Web aplikacija za udomljavanje životinja

Lešković, Filip

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni prijediplomski studij Računarstvo

WEB APLIKACIJA ZA UDOMLJAVANJE ŽIVOTINJA

Završni rad

Filip Lešković

Osijek, 2024 godina.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Filip Lešković
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R4670, 28.07.2021.
JMBAG:	0165091436
Mentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za udomljavanje životinja
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	[Rezervirano: Filip Lešković] Potrebno je napraviti aplikaciju koja će omogućiti lakše i jednostavnije udomljavanje nezbrinutih životinja. Aplikacija bi imala korisničko i administratorsko sučelje. Korisnik bi mogao pregledavati i dati ponudu za udomljavanje neke životinje. Adminstartor bi mogao upravljati s podacima o životinjama za udomljavanje koje bi trebale biti kategorizirane i lako pretražive korisnicima. Administartor bi isto tako prihvaćao ponude za udomljavanje, te bi imao i uvid u arhivu udomljenih životinja.

Datum prijedloga ocjene završnog rada od strane mentora:	16.09.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	25.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	27.09.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 27.09.2024.

Ime i prezime Pristupnika:	Filip Lešković
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R4670, 28.07.2021.
Turnitin podudaranje [%]:	10

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za udomljavanje životinja**

izrađen pod vodstvom mentora izv. prof. dr. sc. Alfonzo Baumgartner

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD

U današnjoj svakodnevnici mogu se pronaći brojne životinje koje završe ostavljene u šumama, poljima i ulicama grada. Takve životinje mogu predstavljati opasnost jer mogu ugrožavati sigurnost prometa i ostalih sudionika. Nažalost te životinje prepuštene su same sebi i na razne načine preživljavaju u okolini u kojoj su ostavljene. Zbog toga postoje lokalna skloništa koja smjeste napuštene životinje u svoje prostorije sve dok one ne pronađu novog vlasnika. Neke životinje imaju poteškoća s pronalaskom novog doma zbog svog izgleda, ponašanja ili nekih drugih čimbenika koji mogu utjecaj na mišljenje udomitelja. Neke će životinje zbog svoje veličine teže naći novi dom zbog toga što manji broj ljudi ima vlastito dvorište, dok će neke starije životinje teže naći novi dom jer su već stare i teže ih je naučiti nešto novo, a ostalo im je manje života od mlađih životinja. Iako su životinje različite svaka na svoj način jedno im je zajedničko, a to da su i one živa bića i zaslužuju imat ljubav i pažnju od svojih vlasnika. U ovom radu napravljena je aplikacija udomljavanje životinja te objašnjeno njeno funkcioniranje. Nakon prvog dijela, uvoda, u drugom dijelu prikazana su već postojeća rješenja ovog problema. U trećem dijelu objašnjene su tehnologije korištene prilikom izrade aplikacije te njen izgled i struktura. U četvrtom dijelu objašnjene su sve korisničke i administratorske funkcije unutar aplikacije uz prikaz koda i slika aplikacije. Zaključak je napisan u petom poglavlju.

1.1. Zadatak završnog rada

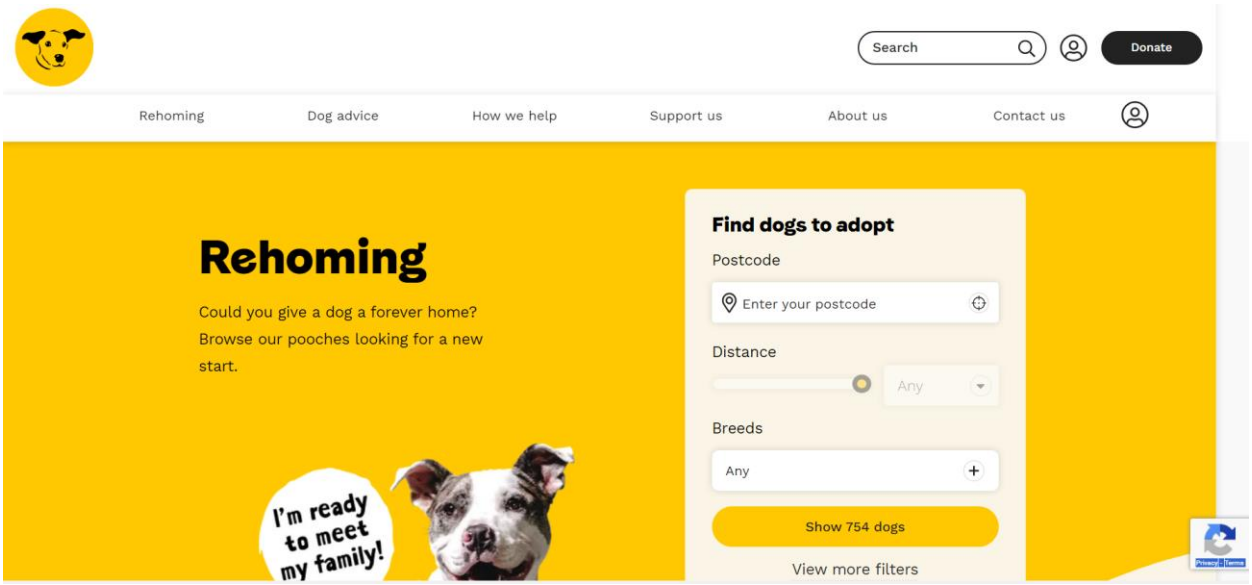
Potrebno je napraviti aplikaciju koja će omogućiti lakše i jednostavnije udomljavanje nezbrinutih životinja. Aplikacija bi imala korisničko i administratorsko sučelje. Korisnik bi mogao pregledavati i dati ponudu za udomljavanje neke životinje. Administrator bi mogao upravljati s podatcima o životinjama za udomljavanje koje bi trebale biti kategorizirane i lako pretraživane korisnicima. Administrator bi isto tako prihvaćao ponude za udomljavanje, te bi imao i uvid u arhivu udomljenih životinja.

