



UNIVERZITET U NOVOM SADU  
PRIRODNO-MATEMATIČKI  
FAKULTET  
DEPARTMAN ZA MATEMATIKU  
I INFORMATIKU



# **Sistem za vođenje stomatološke ordinacije**

(Seminarski rad)

Predmet: Skript jezici

Student:

Branislav Bozejac 20/17

Novi Sad  
2019.godina

## Sadržaj:

1. Uvod .....	3
2. Opis .....	3
3. Klasa “models.person.Person” .....	4
4. Fajl “models/nurse.py” .....	5
4.1. Klasa “models.nurse.Nurse” .....	5
4.2. Metode u fajlu “models/nurse.py” .....	6
5. Fajl “models/doctor.py” .....	8
5.1. Klasa “models.doctor.Doctor” .....	8
5.2. Metode u fajlu “models/doctor.py” .....	9
6. Fajl “models/patient.py” .....	14
6.1. Klasa “models.patient.Patient” .....	14
6.2. Metode u fajlu “models/doctor.py” .....	15
7. Fajl “models/appointment.py” .....	20
7.1. Klasa “models.appointment.Appointment” .....	20
7.2. Metode u fajlu “models/appointment.py” .....	21
8. Fajl “models/admin.py” .....	23
8.1. Klasa “models.admin.Admin” .....	23
8.2. Metode u fajlu “models/admin.py” .....	24
9. Glavni fajl “Dentistry.py” .....	26
10. Direktorijum “database/” .....	29
10.1. Fajl “database/login.txt” .....	29
10.2. Fajl “database/empList.txt” .....	29
10.3. Fajl “database/interventions.txt” .....	30
10.4. Fajl “database/patients.txt” .....	30
10.5. Fajl “database/appointments.txt” .....	31
11. Zaključak .....	32
12. Literatura .....	33

## 1. Uvod

**Python** je skript programski jezik. Program je prvi put implementiran 1989. godine u Holandiji na Univerzitetu Stičing. Programski jezik **python** je platformski nezavistan, objektno-orijentisan, interpreterski i interaktivan. U ovom programskom jeziku je napisan program **Dentistry**.

**Program Dentistry** je napravljen sa ciljem da se olakšaju svakodnevni poslovi u jednoj stomatološkoj ordinaciji poput zakazivanja pacijenata, traženje pacijenata, unošenje novog pacijenta itd.

## 2. Opis

**Program Dentistry** podržava login-sistem za zaposlene (doktore, medicinske sestre i admina). U zavisnosti od tipa zaposlenih, opcije koje mogu da izvršavaju su drugačije, na primer: medicinska sestra može da zakaže intervenciju i da unese novog pacijenta dok doktor to ne može da uradi.

Direktorijumi su koncipirani na sledeći način:

- **Dentistry\_PMF/** - glavni (root) direktorijum
- **models/** - direktorijum gde se nalaze svi .py fajlovi
- **database/** - direktorijum gde se nalaze svi tekst fajlovi koji se koriste u program

Kod samog programa je moguće naći na sledećoj internet adresi:

[https://github.com/banebo/Dentistry\\_PMF](https://github.com/banebo/Dentistry_PMF)

### 3. Klasa “models.person.Person”

Klasa **models.person.Person** *Slika br.1* predstavlja jednu osobu.

Klasa **Person** ima sledeća polja:

1. **\_\_name** – predstavlja privatno polje koje čuva ime osobe
2. **\_\_surname** – predstavlja privatno polje koje čuva prezime osobe

Ovu klasu nasleđuju klase “Patient”, “Doctor” i “Nurse”.

U klasi su definisane sledeće metode:

1. **\_\_str\_\_()** – Vraća string reprezentaciju u obliku “ime:prezime”
2. **get\_name()** – Vraća promenljivu **\_\_name**
3. **get\_surname()** – Vraća promenljivu **\_\_surname** –

```
1  #!/usr/bin/env python3
2
3  class Person:
4      def __init__(self, name, surname):
5          self.__name = name
6          self.__surname = surname
7
8      def __str__(self):
9          return self.get_name()+":"+self.get_surname()
10
11     def get_name(self): return self.__name
12     def get_surname(self): return self.__surname
```

*Slika br.1*

## 4. Fajl “models/nurse.py”

U fajlu **models/nurse.py** predstavljena je klasa **Nurse** koja predstavlja jednu stomatološku sestru i metode koje stomatološka sestra može da izvršava.

Importuju se sledeće biblioteke, moduli i klase *Slika br. 2*:

- `os`
- `models.patient`
- `models.person.Person`
- `models.appointment`
- `models.admin`

```
3 import os
4 from models import patient
5 from models.person import Person
6 from models import appointment
7 from models import admin
```

*Slika br. 2*

### 4.1. Klasa “models.nurse.Nurse”

Klasa **models.nurse.Nurse** *Slika br. 3* predstavlja klasu stomatološke sestre. Ova klasa nasleđuje klasu **models.person.Person** i ima sledeća polja:

1. **\_\_name** – predstavlja privatno polje koje čuva ime stomatološke sestre
2. **\_\_surname** – predstavlja privatno polje koje čuva prezime stomatološke sestre
3. **\_\_id** – predstavlja korisničko ime stomatološke sestre

Klasa **Nurse** ima sledeće metode:

