică ceea ce vede utilizatorul, au fost făcute cu ajutorul HTML, CSS și JavaScript. Scheletul acestor pagini este realizat cu HTML(Hyper Text Mark-up Language) care oferă o structură de bază peste care, folosindu-ne de limbajul CSS cu ajutorul bibliotecii Bootstrap și nu doar, am realizat design-ul acestor pagini. Cascading Style Sheets sau

prescurtat CSS este mijlocul prin care o pagină web este înfrumusețată, oferindu-i astfel un aspect mult mai plăcut și prin urmare o experiență mai plăcută atunci când este folosită de către user.

JavaScript este un limbaj de programare din domeniul de scripting și este folosit pentru a oferi anumite functionalități și efecte în plus. De asemenea, cu ajutorul Ajax-ului am creat câteva funcționalități care dacă sunt utilizate nu mai este nevoie ca pagina să se încarce din nou, ci efectul apăsării acelui buton este văzut imediat.

În momentul de față aceste 3 limbaje sunt atât de utilizate încât vin deja preinstalate în orice browser și având cunoștințele necesare, doar cu ele, pot fi create aplicații cu un grad sporit de complexitate.

II.4. MySQL

Baza de date este locul în care sunt stocate toate informațiile ce vor mai fi folosite din urma interacțiunii utilizatorului cu aplicația web. În baza de date informațiile sunt bine structurate, putând avea mai mult tabele fiecare cu mai multe coloane, iar accesarea lor se face foarte rapid. O bază de date poate să ruleze local, pe mașina celui care folosește aplicația, sau poate să ruleze pe un server așa cum este și în cazul MySQL. În plus, MySQL Workbench este interfața grafică a acestuia putând astfel ca dezvoltatorul aplicației să interacționeze mai ușor cu baza de date.

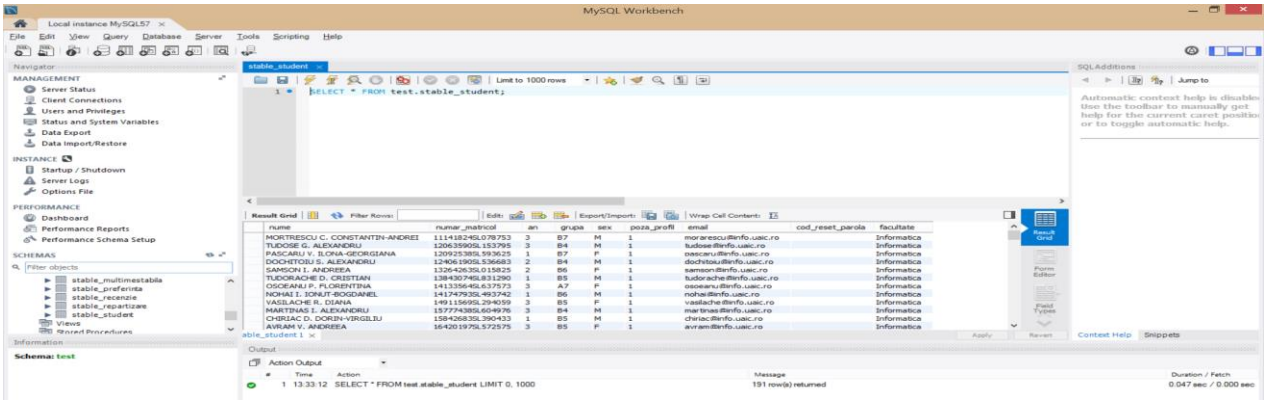


Figura II.2

Operațiile care se pot face asupra unui tabel din baza de date sunt operațiile CRUD:

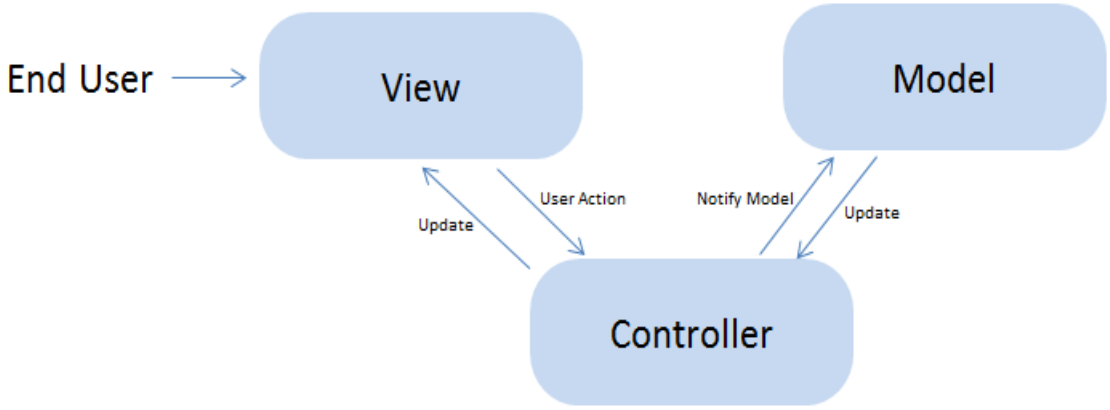
- Create
- Read
- Update
- Delete

Aceste operații sunt foarte ușor de înțeles din această cauză sunt și atât de intens folosite și cu ajutorul lor informația poate fi manipulată în orice fel.