2. POSTOJEĆA RJEŠENJA

U ovom poglavlju opisana su i prikazana već neka postojeća rješenja vezana uz proces udomljavanja životinja.

2.1. DogsTrust

Prema [1], Lady Gertrude Stock 1981. osnovala je projekt DogsTrust koji se danas sastoji od velikog broja stručnog osoblja i volontera. Bave se udomljavanjem nezbrinutih pasa te pomažu vlasnicima pri čuvanju već udomljenih. Kako bi se udomio pas, potrebno je prvo napraviti korisnički račun i poslati otvorenu zamolbu za udomljavanje. Sljedeći korak je opisati kakvu vrstu psa korisnik želi udomiti jer se ne može na točno određenog psa poslati otvorena molba. Zatim osoblje traži korisniku određenog psa sve dok se ne ispune njegovi zahtjevi. Kada korisnik bude zadovoljan preporukom, dobiva poziv za upoznavanje te tamo donosi konačnu odluku. Početna stranica prikazana je na slici 2.1.

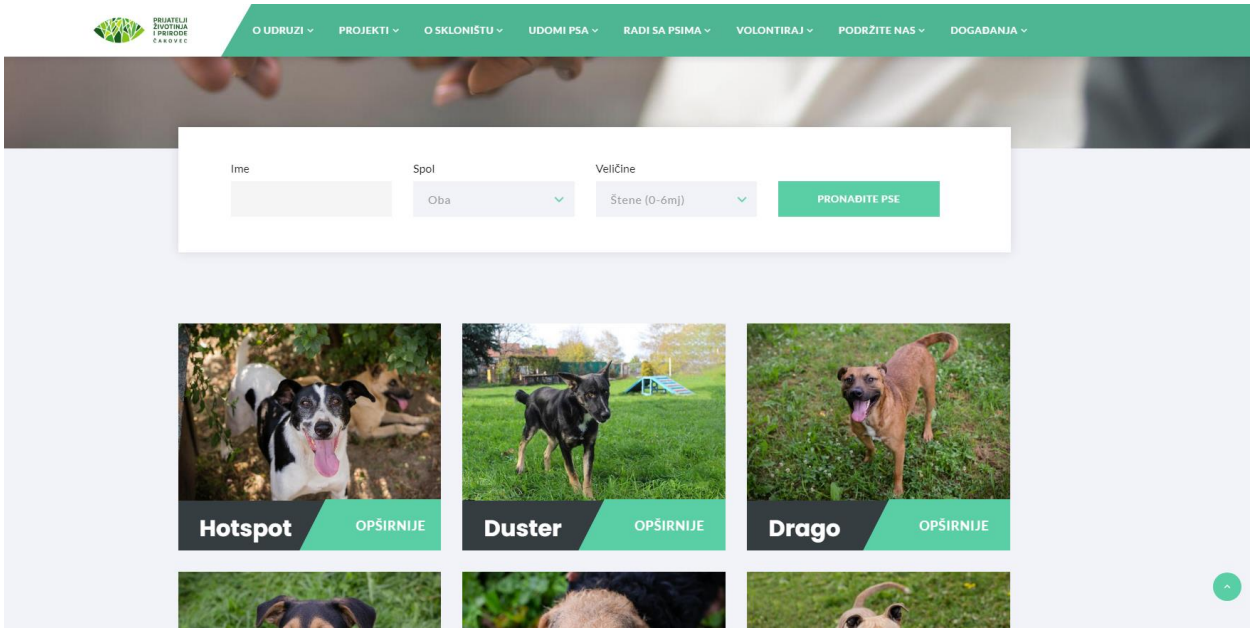


Sl. 2.1. DogsTrust početna stranica

2.2. ZEU Prijatelji životinja i prirode, Čakovec

Udruga je osnovana 2004. kako bi organizirala edukacije zaštite okoliša i životinja te njihovu promidžbu. Od 2012. godine preuzima najveće hrvatsko sklonište za napuštene životinje čiji je i