1. **get\_id()** - Vraća korisničko ime korisnika
2. **\_\_str\_\_()** – Redefinisana metoda iz nadklase koja vraća string formata “korisničko\_ime:ime:prezime:nurse”

```

10 class Nurse(Person):
11     def __init__(self, id, name, surname):
12         super().__init__(name, surname,)
13         self.__id = id
14
15     def __str__(self):
16         return \
17             self.get_id()+" "+super().__str__()+"nurse"
18
19     def get_id(self): return self.__id

```

*Slika br. 3*

## 4.2. Metode u fajlu “models/nurse.py”

U fajlu **models/nurse.py** implementirane su sledeće metode:

- **nurse\_menu(info)** – metod koji štampa opcije pozivajući pomoćni metod `printNurseMenu(info)` i čeka korisnikov unos. Kada se unese validna opcija, pozivaju se metode za izvršavanje određenih operacija *Slika br. 4*. Metod `nurse_menu(info)` kao parameter prima rečnik sa informacijama o trenutnom korisniku. Stomatološka sestra ima opcije da unese novog pacijenta, unese novi pregled, pretražuje pacijente, pretražuje preglede, štampa sve predstojeće preglede sortirane po najskorijim i da promeni šifru.
- **printNurseMenu(info)** – pomoćni metod za metod `nurse_menu(info)`. Štampa opcije stomatološke sestre. – *Slika br. 5*

```

21 def nurse_menu(info):
22     while True:
23         #os.system("clear")
24         printNurseMenu(info)
25         choice = input("\n>> ")
26         while choice.lower() not in ('1','2','3','4','5','6','x'):
27             print("\n[-] Bad option "+choice)
28             printNurseMenu(info)
29             choice = input("\n>> ")
30
31         if choice == '1': appointment.make_appointment()
32         elif choice == '2': patient.new_patient()
33         elif choice == '3': appointment.search_appointment()
34         elif choice == '4': patient.search_patient()
35         elif choice == '5': appointment.print_all_appointments()
36         elif choice == '6': admin.change_password()
37         elif choice == 'x': exit(0)

```

*Slika br. 4*

```
39 def printNurseMenu(info):
40     print("\nLogged-in as %s %s (%s)" % \
41         (info['name'],info['surname'],info['id']))
42
43     print("\n\t[1] Make appointment")
44     print("\t[2] New patient")
45     print("\t[3] Search appointment")
46     print("\t[4] Search patient")
47     print("\t[5] View schedule")
48     print("\t[6] Change password")
49     print("\t[x] Logout and exit")
```

*Slika br. 5*

## 5. Fajl “models/doctor.py”

U fajlu **models/doctor.py** predstavljena je klasa **Doctor** kao i metode koje se izvršavaju nad objektima ove klase npr. `get_doctors()`.

Importuju se sledeće biblioteke, moduli i klase *Slika br. 6*:

- `os`
- `hashlib`
- `datetime`
- `models.person.Person`
- `models.appointment`
- `models.admin`

```
3 import os
4 import hashlib
5 import datetime
6 from models.person import Person
7 from models import patient
8 from models import appointment
9 from models import admin
```

*Slika br. 6*

### 5.1. Klasa “models.doctor.Doctor”

Klasa **Doctor** predstavlja klasu doktor *Slika br. 7*. Ova klasa nasleđuje klasu **models.person.Person**.

Ova klasa je definisana sledećim poljima:

1. **\_\_name** – predstavlja privatno polje koje čuva ime doktora
2. **\_\_surname** – predstavlja privatno polje koje čuva prezime doktora
3. **\_\_id** – predstavlja korisničko ime doktora stomatologije

Klasa **Doctor** ima sledeće metode:

1. **get\_id()** - Vraća korisničko ime korisnika
2. **\_\_str\_\_()** – Redefinisana metoda iz nadklase koja vraća string formata “korisničko\_ime:ime:prezime:doctor”
3. **\_\_equals\_\_(obj)** – Vraća True ako je objekat isti
4. **\_\_hash\_\_()** – Vraća md5 hash objekta



```

11 class Doctor(Person):
12
13     def __init__(self, id, name, surname):
14         super().__init__(name, surname)
15         self.__id = id
16
17     def __str__(self):
18         return self.get_id()+" "+super().__str__()+"doctor"
19
20     def get_id(self): return self.__id
21
22     def __equals__(self, other):
23         if not other:
24             return False
25
26         return (self.__class__ == other.__class__) and \
27             (self.get_id() == other.get_id()) and \
28             (self.get_name() == other.get_name()) and \
29             (self.get_surname() == other.get_surname())
30
31     def __hash__(self):
32         info = self.get_id() + self.get_name() + self.get_surname()
33         return hashlib.md5(info.encode("utf-8")).hexdigest()

```

*Slika br. 7*

## 5.2. Metode u fajlu “models/doctor.py”

U fajlu **models/doctor.py** implementirane su sledeće metode:

- **doctor\_menu(info)** – metod koji štampa opcije pozivajući pomoćni metod `prinDoctorMenu(info)` i čeka korisnikov unos. Kada se unese validna opcija, pozivaju se metode za izvršavanje određenih operacija. Metod `doctor_menu(info)` kao parameter prima rečnik sa informacijama o trenutnom korisniku. Doktor ima opcije da pretražuje zakazane preglede, modifikuje opis pregleda, vidi sve predstojeće preglede, pretražuje pacijente, pogleda kolika mu je mesečna plata, promeni šifru *Slika br. 8*.
- **printDoctorMenu(info)** – pomoćni metod za metod `nurse_menu(info)`. Štampa opcije stomatološke sestre –

```

35 def dr_menu(info):
36     while True:
37         os.system("clear")
38         printDrMenu(info)
39         choice = input("\n>> ")
40         while choice.lower() not in ('1','2','3','4','5','6','x'):
41             print("\n[-] Bad option "+ choice if(choice) else " ")
42             printDrMenu(info)
43             choice = input("\n>> ")
44
45         if choice == '1': appointment.search_appointment()
46         elif choice == '2': appointment.modify_appointment_details()
47         elif choice == '3': appointment.print_all_appointments()
48         elif choice == '4': patient.search_patient()
49         elif choice == '5': print_salary(info)
50         elif choice == '6': admin.change_password(info)
51         elif choice == 'x': exit(0)
52
53 def printDrMenu(info):
54     print("\nLogged-in as dr. %s %s (%s)" % \
55           (info['name'],info['surname'],info['id']))
56
57     print("\n\t[1] Search appointments")
58     print("\t[2] Modify appointment details")
59     print("\t[3] View all appointments")
60     print("\t[4] Search patient")
61     print("\t[5] View salary")
62     print("\t[6] Change password")
63     print("\t[x] Logout and exit")

```

*Slika br. 8*

- **get\_doctor(hash)** – Vraća objekat tipa Doktor na osnovu heša koji se prosledjuje. *Slika br. 9*

```

65 def get_doctor(hash):
66     doctors = get_doctors()
67     for dr in doctors:
68         if hash == dr.__hash__():
69             return dr
70     return None

```

*Slika br. 9*

- **get\_doctors()** – Vraća listu objekata tipa Doctor. Metod učitava podatke o doktorima iz fajla “database/empList.txt”. *Slika br. 10*

```
72 def get_doctors():
73     if not os.path.isfile("database/empList.txt"):
74         return None
75
76     doctors = []
77     file = open("database/empList.txt", "r")
78     for line in file:
79         info = line.strip(" \n").split(':')
80         if info[-1].strip() == "doctor":
81             doctors.append(Doctor(info[0].strip(), \
82                                   info[1].strip(), info[2].strip()))
83
84     file.close()
85     return doctors
```

*Slika br. 10*

- **search\_doctors (name, surname, id=None)** – Vraća listu tipa Doctor kod kojih je ime i prezime isto kao i prosleđeni parametric name i surname, ukoliko jedno od dva nije navedeno, vratiće se svi doktori sa istim imenom ili prezimenom u zavisnosti šta je prosleđeno. Parametar **id** nije obavezan, ukoliko je naveden, pretraživaće se i po **id**-ju. *Slika br. 11*

```

87 def search_doctors(name, surname, id=None):
88     doctors = get_doctors()
89
90     if len(doctors) == 0:
91         print("\n[-] No doctors in database")
92         return None
93
94     doctor_list = []
95     name = name.strip().lower()
96     surname = surname.strip().lower()
97
98     for dr in doctors:
99         drname = dr.get_name().lower()
100         drsurname = dr.get_surname().lower()
101         if not surname:
102             if name == drname:
103                 doctor_list.append(dr)
104         elif not name:
105             if surname == drsurname:
106                 doctor_list.append(dr)
107         else:
108             if (surname==drsurname) and (name==drname):
109                 doctor_list.append(dr)
110
111     return doctor_list

```

*Slika br. 11*

- **get\_salary(info)** – Vraća doktorovu platu koji je trenutno ulogovan. Kao parameter se prosleđuje informacija o trenutnom korisniku. Metod vraća 0 ukoliko doktor nije imao ni jedan pregled u tekućem mesecu. Plata se računa na osnovu cene pregleda koji je odrađen pomnoženu sa 0,4 *Slika br.11*.
- **print\_salary(info)** – Štampa mesečnu zaradu trenutno ulogovanog doktora. *Slika br.12*

```

113 def get_salary(info):
114     doctor_obj = search_doctors(info['name'], info['surname'], id=info['id'])[0]
115     appointments = appointment.get_appointments_dr_patient(doctor_obj, None, either=True)
116     if not appointments:
117         return 0
118
119     now = datetime.datetime.now()
120     salary = 0
121     for appoint in appointments:
122         if (appoint.get_time().month == now.month) and \
123             (now > appoint.get_time()):
124             salary += appoint.get_price() * 0.4 # nesto mora i ordinaciji da ostane
125
126     return salary
127
128 def print_salary(info):
129     salary = get_salary(info)
130     if salary == 0:
131         print("[-] You have done no work this month")
132     else:
133         print("\n[+] This months salary is: %.2d" % salary)
134     input("\n\nPress Enter to continue...")
135     return

```

*Slika br.12*

- **find\_doctor()** – Vraća objekat tipa Doktor. Metod traži da korisnik unese ime ili prezime doktora i na osnovu toga se radi pretraga. Ukoliko je nađeno više doktora sa istim specifikacijama kao datim, onda se poziva metod choose() is fajla “models.appointment”. *Slika br. 13*

```

137 def find_doctor():
138     drName = input("[?] Enter doctor's name: ")
139     drSurname = input("[?] Enter doctor's surname: ")
140     dr_search = search_doctors(drName, drSurname)
141
142     while len(dr_search) == 0:
143         print("\n[-] Doctor not found")
144         drName = input("[?] Enter doctor's name: ")
145         drSurname = input("[?] Enter doctor's surname: ")
146         dr_search = search_doctors(drName, drSurname)
147
148     if len(dr_search) > 1:
149         doctor = appointment.choose(type='doctors', list=dr_search)
150     else:
151         doctor = dr_search[0]
152
153     return doctor

```

*Slika br.13*

## 6. Fajl “models/patient.py”

U fajlu **models/doctor.py** predstavljena je klasa **Doctor** kao i metode koje se izvršavaju nad objektima ove klase npr. `get_doctors()`.