III. Arhitectura aplicației

III.1. MVC

MVC(Model-View-Controller) este un model arhitectural, fiind unul din cele mai utilizate modele care separă logica întregii aplicații în trei părți putând fi astfel ușor de folosit și înțeles datorită modularizării acesteia.



[2]

Figura III.1

III.1.1. Model

Este reprezentarea bazei de date, asemenea unei interfețe care arată structura bazei de date și implicit ce informații va conține aplicația.

Modelul schemei bazei de date din aplicația pentru licență conține 8 tabele:

- Anunt – conține informații despre anunțurile postate de administrator prin intermediul funcționalității „Anunț” din pagina acestuia și conține titlul anunțului, mesajul propriu-zis și un termen limită, dacă este nevoie, pentru anumite anunțuri.
- Camin – conține informații despre camerele căminelor universității ca de exemplu numele căminului, numărul camerei, capacitatea camerei și, în plus, conține facultatea care a primit camera respectivă pentru studenți.
- Coleg – conține numerele matricole a 2 studenți ce au stat în aceeași cameră și căminul în care au fost pentru sistemul de recenzii întrucât un student poate lăsa o recenzie doar unui alt student care i-a fost coleg.

[2] http://www.jsstechitsolutions.com/tutorials/web-dynpro-abap/introduction-to-web-dynpro-abap/mvc-architecture-in-web-dynpro/

- Multime stabila - această tabelă este completată în urma rulării algoritmului de repartizare a studenților în camere și conține informații referitoare la camera de cămin și numerele matricole a studenților ce vor sta în ea în anul respectiv.
- Preferinta – în această tabelă sunt stocate preferințele fiecărui student pentru alegerea colegilor de cameră; tabela conține 2 câmpuri care reprezintă numerele matricole a 2 studenți și importanța preferinței, spre exemplu studentul A îl alege pe studentul B cu importanța 3, asta înseamnă că B se află pe locul 3 în lista de preferințe a studentului A.
- Recenzie – conține numărul matricol al persoanei care a făcut recenzia, numărul matricol al persoanei căreia i-a fost realizată recenzia, efectiv mesajul recenziei și un calificativ reprezentând între 1 și 5 stele.
- Repartizare – această tabelă este completată cu ajutorul funcționalității „Oferire cămin" din pagina administratorului și conține informații despre repartizarea studenților la cămin, ce studenți au fost repartizați la un cămin, la ce cămin și dacă au fost deja asignați unei camere din acesta.
- Student - conține informații personale despre studenții facultăților precum nume, număr matricol, an de studiu, email, informații ce sunt adăugate prin intermediul funcționalității „Creare conturi" din interfața administratorului.

Reprezentarea modelului a câtorva tabele din aplicația web este următoarea:

```
class Student(models.Model):
    nume = models.CharField(max_length=70)
    numar_matricol = models.CharField(max_length=30, primary_key=True)
    facultate = models.CharField(max_length=40)
    an = models.IntegerField(blank=True)
    grupa = models.CharField(max_length=5, blank=True)
    sex = models.CharField(max_length=5)
    poza_profil = models.FileField(default='1')
    email = models.CharField(max_length=50, default="")
    cod_reset_parola=models.CharField(max_length=8,default="",
blank=True)

    def __str__(self):
         return self.numar_matricol + ' - ' + self.nume
```

```
class Recenzie(models.Model):
    class Meta:
       unique_together = (('from_uid', 'to_uid'),)

    from_uid = models.CharField(max_length=30)
    to_uid = models.CharField(max_length=30)
    mesaj = models.CharField(max_length=600)
    calificativ = models.IntegerField(default=1)
    data = models.DateField(u"Conversation Date", auto_now_add=True,
blank=True)

    def __str__(self):
        return self.from_uid + ' catre ' + self.to_uid


class Coleg(models.Model):
    class Meta:
        unique_together = (('coleg1', 'coleg2'),)

    coleg1 = models.CharField(max_length=30)
    coleg2 = models.CharField(max_length=30)
    nume_camin = models.CharField(max_length=30)

    def __str__(self):
        return self.coleg1 + ' cu ' +self.coleg2


class Repartizare(models.Model):
    class Meta:
        unique_together = (('numar_matricol', 'camin'),)

    numar_matricol = models.CharField(max_length=30)
    camin = models.CharField(max_length=30)
    repartizare_camera = models.BooleanField(default=False)
```

De asemenea, Django are un modul de admin destinat vizualizării modelului, precum este prezentat mai jos în Figura 2, și a informațiilor din baza de date care poate fi accesat la http://127.0.0.1:8000/admin/ urmat de numele aplicației pentru care este modelul, dar accesarea se

poate face prin intermediul credențialelor unui cont denumit super-user. Contul de super-user pentru această aplicație are username: silviu și parola: licentasilviu. Tot aici datele pot fi modificate și se pot adăuga/șterge informații în/din baza de date.