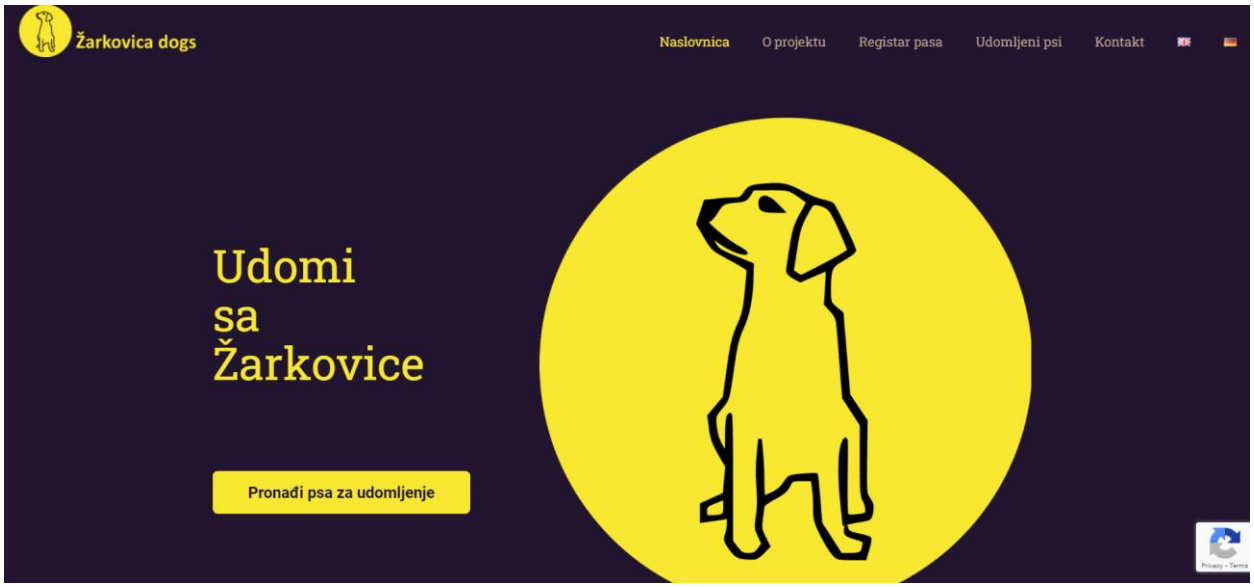
dan danas vlasnik. Djeluje na području Međimurske županije. U procesu udomljavanja udomitelj vodi razgovor s osobljem kakvog točno psa želi udomiti. Potom osoblje dolazi u kuću udomitelja te provjerava uvjete u kojima će živjeti pas. Na kraju se potpisuje ugovor o udomljavanju te pas pronalazi svog novog vlasnika [2]. Početna stranica prikazana je na slici 2.2.



Sl. 2.2. ZEU Prijatelj životinja i prirode početna stranica

2.3. Žarkovica Dogs

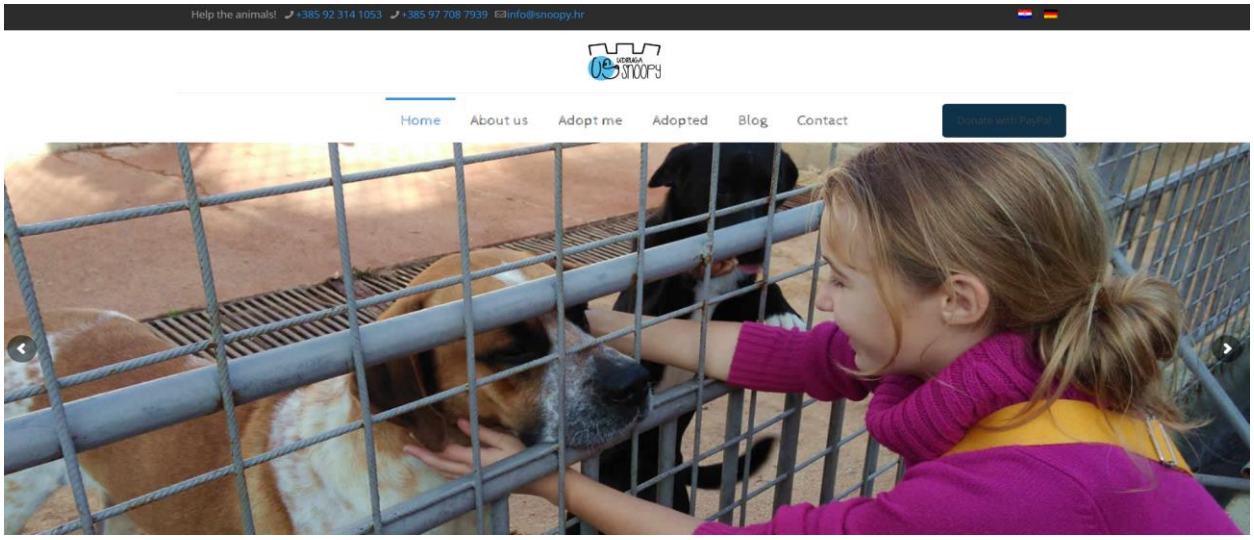
Prema [3], Žarkovica Dogs neregistrirano je sklonište za pse koje ima u cilju pronaći utočište za sve neudomljene pse. Nositelj projekta je Grad Dubrovnik, dok u projektu sudjeluju i Udruga Pobjede i ZEU Prijatelji prirode i životinja. Udomljavanje funkcionira na način da udomitelj prvo pronađe psa kojeg želi udomiti te potom poslati upit za udomljavanje istog. Na upit odgovara skloništa u kojem se pas trenutno nalazi. Razlog tome je što ovaj projekt zastupaju i druga skloništa koja su u suradnji s njima kako bi što prije našli domove psima. Početna stranica prikazana je na slici 2.3.