Importuju se sledeće biblioteke, moduli i klase *Slika br. 14*:

- `os`
- `hashlib`
- `models.person.Person`
- `models.appointment`

```
3  import os
4  import hashlib
5  from models.person import Person
6  from models import appointment
```

*Slika br. 14*

### 6.1. Klasa “models.patient.Patient”

Klasa **Patient** predstavlja klasu pacijenta *Slika br. 15*. Ova klasa nasleđuje klasu **models.person.Person**.

Ova klasa je definisana sledećim poljima:

1. **\_\_name\_\_** – predstavlja privatno polje koje čuva ime pacijenta
2. **\_\_surname\_\_** – predstavlja privatno polje koje čuva prezime pacijenta
3. **\_\_allergies\_\_** – predstavlja privatno polje koje čuva pacijentove alergije
4. **\_\_illness\_\_** – predstavlja privatno polje koje čuva pacijentove hronične bolesti
5. **\_\_contactInfo\_\_** – predstavlja privatno polje koje čuva kontakt pacijenta

Klasa **Patient** ima sledeće metode:

1. **get\_id()** - Vraća korisničko ime korisnika
2. **\_\_str\_\_()** – Redefinisana metoda iz nadklase koja vraća string formata “korisničko\_ime:ime:prezime:doctor”
3. **\_\_equals\_\_(obj)** – Vraća True ako je objekat isti
4. **\_\_hash\_\_()** – Vraća md5 hash objekta
5. **get\_allergies()** – Vraća alergije pacijenta
6. **get\_contactInfo()** – Vraća pacijentov kontakt
7. **get\_illness()** – Vraća pacijentove hronične bolesti

```

8 class Patient(Person):
9     def __init__(self, name, surname, contactInfo, allergies, illness):
10         super().__init__(name, surname)
11         self.__allergies = allergies
12         self.__illness = illness
13         self.__contactInfo = contactInfo
14
15     def __str__(self):
16         return super().__str__()+" "+self.get_allergies()+" "+self.get_illness()
17
18     def get_allergies(self): return self.__allergies
19     def get_illness(self): return self.__illness
20     def get_contactInfo(self): return self.__contactInfo
21
22     def __equals__(self, other):
23         return (self.__class__ == other.__class__) and \
24             (self.get_name() == other.get_name()) and \
25             (self.get_surname() == other.get_surname()) and \
26             (self.get_contactInfo() == other.get_contactInfo()) and \
27             (self.get_allergies() == other.get_allergies()) and \
28             (self.get_illness() == other.get_illness())
29
30     def __hash__(self):
31         info = self.get_name()+self.get_surname()+self.get_contactInfo()
32         return hashlib.md5(info.encode('utf-8')).hexdigest()

```

*Slika br. 15*

## 6.2. Metode u fajlu “models/patient.py”

U fajlu **models/patient.py** implementirane su sledeće metode:

- **search\_patient()** – Traži od korisnika unos imena ili prezimena pacijenta i na osnovu toga u zavisnosti od prosleđenih informacija štampa sve o pacijentu. Ukoliko postoji više pacijenata sa istim informacijama kao prosleđenim, onda se štampaju svi nađeni pacijenti. *Slika br. 16*

```

34 def search_patient():
35     #os.system("clear")
36     print("\n\tSEARCH PATIENT\n")
37
38     name = input("[?] Enter name: ").strip().lower()
39     surname = input("[?] Enter surname: ").strip().lower()
40
41     while not(name or surname):
42         print("\n[-] Please enter something")
43         name = input("[?] Enter name: ").strip().lower()
44         surname = input("[?] Enter surname: ").strip().lower()
45
46     patients = get_patients()
47     foundPatients = []
48
49     for p in patients:
50         pname = p.get_name().lower().strip()
51         psurname = p.get_surname().lower().strip()
52         if not name:
53             if surname == psurname:
54                 foundPatients.append(p)
55         elif not surname:
56             if name == pname:
57                 foundPatients.append(p)
58         else:
59             if (name == pname) and (surname == psurname):
60                 foundPatients.append(p)
61
62     if len(foundPatients) == 0:
63         print("\n[-] Patient {:s} {:s} not found".format(name.title(), surname.title()))
64         input("\n\nPress Enter to continue...")
65         return
66
67     print("\n[*] Found {:d} patients mathing {:s} {:s}".format( \
68         len(foundPatients), name.title(), surname.title()))
69
70     for patient in foundPatients:
71         print("\n\t{:12}  {:>s}".format("Name:", patient.get_name()))
72         print("\t{:12}  {:>s}".format("Surname:", patient.get_surname()))
73         print("\t{:12}  {:>s}".format("Contact:", patient.get_contactInfo()))
74         print("\t{:12}  {:>s}".format("Alergies:", patient.get_alergies()))
75         print("\t{:12}  {:>s}".format("Illnesses:", patient.get_illness()))
76
77     input("\n\nPress Enter to continue...")
78     print()
79     return
80