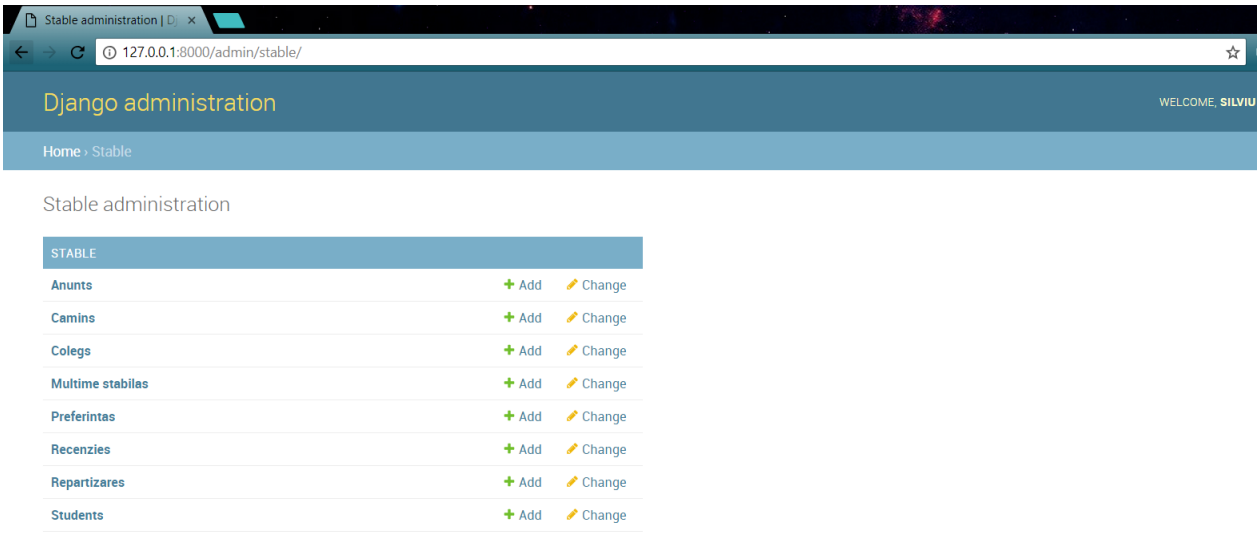


Figura III.2

III.1.2. View

Reprezintă partea vizuală a aplicației, ceea ce vede utilizatorul atunci când accesează dintr-un browser aplicația, este partea cu care acesta interacționeză, totodată colectând și date.

```
{% extends 'stable/base_admin.html' %}
{% load staticfiles %}

{% block descriere_comanda %}
    <div id="form-main">
        <div id="form-div">
            <form class="montform" method="post">
                {% csrf_token %}
                <p class="titlu">
                    <input name="titlu" type="text" required
class="feedback-input" placeholder="Titlu" id="titlu" />
                </p>
                <p class="text">
                    <textarea name="message" required class="feedback-input"
id="comment" placeholder="Mesaj"></textarea>
                </p>
                <p class="data">
```

```
                    <input name="data" type="date" class="feedback-input"
id="data"/>
                </p>
                <div class="submit">
                    <button type="submit" class="button-
blue">Trimite</button>
                    <div class="ease"></div>
                </div>
            </form>
        </div>
    </div>
{% endblock %}

{% block comenzi %}
    <li><a href="{% url 'administrator' %}"> Repartizare </a></li>
    <li><a href="{% url 'anunt' %}" class="selected"> Anunț </a></li>
    <li><a href="{% url 'camin' %}"> Cămin </a></li>
    <li><a href="{% url 'repartizare_camin' %}"> Oferire cămin </a></li>
    <li><a href="{% url 'cont' %}"> Creare conturi </a></li>
    <li><a href="{% url 'avansare' %}"> Avansare </a></li>
    <li><a href="{% url 'excel' %}" > Excel </a></li>
    <li><a href="{% url 'resetare' %}"> Resetare </a></li>
{% endblock %}
```

Această pagină este pagina din care administratorul postează anunțuri. După cum se poate observa pagina extinde base_admin.html întrucât blocul de cod de acolo este comun mai multor pagini și cu ajutorul templetizării putem transpune informația în pagină după bunul plac utilizând tag-ul block.

III.1.3. Controller

Controller-ul este partea ce leagă modelul de view. Aici se tratează cererile care vin din view, adică din interfața utilizatorului și se manipulează informația din model conform cerințelor, iar apoi este actualizat view-ul fiind ca un răspuns al acțiunilor utilizatorului.

Având drept punct de plecare acest model arhitectural framework-ul Django are propria logică când vine vorba de implementare, astfel acesta interpretează diferit noțiunea de MVC, dar funcționalitatea este neschimbată.

III.2. MTV

Modelul arhitectural pe care Django îl urmează se numește MTV(model-template-view). În viziunea celor ce au creat acest framework controller-ul din MVC este tratat de Django însăși, astfel că la crearea unei aplicații programatorul interacționează cel mai mult cu model-ul, template-ul și view-ul după cum se poate observa în Figura 2 de mai jos.