Sl. 2.3. ZEU Žarkovica dogs početna stranica

2.4. Snoopy Pula

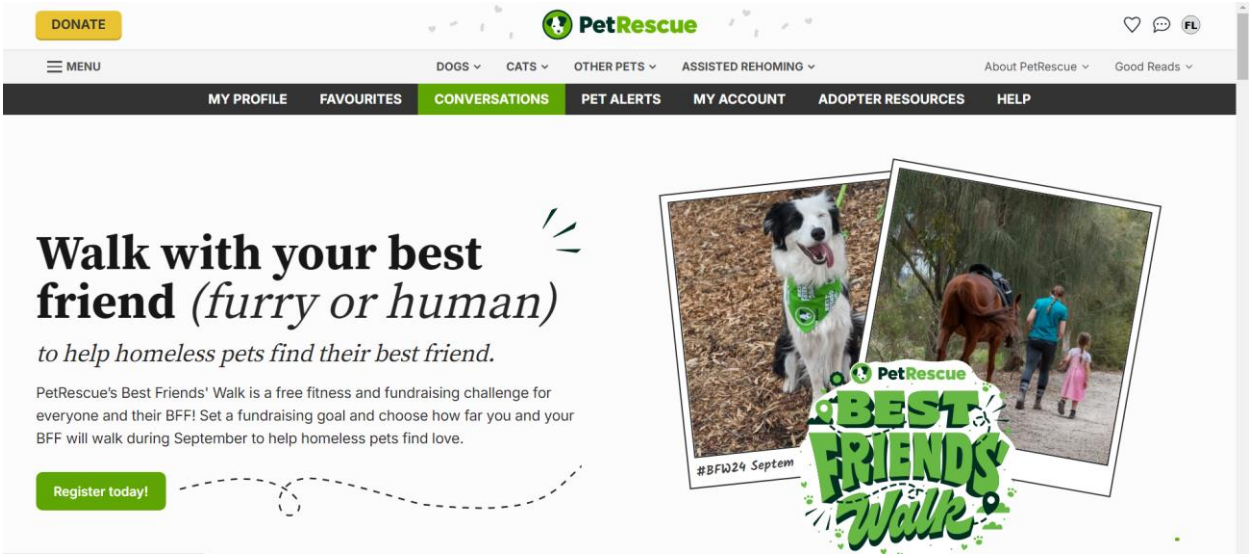
Privremeno sklonište „Snoopy“ nalazi se na području austrougarske utvrde Pomer u blizini gradskog odlagališta (Kaštijun). Trenutno se u skloništu nalazi osamdesetak napuštenih pasa o kojima brinu volonteri tako što im donose svježu vodu i osiguravaju hranu. Proces udomljavanja se odvija na način da se udomitelj ispunjava obrazac te dobiva psa na probno razdoblje od tjedan dana. U tih tjedan dana dolaze volonteri kako bi provjerili uvjete u kojima pas živi. Ako udomitelj bude zadovoljan sa psom, potpisuje ugovor te udomljuje istog [4]. Početna stranica prikazana je na slici 2.4.



Sl. 2.4. Snoopy Pula početna stranica

2.5. PetRescue

PetRescue nastala je 2004. godine kao prva nacionalna web stranica za usvajanje kućnih ljubimaca. Pomoću njihovih programa, danas pomažu milijunima Australaca udomiti neke od svojih ljubimaca u skloništima. Kako bi se udomili s ove web stranice, prema [5], potrebno je prvo napraviti račun. Potom se šalje otvorena zamolba za udomljavanje životinje tako što se u njoj napiše zašto bi baš htjeli tu životinju udomiti. Nakon toga administratori provjeravaju zamolbu i šalju daljnje upute za usvajanje. Početna stranica prikazana je na slici 2.5.



Sl. 2.5. PetRescue početna stranica

3. KORIŠTENE TEHNOLOGIJE I IZRADA WEB APLIKACIJE

U ovom poglavlju navedene su i objašnjene sve tehnologije koje su korištene prilikom izrade aplikacije te je prikazano njen izgled i struktura.

3.1. HTML

HTML (engl. *Hypertext Markup Langage*) jezik je za označavanje hipertekstualnih elemenata pomoću kojih su strukturirane web stranice. Prema [6], *hypertext* odnosi se na poveznice koje međusobno povezuju web stranice. Web stranice sastoje se od HTML oznaka (engl. *tags*) koje definiraju gdje će podatci biti prikazani unutar nje same. Oznake su odjeljenje od ostatka teksta pomoću znakova < i > tako što se oznaka napiše unutar njih.