```

*Slika br. 16*

- **new\_patient()** – Traži da se unesu informacije o pacijentu, pritom proveravajući ispravnost unosa. Ukoliko su informacije ispravne, informacije o korisniku se zapisuju u “database/patients.txt”. *Slika br. 17*



```

81 def new_patient():
82     #os.system("clear")
83     print("\n\tNEW PATIENT\n")
84
85     name = input("[?] Patient name: ").strip().title()
86     while not name:
87         name = input("[?] Enter a valid patient name: ").strip().title()
88
89     surname = input("[?] Patient surname: ").strip().title()
90     while not surname:
91         surname = input("[?] Enter a valid patient surname: ").strip().title()
92
93     contactInfo = input("[?] Patient contact [email or phone]: ").strip()
94     while not contactInfo.strip():
95         contactInfo = input("[?] Please enter contact info [email or phone]: ").strip()
96
97     allergies = input("[?] Patient allergies: ").strip()
98     if not allergies:
99         allergies = "None"
100
101     illness = input("[?] Patient illnesses: ").strip()
102     if not illness:
103         illness = "None"
104
105     info = name+":"+surname+":"+contactInfo+":"+allergies+":"+illness
106     file = open("database/patients.txt", "a")
107     file.write(info+"\n")
108     file.close()
109
110     print("\t[+] Done")
111     input("\nPress Enter to continue...")
112     return

```

*Slika br. 17*

- **get\_patient(hash)** – Vraća objekat tipa Patient na osnovu prosleđenog hash-a.  
*Slika br. 18*

```

114 def get_patient(hash):
115     patients = get_patients()
116
117     for p in patients:
118         if p.__hash__() == hash:
119             return p
120
121     return None

```

*Slika br. 18*

- **search\_patients(name, surname)** - Vraća listu objekata tipa Patient na osnovu prosleđenih parametara name i surname. *Slika br. 19*

```

123 def search_patients(name, surname):
124     patients = get_patients()
125
126     if len(patients) == 0:
127         print("[-] No patients in database")
128         return None
129
130     patient_list = []
131     name = name.strip().lower()
132     surname = surname.strip().lower()
133
134     if not (name.strip() or surname.strip()):
135         return []
136
137     for p in patients:
138         pname = p.get_name().lower()
139         psurname = p.get_surname().lower()
140         if not surname:
141             if name == pname:
142                 patient_list.append(p)
143         elif not name:
144             if surname == psurname:
145                 patient_list.append(p)
146         else:
147             if (surname==psurname) and (name==pname):
148                 patient_list.append(p)
149
150     return patient_list

```

*Slika br. 19*

- **get\_patients()** – Vraća listu objekata tipa Patient. Informacije o pacijentima se učitavaju iz “database/patients.txt” fajla. *Slika br. 20*

```

152 def get_patients():
153     if not os.path.isfile("database/patients.txt"):
154         return None
155
156     patients = []
157     file = open("database/patients.txt", "r")
158     for line in file:
159         info = line.strip(" \n").split(':')
160         patients.append(Patient(info[0], info[1], info[2], info[3], info[4]))
161
162     file.close()
163     return patients

```

*Slika br. 20*

- **find\_patient()** – Vraća objekat tipa Patient na osnovu prodataka koje unosi korisnik. Ukoliko ima više pacijenata sa istim podacima kao prosleđenim, onda se poziva metoda appointment.choose(). *Slika br. 21*

```
165 def find_patient():
166     patientName = input("[?] Enter patient name: ").strip()
167     patientSurname = input("[?] Enter patient surname: ").strip()
168     patient_search = search_patients(patientName, patientSurname)
169
170     while len(patient_search) == 0:
171         print("\n[-] Patient not found")
172         patientName = input("[?] Enter patient name: ")
173         patientSurname = input("[?] Enter patient surname: ")
174         patient_search = search_patients(patientName, patientSurname)
175
176     if len(patient_search) > 1:
177         patient = appointment.choose(list=patient_search)
178     else:
179         patient = patient_search[0]
180         print("\t[*] One patient found: "+patient.get_name()+" "+ \
181             patient.get_surname())
182
183     return patient
```

*Slika br. 21*

## 7. Fajl “models/appointment.py”

U fajlu **models/appointment.py** predstavljena je klasa **Appointment** kao i metode koje se izvršavaju nad objektima ove klase.

Importuju se sledeće biblioteke, moduli i klase *Slika br. 22*:

- os
- datetime
- models
- models.doctor
- models.patient

```
3 import os
4 import datetime
5 import models
6 import models.doctor as doctor
7 from models import patient
```

*Slika br. 22*

### 7.1. Klasa “models.appointment.Appointment”

Klasa **models.appointment.Appointment** predstavlja jedan pregled.

Ova klasa je definisana sledećim poljima:

1. **\_\_time** – predstavlja privatno polje koje čuva datum i vreme pregleda kao objekat klase `datetime.datetime`.
2. **\_\_doctor** – predstavlja privatno polje sa hešom doktora
3. **\_\_patient** – predstavlja privatno polje sa hešom pacijenta
4. **\_\_intervention** – predstavlja privatno polje sa nazivom intervencije
5. **\_\_description** – predstavlja privatno polje sa detaljima pregleda

Klasa **Patient** ima sledeće metode:

- **get\_id()** - Vraća korisničko ime korisnika
- **\_\_str\_\_()** – Redefinisana metoda iz nadklase koja vraća string formata “YYYY-MM-DD H-M:patientID:doctorID:intervencija:detalji”
- **\_\_equals\_\_(obj)** – Vraća True ako je objekat isti
- **get\_patientID()** – Vraća hešovan id pacijenta
- **get\_time()** – Vraća vreme pregleda
- **get\_doctorID()** – Vraća hešovan id doktora
- **get\_intervention()** – Vraća naziv intervencije