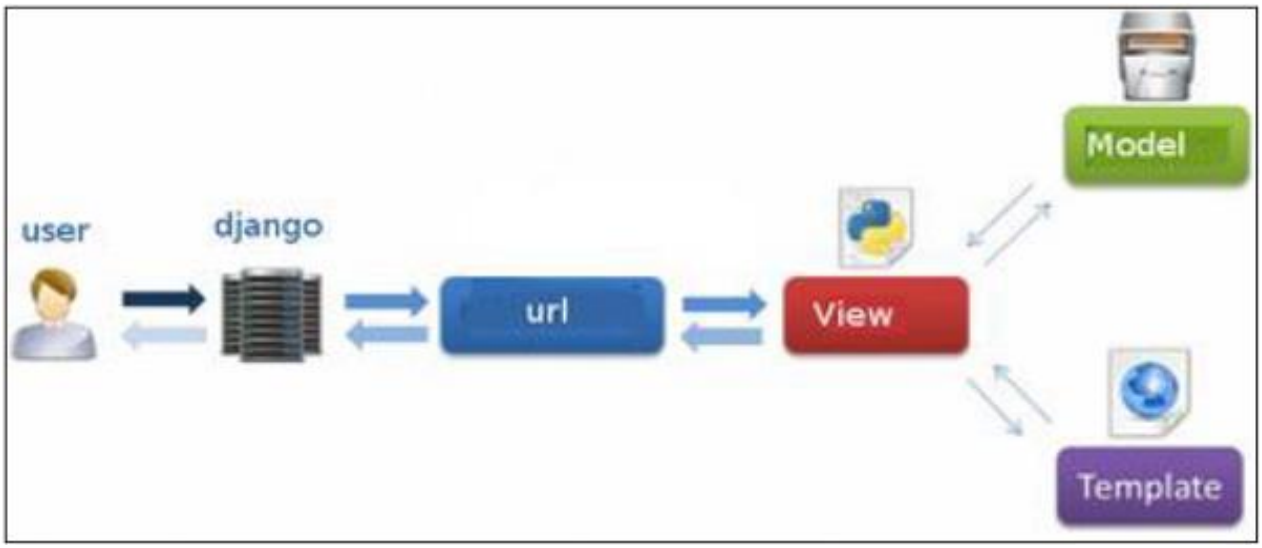


Figura III.2 [3]

Dacă ar fi să facem o paralelă cu MVC observăm următoarele:

- Model: această parte rămâne neschimbată
- Template: în logica acestui framework template-ul este view-ul din MVC întrucât cu ajutorul acestuia se programează cum să fie prezentată informația utilizatorului prin intermediul unei aplicații web
- View: această parte este echivalentul controller-ului din MVC astfel că ambele fac legătura între celelalte 2 părți ale arhitecturii, model și respectiv template/view și tratează cererile astfel încât aici se programează ce informație să îi fie prezentată utilizatorului.

Chiar și cu aceste diferențe ambele arhitecturi conduc către același rezultat creatorii framework-ului spunând că „La finalul zilei, bineînțeles, totul se rezumă la atingerea obiectivului. Și, indiferent cum sunt numite lucrurile, Django duce lucrurile la bun sfârșit într-o manieră care este cea mai logică pentru noi”[4].

[3] https://www.tutorialspoint.com/django/django_overview.htm
[4] https://docs.djangoproject.com/en/2.0/faq/general/

Interpretând Figura 3 de mai sus observăm că utilizatorul, prin intermediul unui browser, accesează o pagină web care este creată cu framework-ul Django introducând un url. După ce url-ul a fost introdus view-ul îl interpretează și accesează funcția de callback specifică acestuia. Callback-ul manipulează informația din model în funcție de cerințele utilizatorului și apoi cu ajutorul template-ului rezultatul este prezentat în pagina website-ului.

Această mapare dintre url și funcția de callback se realizează în fișierul urls.py al fiecărei componente ale aplicației. Mai jos este prezentat un asemenea fișier cu câteva mapări:

```
from django.conf.urls import include, url
from django.contrib import admin
from django.conf import settings
from django.conf.urls.static import static
from stable import views
from stable import administrator

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^index/', include('stable.urls')),
    url(r'^login/$', views.Login.as_view(), name='login'),
    url(r'^resetPass/$', views.ResetPass.as_view(), name='resetPass'),
]
if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,
document_root=settings.STATIC_ROOT)
    urlpatterns += static(settings.MEDIA_URL,
document_root=settings.MEDIA_ROOT)
```

IV. Algoritmul stable matching roommates

Această problemă este asemănătoare cu problema de stable marriage, dar aici toți participanții fac parte din același tabel în loc să fie împărțiți în două mulțimi egale(barbați, respective femei). Dat fiind un număr de N participanți având fiecare N-1 preferințe în ordine descrescătoare, începând de la opțiunea cea mai bună se cere să se găsească cele N/2 mulțimi, astfel încât să avem o potrivire stabilă. Pentru rezolvarea acestei problem putem împărți modul de lucru în două etape.

Etapa 1:

a. Fiecare participant își alege persoana preferată, ordinea de începere neavând importanță întrucât nu afectează rezultatul algoritmului
b. Persoanele propuse aleg partenerul „cel mai preferat”(cu cât este mai la începutul listei de preferințe cu atât este mai dorit)
c. Cei care au fos respinși continuă să facă propuneri până când sunt aleși
d. Fiecare participant respinge potențialii parteneri care sunt mai puțin doriți decât alegerea curentă acceptată

Etapa 2:

a. Se ia un participant care are mai mult de o opțiune(să spunem că este studentul A)
b. Se scrie a doua opțiune a lui A(să spunem că este B), apoi se scrie ultima opțiune a lui B și se repetă acest procedeu până când valoarea A se regăsește din nou
c. Elementele din perechea (k, k+1) cu k = 1 având pasul 2 se resping reciproc
d. Se repetă această etapă până când fiecare participant are o singură opțiune

Un algoritm eficient pentru rezolvarea acestei problem este algoritmul lui Irving având o complexitate de $O(n^2)$.