3.2. CSS

Prema [7], CSS (engl. *Cascading Style Sheet*) je jezik koji opisuje način na koji će se prezentirati HTML elementi. CSS sintaksa sastoji se od selektora i deklaracije. Selektori označuju elemente unutar HTML dokumenta koji se oblikuju, dok se pomoću deklaracije definira na koji način će elementi biti oblikovani. Deklaracija se sastoji od parova svojstvo-vrijednost te se unutar jednog selektora može nalaziti više deklaracija. CSS se može primijeniti na HTML element pomoću atributa *style*, u oznaci <*Head*> pomoću oznake <*Link*>, koja sadrži putanju do *.css* datoteke, i unutar oznake <*Style*> koja se nalazi u oznaci <*Head*> gdje se definiraju selektori koji će se odnositi na elemente unutar te HTML datoteke.

3.3. JavaScript

JavaScript je lagan (engl. *lightweight),* jednonitni (engl. *single-threaded*) programski jezik koji se može primijeniti na više platformi. Poznat je kao i skriptni jezik jer se koristi za dodavanje dinamičkih interakcija na web stranice, a uz HTML i CSS jedan je od glavnih jezika za web programiranje. Može se primjenjivati na klijentskoj i na poslužiteljskoj strani. Na klijentskoj strani služi za dostavljanje objekata za kontrolu preglednika te omogućavanje reagiranja elemenata na korisnikove događaje, dok na poslužiteljskoj strani omogućuje komuniciranje aplikacije s bazom podataka, kontinuirani tok podatka unutar aplikacije te manipuliranje datotekama na poslužitelju.

3.4. Bootstrap

Prema [8], *bootstrap* je besplatan *front-end* razvojni okvir (engl. *framework*) otvorenog koda (engl. *open source*). Svrha ovog razvojnog okvira je lakše dizajniranje responzivnih web stranica i aplikacija pomoću već gotovih dizajn predložaka za brojne HTML elemente. Osim HTML-a, temeljen je i na CSS-u i JavaScript programskom jeziku.

3.5. Django

Django je besplatan *Python* razvojni okvir (engl. *framework*) otvorenog koda i visoke razine namijenjen za brz razvoj te čist i pragmatičan dizajn. Ima uspješnu i aktivnu zajednicu, dobru dokumentaciju te besplatnu i plaćenu podršku za korisnike. Prema [9], vrlo je svestran, što omogućuje implementiranje raznih vrsta web aplikacija. Osim toga posjeduje veliku količinu sigurnosnih elemenata koje rješavaju učestale i tipične pogreške pomoću već integriranih procesa. Također uključuje zaštitu protiv raznih opasnosti za aplikacije, poput SQL (engl. *Structured Query Language*) ubrizgavanja (engl. SQL *injection*) , *clickjackinga* i drugih. Skalabilnost je također jedna od prednosti ovog razvojnog okvira jer omogućuje neovisnost svakog dijela arhitekture od drugih, što daje pojednostavljuje programeru zamjenu ili promjenu istih. Održavanje je isto tako važna karakteristika *Djanga* jer poštuju brojne dizajn uzorke i principe (engl. *design patterns and principles*) poput DRY (engl. *Don't repeat yourself*) koji, prema [10], zagovara smanjenje dupliciranog koda unutar programa.

3.6. Izrada web aplikacije

3.6.1. Baza podataka

Django koristi ORM (engl. *Object-Relational Mapping*) tehniku koja omogućuje interakciju s bazom podataka pomoću objekata. Nema pisanja SQL upita već se komunikacija odvija preko *python* koda. Tablice unutar baze predstavljeni su kao modeli (klase), dok je svaki redak unutar tablice instanca te klase. Modeli aplikacije nalaze se u datoteci *models.py* u kojoj se nalaze 4 modela: *User, Animal, AnimalType* i *AdoptingRequest*. *User*, prikazan slikom 3.1, predstavlja korisnički račun koji se koristi za podnošenje zahtjeva za udomljavanje životinje. Životinje se spremaju u bazi pod model *Animal*, a u njemu se nalazi model *AnimalType* koji služi dodatnu specifikaciju i lakšu kategorizaciju životinja, prikazano na slici 3.2. *User* šalje zahtjev za udomljavanje životinja koji se sprema u bazi pod model *AdoptinqRequest*, prikazano na slici 3.3.