- **get\_description()** – Vraća opis pregleda
- **tableRepr()** – Vraća string u tabelarnom formatu
- **get\_price()** – Vraća cenu pregleda, podaci o ceni se učitavaju iz fajla “database/interventions.txt”
- **get\_patient()** – Vraća objekat tipa Pacijent
- **get\_doctor()** – Vraća objekat tipa Doctor
- **set\_description(desc)** – Postavlja prosleđen parametar desc kao detalji pregleda

Kod fajla **appointment.py** dostupan je na internet adresi:

[https://github.com/banebo/Dentistry\\_PMF/blob/master/models/appointment.py](https://github.com/banebo/Dentistry_PMF/blob/master/models/appointment.py)

## 7.2 Metode u fajlu “models/appointment.py”

U fajlu **models/appointment.py** implementirane su sledeće metode:

- **make\_appointment()** – Korisnik unosi informacije o pregledu, metoda proverava validnost informacija tokom unosa. Ukoliko uneto vreme pregleda nije validno ili je odabrani doktor zauzet (ima već zakazan pregled u to vreme) korisnik će morati da opet unese vreme pregleda. Kada su sve informacije ispravne, informacije o pregledu će biti zapisane u fajl “database/appointments.txt” u formatu “YYYY-MM-DD H-M:patientHash:doctorHash:tipIntervencije:opis”
- **choose\_intervention()** – Štampa sve intervencije koje učitava iz fajla “database/intervention.txt” sa njihovom cenom. Korisnik unosi koju intervenciju želi i vraća se korisniku dati pregled i cena.
- **get\_interventions()** – Vraća listu informacija o tipovima pregleda koji se učitavaju iz fajla “database/interventions.txt”
- **print\_interventions()** – Štampa sve intervencije, ovo je pomoćni metod metodi choose\_interventions()
- **choose\_date(doctor)** – Traži se validan unos vremena pregleda, metodu kao parameter prosleđuje objekat tipa Doctor.
- **bad\_time(doctor, time)** – Vraća true ako je dati doktor zauzet u to vreme
- **print\_choices(list)** – Metodi se kao parameter prosleđuje list, na osnovu tipa podataka, štampaju se ili doktori ili bilo koji drugi objekat klase Person.
- **print\_appoint\_choices(list)** – Metodi se kao parameter prosleđuje lista objekata tipa Appointment, podaci se štampaju u tabelarnom formatu
- **get\_appointmentHeader()** – Vraća header za tabelarni prikaz pregleda
- **choose(type=None, list=None)** – U zavisnosti od prosleđenih parametara od korisnika se traži da odabere određene podatke npr. doktora, pacijenta, pregled
- **get\_appointments()** – Vraća listu objekata tipa Appointment. Podaci o pregledu se učitavaju iz fajla “database/appointments.txt”
- **get\_appointments\_dr\_patient(doctor\_obj, patient\_obj, either=False)** – Vraća sve preglede na osnovu prosleđenih parametara doctor\_obj i patient\_obj
- **search\_appointment()** – Traži se od korisnika da unese podatke o doktoru ili pacijentu i na osnovu datih informacija se štampaju pregledi.
- **canStr2Int(c)** – Pomoćni metod koji proverava da li je prosleđeni parameter c ceo broj

- **print\_all\_appointments()** – Štampa sve predstojeće preglede u tabelarnom prikazu
- **get\_indexOf(obj, list)** – Pomoćni metod koji vraća index datog parametra obj u listi list
- **modify\_appointment\_details()** – Traži i modifikuje detalje pregleda

## 8. Fajl “models/admin.py”

U fajlu **models/admin.py** predstavljena je klasa **Admin** koja predstavlja jednog admina i metode koje admin može da izvršava. *Slika br. 23*

Importuju se sledeće biblioteke, moduli i klase:

- `os`
- `getpass`
- `crypt`
- `models.person.Person`

```
1  #!/usr/bin/env python3
2
3  import os
4  import getpass
5  import crypt
6  from models.person import Person
```

*Slika br. 23*

### 8.1 Klasa “models.admin.Admin”

Klasa **Admin** predstavlja klasu administrator, nasleđuje klasu **Person** *Slika br. 24*.

Administrator može da registruje nove zaposlene.

Ova klasa je definisana sledećim poljima:

1. **\_\_name** – ime admina
2. **\_\_surname** – prezime admina

Ova klasa ima sledeće metode:

1. **\_\_str\_\_()** – Vraća string formata “ime:prezime:admin”

```

8  class Admin(Person):
9      def __init__(self, name, surname):
10         super().__init__(name, surname)
11
12     def __str__(self):
13         return super().__str__() + ":admin"

```

*Slika br. 24*

## 8.2 Metode u fajlu “models/admin.py”

U fajlu **models/admin.py** implementirane su sledeće metode:

- **printAdminMenu()** – Štampaju se opcije *Slika br. 25*

```

28  def printAdminMenu(info):
29      print("\nLogged in as " + info['id'])
30
31      print("\n\t[1] Register employee")
32      print("\t[x] Exit")