Pentru a explica mai bine procedeul de rezolvare voi lua un exemplu concret. După cum se poate observa mai jos în Figura 1 avem 6 participanți, fiecare având câte 5 preferințe. De asemenea, în această figură este prezentată starea mulțimii după etapa 1 ce va fi prezentată în cele ce urmează.

A	B	~~D~~	F	~~C~~	~~E~~
B	~~D~~	E	F	A	~~C~~
C	D	E	~~F~~	~~A~~	~~B~~
D	F	C	~~A~~	~~E~~	~~B~~
E	~~F~~	C	~~D~~	B	~~A~~
F	A	B	D	~~C~~	~~E~~

Figura IV.1

Avem următoarele acțiuni posibile:

- propunere
- respingere
- acceptare

Urmând pașii etapei numărul 1 vom avea următoarele propuneri și respingeri:

1. A propune B (iar acesta acceptă)
2. B propune D (iar acesta acceptă)
3. C propune D (îl respinge pe B de la nr. 2 și acceptă propunerea actuală întrucât C este mai dorit decât B în lista lui D)
4. B propune E(acesta acceptă)
5. D propune F(acesta acceptă)
6. E propune F(acesta respinge propunerea întrucât l-a acceptat pe D la nr. 5 care este mai dorit decât E)
7. E propune C(acesta acceptă)
8. F propune A(acesta acceptă)
9. Fiecare participant respinge preferințele care sunt mai puțin dorite decât partenerul curent acceptat

În acest moment starea mulțimii corespunde cu Figura 1.

Pentru etapa a doua vom găsi perechile care se resping scopul final fiind ca fiecare persoană să aibă un singur partener.

Urmând pașii celei de a doua etape ajungem la Figura 2 astfel:

1. Alegem un participant cu mai mult de o preferință rămasă și îl trecem în tabel(îl alegem pe A)
2. În dreptul participantului anterior trecem ultima preferință a acestuia(F)
3. Trecem ultima preferință a lui F(participantul D)
4. Trecem a doua preferință a lui D(participantul C)
5. Trecem ultima preferință a lui C(participantul E)
6. A doua preferință a lui E(participantul B)
7. Ultima preferință a lui B(participantul A)
8. A se repetă deci trebuie să ne oprim(vom respinge bidirecțional perechile de participanți F-D, C-E și B-A)

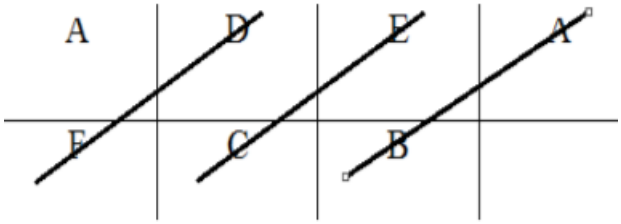


Figura IV.2

Mai sunt participanți care au mai mult de o preferință și potrivit subpunctului d din etapa 2 trebuie să mai repetăm încă o dată instrucțiunile acestei etape; în Figura 3 este prezentat acest lucru.

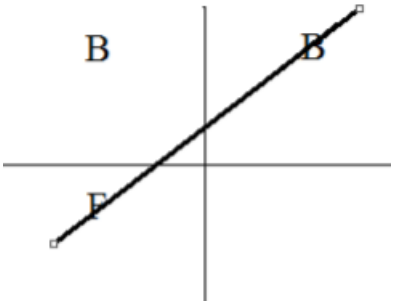


Figura IV.3

După iterația a doua a acestei etape putem observa în Figura 4 de mai jos că acum fiecare participant are câte un partener, deci s-au găsit cele N/2 mulțimi, în cazul nostru 3 mulțimi.

A	~~B~~	~~D~~	F	~~C~~	~~E~~
B	~~D~~	E	~~F~~	~~A~~	~~C~~
C	D	~~E~~	~~F~~	~~A~~	~~B~~
D	~~F~~	C	~~A~~	~~E~~	~~B~~
E	~~F~~	~~C~~	~~D~~	B	~~A~~
F	A	~~B~~	~~D~~	~~C~~	~~E~~

Figura IV.4

Rezultatul algoritmului este cele 3 mulțimi având fiecare câte 2 participanți și anume:

- A – F
- B – E
- C – D

V. Implementare

Cum fiecare lucrare de licență ar trebui să aibă la bază un algoritm, nici aceasta nu face rabat, principalul algoritm fiind cel de stable matching roommates. În cele ce urmează voi prezenta codul acestui algoritm pentru o mai bună înțelegere a acestuia.