Kako bi se kreirale i ažurirale tablice unutar baze podataka, potrebno je u konzolu napisati naredbu *python manage.py migrate*, što omogućuje sinkronizaciju modela unutar programa i tablica unutar baze podataka.

```
class User(AbstractBaseUser, PermissionsMixin):
    email = models.EmailField(unique=True)
    first_name = models.CharField(max_length=64)
    last_name = models.CharField(max_length=64)
    oib = models.CharField(max_length=11, null=False, blank=False)
    address = models.CharField(max_length=128, null=False, blank=False)
    city = models.CharField(max_length=64, null=False, blank=False)
    date_of_birth = models.DateField(null=True, blank=True)
    gender = models.CharField(max_length=12, null=True, blank=True)
    avatar_url = models.CharField(max_length=128,
                                  default="https://www.shutterstock.com/image-vector/default-avatar-profile-icon-social-600nw-1677509740

    is_staff = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)

    objects = CustomUserManager()

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['first_name', 'last_name', 'oib', 'address', 'city']

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

Sl. 3.1. Model *User*

```
class AnimalType(models.Model):
    CAT = "CAT"
    DOG = "DOG"
    TYPE_CHOICES = [
        (CAT, "Cat"),
        (DOG, "Dog"),

    ]
    lifespan=models.CharField(max_length=128)
    type=models.CharField(max_length=64,choices=TYPE_CHOICES)
    breed=models.CharField(max_length=64)
    def __str__(self) -> str:
        return f"{self.breed}"

class Animal(models.Model):
    name = models.CharField(max_length=128)
    age= models.SmallIntegerField(default=5)
    description = models.CharField(max_length=1024)
    is_adopted = models.BooleanField(default=False)
    type= models.ForeignKey(AnimalType,on_delete=models.CASCADE)
    height=models.CharField(max_length=64)
    weight=models.CharField(max_length=64)
    avatar_url=models.CharField(max_length=128, null=True)
    gender=models.CharField(max_length=12,null=True,blank=True)
    def __str__(self) -> str:
        return f"{self.name}-{self.type.breed}"
```

Sl. 3.2. Modeli *Animal* i *AnimalType*

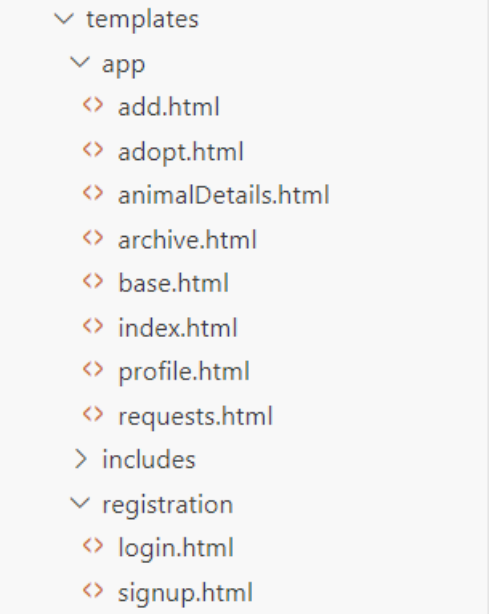
```python
class AdoptingRequest(models.Model):
    PENDING = "PENDING"
    APPROVED = "APPROVED"
    DECLINED = "DECLINED"