```

*Slika br. 25*

- **registerPage()** – Metoda za unos novog korisnika tj. zaposlenog. Metod traži da se unesu ime, prezime, korisničko ime, šifra, tip korisnika. Prilikom unosa informacija, metod proverava da li su validne. Korisničko ime mora da bude jedinstveno, i prilikom unosa se proverava da li već postoji. Ukoliko su sve informacije validne, metod štampa liniju formata “korisničkoIme:ime:prezime:tipZaposlenog” u fajl “database/empList.txt”. Informacije o korisničkom imenu i lozinci se smeštaju u fajl “database/login.txt” u formatu “username:šifra”. Radi sigurnosti sistema, šifra se hešuje SHA512 algoritmom za hešovanje pomoću funkcije *crypt.crypt()*.
- **username\_exists(username)** – Pomoćni metod metoda *registerPage()*. Kao parametar prima *korisničko\_ime* i proverava da li korisničko ime već postoji u sistemu. Informacije čita iz fajla “database/empList.txt”. *Slika br. 26*



```

76 def username_exists(username):
77     if not os.path.isfile("database/emplist.txt"):
78         return False
79
80     file = open("database/emplist.txt", "r")
81     for line in file:
82         if line.split(":")[0].strip() == username:
83             file.close()
84             return True
85
86     return False

```

*Slika br. 26*

- **change\_password(info)** – Ovaj metod kao parametar prima rečnik sa informacijama o korisniku. Prvo se učitaju sve informacije o korisnicima i njihovim (hešovanim) šiframa. Potom se traži od korisnika da unese trenutnu šifru, jer korisnik može da menja isključivo sopstvenu šifru. Ukoliko je uneo tačnu šifru, traži se da unese novu šifru. Nakon validacije, šifra se enkriptuje SHA512 algoritmom za hešovanje i informacije se štampaju u fajl “database/login.txt” u formatu “username:password”.

## 9. Glavni fajl “Dentistry.py”

Fajl “Dentistry.py” predstavlja glavni program. Program samo traži da se korisnik uloguje na sistem i na osnovu toga se pozivaju određene funkcije.

Skripta importuje sledeće klase, objekte, module *Slika br. 27*:

- *getpass*
- *os*
- *sys*
- *crypt*
- *models.patient*
- *models.nurse*
- *models.doctor*
- *models.admin*
- *models.appointment*

```
3 import getpass
4 import os
5 import sys
6 import crypt
7 import models.patient as patient
8 import models.nurse as nurse
9 import models.doctor as doctor
10 import models.admin as admin
11 import models.appointment as appointment
```

*Slika br. 27.*

U glavnom fajlu su implementirane sledeće metode:

- *loginPage()* – Glavna stranica programa, traži da korisnik unese korisničko ime i lozinku. Dokle god informacije nisu ispravne, proces upita se ponavlja. Kada se unesu ispravne informacije, metod vraća informacije o korisniku u rečniku.
- *login(username, password)* – Metod koji proverava validnost informacija. Kao parametar se prosleđuju *korisničko ime* i *lozinka*. Informacije se učitavaju iz fajla “database/login.txt”. *Slika br. 28*

```

29 def login(username, password):
30
31     if not (username.strip() and password):
32         return False
33
34     if not os.path.isfile("database/login.txt"):
35         print("[-] Login file is missing, can't login.")
36         exit(1)
37
38     file = open("database/login.txt", "r")
39     c = 0
40     for line in file:
41         c += 1
42         info = line.strip("\n").split(":")
43         if len(info) != 2:
44             continue
45
46         if info[0] == username:
47             if crypt.crypt(password, info[1]) == info[1]:
48                 return True
49
50     file.close()
51     return False

```

Slika br. 28.

- *getInfo(username)* – Traži korisnikove informacije Slika br. 29. Kao parameter se prosleđuje *korisničko ime*. Informacije o postojanju korisnika se učitavaju iz fajla “database/empList.txt”. Ukoliko korisnik sa datim korisničkim imenom postoji, metod vraća rečnik u formatu [*‘id’:username* , *‘name’:ime*, *‘prezime’:prezime*, *‘empType’:tipKorisnika*], u suprotnom metod vraća *None*.

```

53 def getInfo(username):
54     if not username.strip():
55         return None
56
57     if not os.path.isfile("database/empList.txt"):
58         print("[-] Cant fetch data, file missing!")
59         return
60
61     if username == "admin":
62         return {'id': 'admin'}
63
64     file = open("database/empList.txt", "r")
65     for line in file:
66         if line.split(":")[0] == username:
67             info = line.split(":")
68             if len(info) == 4:
69                 return {'id': info[0], 'name': info[1], 'surname': info[2], \
70                     'empType': info[3].strip("\n")}
71
72     file.close()
73     return None

```

Slika br. 29

- *main()* – Glavni metod programa. Poziva se funkcija *loginPage()*. Na osnovu informacija koje vrati metod *loginPage()* korisnik određenog tipa se prijavljuje na sistem. Ukoliko *loginPage()* vrati *None* onda se proces ponavlja *Slika br. 30*.

```
75 def main():
76     os.system("clear")
77
78     while True:
79         info = loginPage()
80         if info == None:
81             continue
82
83         if info['id'] == "admin":
84             admin.admin_menu(info)
85         if info['empType'] == "doctor":
86             doctor.dr_menu(info)
87         elif info['empType'] == "nurse":
88             nurse.nurse_menu(info)
89         else:
90             print("[-] Something went wrong")
91             exit(1)
92
93     exit(0)
```

*Slika br. 30*

## 10. Direktorijum “database/”

Direktorijum “database/” predstavlja bazu podataka. U ovom direktorijumu smešteni su sledeći fajlovi:

- *appointment.txt* – Čuva informacije o svim pregledima
- *empList.txt* – Čuva informacije o korisnicima
- *interventions.txt* – Čuva informacije o intervencijama
- *login.txt* – Čuva informacije o korisnicima i šiframa
- *patients.txt* – Čuva informacije o pacijentima

### 10.1 Fajl “database/login.txt”

U ovom fajlu se čuvaju informacije o korisničkim imenima i njihovim šiframa.