Funcția „stable” este funcția din Figura 1 și 4 ce realizează repartizarea studenților la cămin, apelând și câteva funcții adiționale în interiorul acesteia care de asemenea vor fi prezentate și ele.

```
40  def stable(multime):
41      while True:
42          for st in multime:
43              if st['propose'] == "":
44                  for option in st['preferences']:
45                      accept = free(multime, option)
46                      if accept == "":
47                          st['propose'] = option
48                          multime = propose(multime, st['name'], option)
49                          break
50                      else:
51                          better_choice = who_is_beter(multime, st['name'], option, accept)
52                          if better_choice == st['name']:
53                              multime = abandon_current_accept(multime, option)
54                              st['propose'] = option
55
56                          multime = propose(multime, better_choice, option)
57                           if better_choice == st['name']:
58                              break
59
60          if len([st for st in multime if st['propose'] == ""]) == 0:
61              break
62
63      for st in multime:
64          to_delete = st['preferences'][st['preferences'].index(st['accept']) + 1:]
65          for it in to_delete:
66              for i in multime:
67                  if it == i['name']:
68                      try:
69                          i['preferences'].remove(st['name'])
70                          st['preferences'].remove(it)
71                      except Exception:
72                          pass
73                      break
```

Figura V.1

Acestă funcție primește drept input o listă de dicționare, fiecare dicționar reprezentând un student și având următoarele informații {'name': , 'propose': , 'accept': ,'preferences': []} numele, pe cine propune el, pe cine acceptă și o listă în care se află preferințele acestuia.

În prima structură repetitivă se ia fiecare element din listă și respectivul student dacă nu a propus deja pe cineva(linia 43) va lua studenții din lista de preferințe introdusă de el din pagina de

profil până va accepta unul. Studentul selectat din lista de preferințe este verificat dacă a acceptat pe altcineva cu ajutorul funției „free” (linia 45) prezentată de asemenea mai jos în figura 2.

```
1  def free(multime, option):
2      for st in multime:
3          if option == st['name']:
4              return st['accept']
5
```

Figura V.2

Dacă a acceptat se verifică dacă opțiunea curentă este una mai bună pentru a renunța la cea anterioară și a accepta-o pe cea nouă (liniile 51 – 58) cu ajutorul funcțiilor „who_is_better”, „abandon_current_accept” și „propose” prezentate în figura 3 mai jos , altfel va accepta direct propunerea întrucât nu are pe nimeni (liniile 46 – 49). Bucla respectivă se oprește atunci când fiecare student a propus pe cineva, deci implicit și fiecare student a acceptat pe cineva.

```
5
6
7  def propose(multime, searcher, option):
8      for st in multime:
9          if option == st['name']:
10             st['accept'] = searcher
11     return multime
12
13
14 def who_is_beter(multime, searcher, option, current):
15     for st in multime:
16         if option == st['name']:
17             for p in st['preferences']:
18                 if p == searcher:
19                     return searcher
20                 if p == current:
21                     return current
22
23
24 def abandon_current_accept(multime, option):
25     for st in multime:
26         if option == st['name']:
27
28             current_accept = st['accept']
29             try:
30                 st['preferences'].remove(current_accept)
31             except Exception:
32                 pass
33
34             for st_rm in multime:
35                 if st_rm['name'] == current_accept:
36                     st_rm['preferences'].remove(option)
37                     st_rm['propose'] = ""
38     return multime
39
```

Figura V.3

Funcția who_is_better verifică dacă pentru studentul actual este mai bun colegul care îi este asignat deja sau poate renunța la el în favoarea noului-venit. Acest lucru se realizează prin verificarea listei de preferințe, prima persoană care este găsită în listă va fi cea selectată întrucât, fiind mai la începutul listei față de persoana cealaltă,

În acest punct mai este doar un pas până să ajungem la o repartizare în camere a studenților. Acest pas este regăsit în figura 4 de mai jos și constă în eliminarea ciclurilor.

```
74
75      while True:
76          first_line = []
77          second_line = []
78          for st in multime:
79              if len(st['preferences']) > 1:
80                  first_line.append(st['name'])
81                  second_line.append(st['preferences'][1])
82                  while len(first_line) == len(set(first_line)):
83                      for i in multime:
84                          if i['name'] == second_line[-1]:
85                              try:
86                                  first_line.append(i['preferences'][-1])
87                              except Exception:
88                                  pass
89                              break
90                      for i in multime:
91                          if i['name'] == first_line[-1]:
92                              second_line.append(i['preferences'][1])
93                              break
94                  break
95
96          for i in range(1, len(first_line)):
97              for st in multime:
98                  if st['name'] == first_line[i]:
99                      st['preferences'].remove(second_line[i - 1])
100                 if st['name'] == second_line[i - 1]:
101                     st['preferences'].remove(first_line[i])
102         if len([st for st in multime if len(st['preferences']) > 1]) == 0:
103             break
104     return multime
```

Figura V.4

Scopul acestei secvențe de cod este ca fiecare student în lista de preferințe a lui să rămână cu un singur număr matricol acesta fiind de fapt numărul matricol al viitorului coleg de cameră. La finalul acestei funcții lista de dicționare este returnată, acest rezultat fiind preluat de altă funcție care îl va parsa cu scopul de a interpreta rezultatul și în cele din urmă de a introduce în baza de date mulțimile stabile.

În ceea ce privește repartizarea studenților în camere de 3, 4 și 5 persoane algoritmul de bază rămâne același, doar mulțimea oferită drept parametru funcției stable diferă.