    STATUS_CHOICES = [
        (PENDING, "Pending"),
        (APPROVED, "Approved"),
        (DECLINED, "Declined"),
    ]
    user_id=models.ForeignKey(User,on_delete=models.CASCADE)
    animal_id=models.ForeignKey(Animal,on_delete=models.CASCADE)
    created_at= models.DateTimeField(default=timezone.now)
    status = models.CharField(max_length=10, choices=STATUS_CHOICES)
    description= models.CharField(max_length=3000,null=True,blank=True)
```

Sl. 3.3. Model *AdoptingRequest*

3.6.2. Izgled i struktura aplikacije

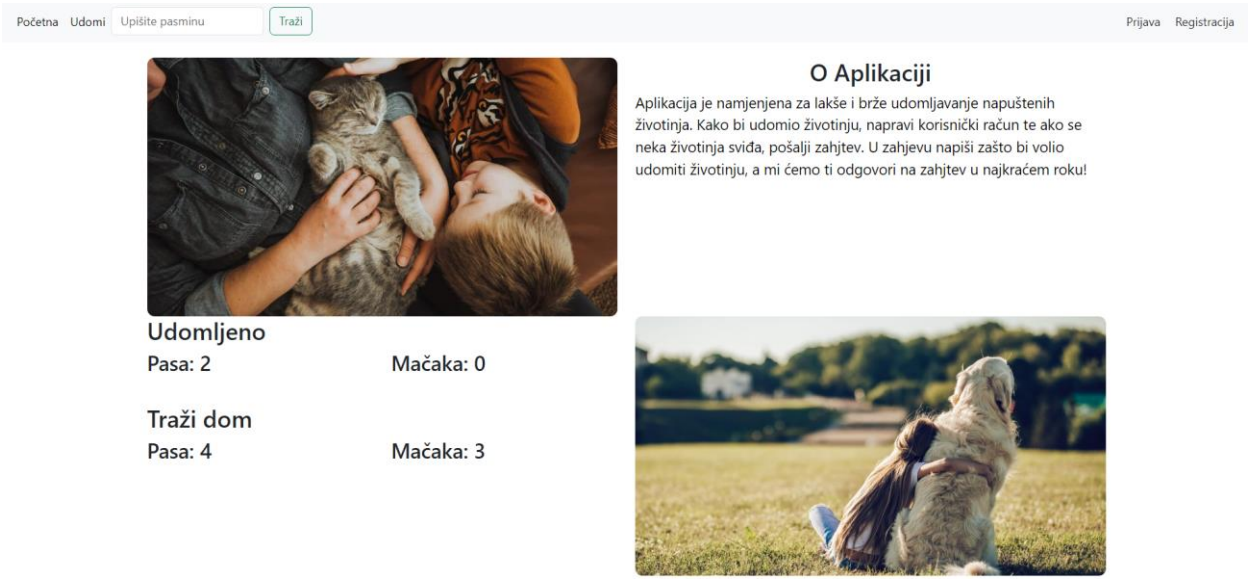
Aplikacija posjeduje više HTML dokumenata, što je i prikazano na slici 3.4.



Sl. 3.4. HTML datoteke unutar direktorija *templates*

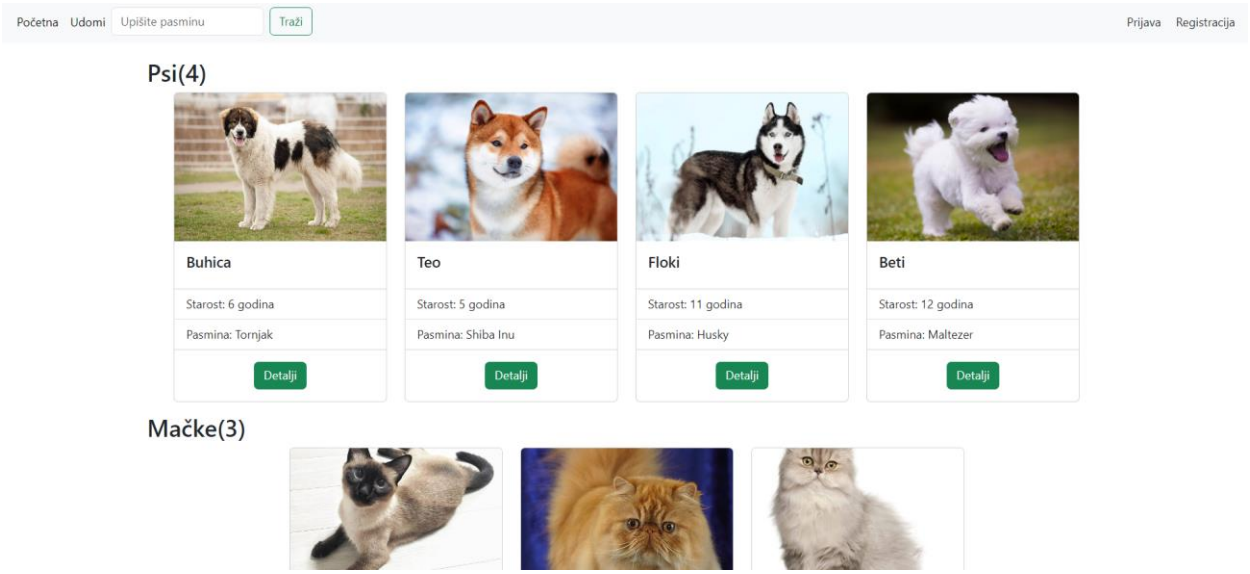
Neke od tih HTML dokumenata nije moguće pristupiti u aplikaciji kao obični korisnik, poput *archive.html* i *requests.html*, već se moraju posjedovati administratorske ovlasti. Preko URL-a (engl. *Uniform Resource Locator*) je moguće pristupiti stranicama, ali na njima ništa nije prikazano korisniku zbog uvjetovanja u kodu da je sadržaj vidljiv samo administratoru. Osim toga unutar aplikacije se nalazi i *bs_form.html* datoteka koja služi za prikazivanje obrazaca unutar *login.html*, *add.html* i *signup.html* dokumenta. *Base.html* je datoteka koja zapravo predstavlja kostur cijele aplikacije. Unutar nje se nalazi oznaka <*Head*> koja posjeduje sve važne skripte i linkove za funkcioniranje cijele aplikacije. Također unutar *base.html* dokumenta nalazi se i navigacijska traka koja se prikazuje na svim dijelovima aplikacije. To se postiže s naredbom *extends* tako što se ona navede u svaki preostali HTML dokument. Pomoću toga svaki HTML dokument poprima *base.html* čime se poštuje DRY načelo gdje se izbjegava dupliciranje koda.