Linije u fajlu su formatirane na sledeći način *Slika br. 31*:

*username : hashed\_password*

```
1 admin:$6$MG6tUHGGCAws.Kn3$5/.oYjCFvEe9eehN1fOMBK1XmqmIyg/UF8PoS8rCJ0dgWgGA0h2CIeZnEtjLkD4PWYnX4xR1ytf/1EbGzJ03j.  
2 dr:$6$sPsIFqrHSsfB14nc$2bv49tPemXJYtSEM79.wTpVL01ki9ZbPW5WXvNqY06NUyBGyP5SfIMpjxdYrvWJXEZiVRkxGZowpsXaFfMj6/  
3 nurse:$6$MmS83nXongEcGTcN$1zn3M1P.64q9up01rYlUmqFyKJf/qr.10VxCI8AktSC3WZQpmhC9DQ9ILh.ixR931bfnQPbOMfdGrz9JBI14v.  
4 dr2:$6$x071eer7gKaTqLbM$quCUXTBuT4x85qZQZqbFvYfHA0BzyFYC3c2GocmeS9KgpEK41FdaddRL8F7I.lyN.eCBdV/jbtmbJEQ3qUprG0
```

*Slika br. 31*

### 10.2 Fajl “database/empList.txt”

U ovom fajlu se čuvaju informacije o zaposlenima.

Linije u fajlu su formatirane na sledeći način *Slika br. 32*:

*user\_id : name : surname : employee\_type*

```
1 nurse:Andjelka:Bozic:nurse  
2 dr2:Milutin:Strahinjic:doctor
```

*Slika br. 32*

### 10.3. Fajl “database/interventions.txt”

U ovom fajlu se čuvaju informacije o intervencijama.

Linije u fajlu su formatirane na sledeći način *Slika br. 33*:

*intervention\_name : intervention\_price*

```
1  Ciscenje kamenca:500
2  Plomba:800
3  Vadjenje zuba:1200
4  Beljenje zuba:1500
5  Fasete:2000
6  Ugradjivanje implantata:3000
7  Fiksna proteza:1500
8  Lecenje zuba:2000
9  Apikotomija zuba:1800
10 Pregled:500
```

*Slika br. 33*

### 10.4. Fajl “database/patients.txt”

U ovom fajlu se čuvaju informacije o pacijentu.

Linije su formatirane na sledeći način *Slika br. 34*:

*ime : prezime : kontakt : alergije : bolesti*

```
1  P:P:p:adgasdg:adsg:llsls
2  P:Q:male:p@q:no:no
3  Milutin:Q:ttt:None:None
4  Bogdan:Mikovic:13412:None:None
```

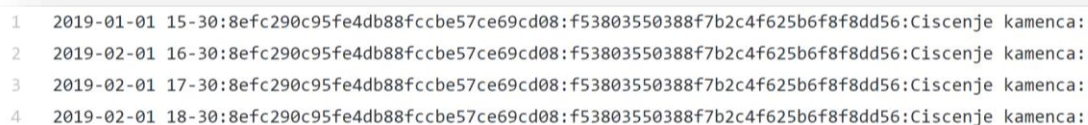
*Slika br. 34*

## 10.5. Fajl “database/appointments.txt”

U ovom fajlu se čuvaju informacije o svim pregledima.

Linije u fajlu su formatirane na sledeći način *Slika br. 35*:

*YYYY-DD-MM H-M : patient\_hash : doctor\_hash : appointment\_type : details*



```
1 2019-01-01 15-30:8efc290c95fe4db88fccbe57ce69cd08:f53803550388f7b2c4f625b6f8f8dd56:Ciscenje kamenca:
2 2019-02-01 16-30:8efc290c95fe4db88fccbe57ce69cd08:f53803550388f7b2c4f625b6f8f8dd56:Ciscenje kamenca:
3 2019-02-01 17-30:8efc290c95fe4db88fccbe57ce69cd08:f53803550388f7b2c4f625b6f8f8dd56:Ciscenje kamenca:
4 2019-02-01 18-30:8efc290c95fe4db88fccbe57ce69cd08:f53803550388f7b2c4f625b6f8f8dd56:Ciscenje kamenca:
```

*Slika br. 35.*

## **11. Zaključak**

Program Dentistry je veoma koristan i jednostavan program za korišćenje. Ovaj program omogućava doktorima i stomatološkim sestrama da lakše unose sve podatke o pacijentima i uslugama a takođe im omogućava i da na jednostavan način dođu do podataka koji su im u datom trenutku neophodni, na primer: da zakažu pregled, da pretražuju preglede pacijenata, da vide sve predstojeće preglede, da unesu podatke za novog pacijenta, itd.



## 12. Literatura

- 1) <https://docs.python.org/3/>
- 2) <https://docs.python.org/3/library/hashlib.html?highlight=hashlib#module-hashlib>
- 3) <https://docs.python.org/3/library/os.html?highlight=os#module-os>
- 4) <https://docs.python.org/3/library/crypt.html?highlight=crypt#module-crypt>
- 5) <https://docs.python.org/3/library/getpass.html?highlight=getpass#module-getpass>
- 6) <https://docs.python.org/3/library/datetime.html?highlight=datetime#module-datetime>