Astfel pentru camerele de 3 persoane întâi se vor face perechi de câte 2, apoi cu ajutorul unei funcții vom calcula un punctaj ce reprezintă cât de potrivită este alegerea făcută. Pentru fiecare pereche în parte punctajul se calculează adunând indexul din lista de preferințe a numărului matricol al colegului, spre exemplu, pentru perechea (A, B) scorul va fi poziția unde se află B în lista de preferințe a lui A adunat cu poziția unde se află A în preferințele lui B. În cazul în care avem un coleg fictiv se va aduna un număr foarte mare pentru a nu influența repartizarea. Un coleg fictiv este doar un dicționar în lista de studenți primită drept parametru și este introdus atunci când avem un număr impar de elemente. După calcularea acestui punctaj lista este ordonată crescător în funcție de aceste valori și doar primele X mulțimi vor fi păstrate, adică cele mai bune potriviri, unde X este rezultatul împărțirii numărul de studenți la 3 rotunjit superior, adică minimul de camere în care vor încăpea studenții respectivi. Perechile care nu sunt în primele X vor fi desfăcute, iar acum în listă vom avea dicționare care reprezintă un singur student și dicționare pentru perechi de câte 2 studenți. Pentru a putea rula din nou algoritmul mai trebuie doar ca perechile de câte 2 studenți să aibă în lista de preferințe studenții singuri și invers deoarece o pereche nu trebuie să poată alege o altă pereche.

Pentru camerele de 4 persoane se execută o dată algoritmul creând perechi de câte 2, apoi se adaugă un coleg fictiv sau se șterge unul, dacă există, în cazul în care numărul de elemente este impar și în cele din urmă se mai rulează încă o dată algoritmul cu noua mulțime obținând acum perechi de 4 studenți.

La camerele de 5 persoane întâi se obțin perechi de câte 2, apoi cu ajutorul funcției de scor prezentată și la repartizarea în camere cu 3 locuri se aleg primele 2*X perechi, unde X este rezultatul împărțirii numărului de studenți la 5 rotunjit superior. Cu aceste 2*X perechi se execută o dată algoritmul având astfel perechi de câte 4, iar perechile rămase anterior vor fi desfăcute. Preferințele vor fi reactualizate, perechile de câte 4 neputând avea în lista lor încă o pereche, ci doar studenți singuri și după încă o execuție a funcției stable vom avea drept rezultat perechi de câte 5 persoane.

Când pagina de recenzii este accesată recenziile sunt afișate în ordine descrescătoare față de cum au fost postate, de la cea mai recentă la cea mai veche. Astfel putând avea un număr foarte mare de recenzii am hotărât realizarea unei paginări a acestora.

Inițial în views.py sunt extrase recenziile din baza de date și apoi împărțite în pagini cu câte 5 recenzii/pagină am ales cu ajutorul librăriei paginator după cum se poate observa mai jos.

```
from django.core.paginator import Paginator
template_name = 'stable/recenzie.html'
page = request.GET.get('page', 1)
recenzii = obtine_recenzii()
p = Paginator(recenzii, 5)
return render(request, self.template_name, {'recenzii': p.page(page)})
```

Apoi în pagina HTML aceste recenzii sunt afișate

```
{% if recenzii.has_other_pages %}
  <ul class="pagination">
    {% if recenzii.has_previous %}
      <li><a href="?page={{ recenzii.previous_page_number }}"> &laquo;</a>
</li></li>
    {% else %}
      <li class="disabled"><span>&laquo;</span></li>
    {% endif %}
    {% for i in recenzii.paginator.page_range %}
      {% if users.number == i %}
        <li class="active"><span>{{ i }}<span class="sr-only">
(current)</span></span></li>
      {% else %}
        <li><a href="?page={{ i }}">{{ i }}</a></li>
      {% endif %}
    {% endfor %}
    {% if recenzii.has_next %}
      <li><a href="?page={{ recenzii.next_page_number }}">&raquo;</a></li>
    {% else %}
      <li class="disabled"><span>&raquo;</span></li>
    {% endif %}
  </ul>
{% endif %}[5]
```

[5] https://simpleisbetterthancomplex.com/tutorial/2016/08/03/how-to-paginate-with-django.html

După ce un utilizator se loghează, pe prima pagină ce îi este afișată poate vedea unde se situează fiecare cămin al Universității Alexandru Ioan Cuza pe o harta cu ajutorul unor markeri în care se specifică exact coordonatele geografice. Acestu lucru se realizează într-o funcție de JavaScript, iar un marker se adaugă astfel:

```
function myMap(){
    var fii = {lat: 47.1740032, lng: 27.5748576};
    var mapOptions = {
        center: fii,
        zoom: 15,
        mapTypeId: google.maps.MapTypeId.HYBRID
    }

    var map = new google.maps.Map(document.getElementById("map"),
mapOptions);
    var marker = new google.maps.Marker({
        position: {lat: 47.1882123, lng: 27.5625704},
        map: map,
        title: 'Cămin C1'
    });
}[6]
```

La prima logare sau de fiecare dată când utilizatorul își uită parola acesta își va schimba parola prin accesarea „Resetare parolă” din pagina de logare unde este va fi nevoit să își introducă adresa de email primită de la facultate pentru a primi un cod de resetare urmând ca acest cod să îl introducă în formular pentru schimbarea cu succes a parolei. Pentru trimiterea email-urilor am creat un cont generic de gmail numit stablematchingroommates iar trimiterea efectivă a unui email se face astfel:

```
import smtplib
fromaddr = 'stablematchingroommates@gmail.com'
toaddrs = destinatar
msg = "Foloseste codul urmator pentru resetarea parolei: " + cod_verificare
+ "."
username = 'stablematchingroommates@gmail.com'
```

[6] https://developers.google.com/maps/documentation/javascript/markers

```
password = 'StableRommates135680'
SUBJECT = "Resetare parola"

server = smtplib.SMTP('smtp.gmail.com:587')
server.ehlo()
BODY = '\r\n'.join(['To: %s' % destinatar,
                    'From: %s' % fromaddr,
                    'Subject: %s' % SUBJECT,
                    '', msg])
server.starttls()
server.login(username, password)
server.sendmail(fromaddr, toaddrs, BODY)
server.quit()
```
[7]

Totuși pentru ca un email să poată fi trimis trebuie ca din setările contului de Google să activăm accesul aplicațiilor mai puțin sigure.

[7] https://stackoverflow.com/questions/6270782/how-to-send-an-email-with-python

VI. Manual de utilizare

În vederea pornirii aplicației pe altă mașină sunt necesari parcurgerea următorilor pași:

- Instalare Python(versiunea 3.6.1) - https://www.python.org/downloads/
- Instalare Django(versiunea 1.10.6) – pip install django==1.10.6 în Command Prompt
- Instalare MySQL Community Server 5.7 - https://dev.mysql.com/downloads/windows/installer/5.7.html
    - Pachetul „Developer default”
    - Când trebuie să adăugați o parolă pentru Workbench introduceți parola „Silviu01”
- Descărcarea codului sursă - https://github.com/silviudragan/Stable-matching-roommates
- Deschideți Workbench și accesați „Local Instance Router” folosind parola de mai sus
- Creați o nouă schemă cu numele „test” prin intermediul butonului „Create a new schema in the connected server”
- Acum trebuie să importăm baza de date: click „Data Import/Restore”, alegem calea către baza de date, apăsăm „Load Folder Contents” și apoi „Start Import”. În acest moment, după ce dăm un refresh, ar trebui să avem noile tabele cu datele aferente.
- Intrăm în directorul descărcat de pe GitHub iar apoi în directorul „website” și deschidem o fereastră de comandă
- Rulăm comanda „python manage.py runserver” și apoi accesând dintr-un browser http://127.0.0.1:8000/login/ vom putea folosi aplicația
- Pentru a putea modifica codul sursă va trebui instalat PyCharm de la JetBrains https://www.jetbrains.com/pycharm/download/#section=windows

Concluzii

Pe scurt, punerea în funcțiune a acestei aplicații ar fi o îmbunătățire adusă procesului de repartizare a studenților la cămin întrucât nu va mai fi nevoie ca studenții să vină în timpul vacanței în Iași pentru a lua dispoziția de cazare de la facultate împreună cu prietenii lor pentru a putea lua o aceeași cameră, ci vor afla prin intermediul aplicației în ce cameră au fost repartizați și cu ce colegi în funcție de preferințele introduse, bineînțeles.

Aplicația este simplistă și ușor de utilizat, atât partea pentru utilizatorii normali(studenți) care cu doar câteva click-uri pot alege cu cine sta în cameră sau pot lăsa o recenzie, cât și partea de administrator care prin realizarea unor importuri de fișiere .CSV au totul pregătit pentu a se loga studenții sau pentru a face repartizarea lor la cămin. Aceste lucruri fac ca aplicația să fie eligibilă pentru toate facultățile universității, nefiind nevoie de cunoștințe tehnice pentru a o putea utiliza. Cu toate acestea ar putea fi testată în primul an de Facultatea de Informatică timp în care ar putea fi îmbunătățită prin implementarea unor noi funcționalități și apoi să fie pusă la dispoziția întregii universități.

Funcționalitățile care ar putea fi adăugate pentu a îmbunătăți aplicația sunt următoarele:

- Posibilitatea de a opta pentru cămin tot prin intermediul aplicației, atât la cazări cât și la recazări
- Recomandare de studenți: când te loghezi pentru prima dată va trebui să răspunzi la câteva întrebări generale cum ar fi ce hobby-uri ai sau ce filme/muzică preferi și pe baza acestor răspunsuri îți vor fi recomandate persoane care au răspuns similar cu tine, putând astfel să îi introduci în lista de preferințe dacă dorești
- Posibilitatea logării în aplicație prin intermediul contului de eSIMS

Bibliografie

1. Robert W. Irving, An Efficient Algorithm for the “Stable Roommates” Problem
2. Douglas Crockford, JavaScript: The Good Parts
3. Jon Ducket, HTML & CSS Design and Build Websites
4. Documentația oficială Django
   https://docs.djangoproject.com/en/2.0/
5. Librăria xlsxwriter pentru crearea fișierelor Excel
   https://xlsxwriter.readthedocs.io/
6. Securitatea în Django
   https://docs.djangoproject.com/en/2.0/topics/security/
7. Templetizarea în Django
   https://docs.djangoproject.com/en/2.0/topics/templates/
8. Design Pattern Django
   https://djangobook.com/model-view-controller-design-pattern/
9. Coding style
   https://www.python.org/dev/peps/pep-0008/
10. https://simpleisbetterthancomplex.com/tutorial/
11. https://www.tutorialspoint.com/python/python_database_access.htm