Na početnoj stranici, *index.html*, prikazane su neke osnovne informacije o aplikaciji (kako se ona koristi i podatke o životinjama). Početna stranica prikazana je na slici 3.5.



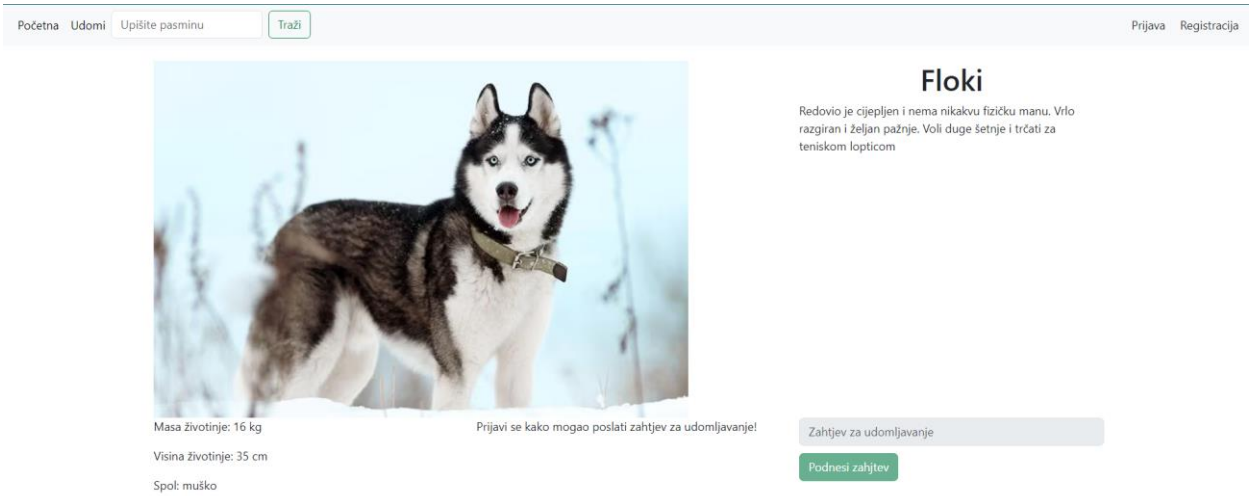
Sl. 3.5. Početna stranica

Pritiskom na poveznicu Udomi na navigacijskoj traci, otvara se HTML dokument *adopt.html* na kojem su prikazane sve dostupne životinje za udomljavanje, što je i prikazano na slici 3.6.



Sl. 3.6. Udomi stranica

Kako bi se pristupilo detaljima životinje, potrebno je kliknuti na dugme *Detalji*. Unutar njega nalaze se sve važne informacije u vezi životinje i mogućnost slanja zahtjeva za udomljavanje. Detalji životinje prikazani su na slici 3.7. Vrlo važna stavka je da dugme *Podnesi zahtjev* ne može biti pritisnuto jer korisnik nije prijavljen u aplikaciju.



Sl. 3.7. Stranica s detaljima životinje

Sve stranice unutar aplikacije povezane su unutar jedne datoteke imena *urls.py*. Unutar nje definirane su sve rute koje su korištene u aplikaciji, prikazano na slici 3.7.

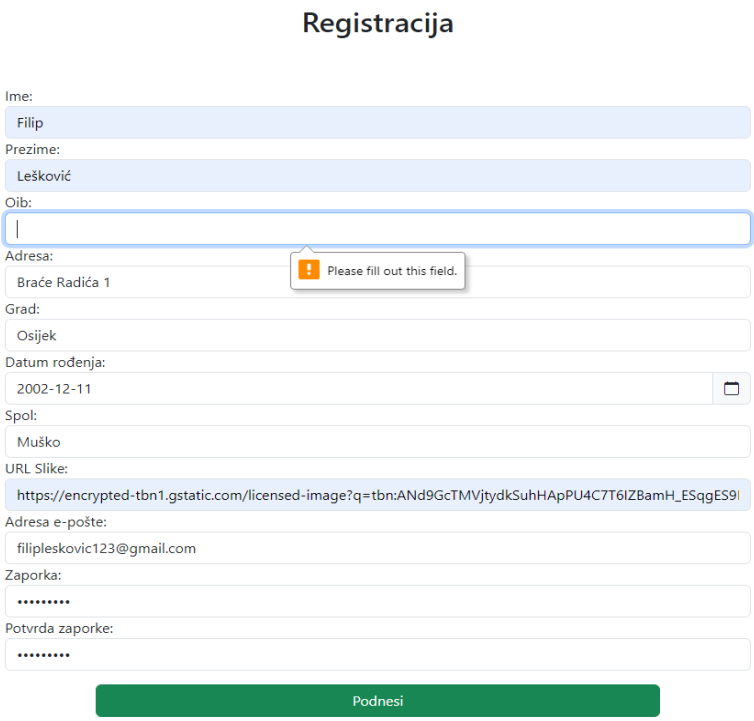
```python
from django.urls import path
from . import views
app_name = "app"
urlpatterns = [
    path("", views.index, name="index"),
    path("adopt/",views.adopt,name="adopt"),
    path("<int:animal_id>/", views.animalDetails, name="animalDetails"),
    path("<int:animal_id>/request/new/", views.new_request, name="new_request"),
    path('addAnimal/', views.addAnimal, name="addAnimal"),
    path('addBreed/',views.addBreed,name='addBreed'),
    path("request",views.requests,name="requests"),
    path('archive',views.archive,name="archive"),
    path("<int:req_id>/request/decision/",views.request_decision,name="request_decision"),
    path("searchAnimals/",views.searchBreeds,name="searchAnimals"),
    path("<int:user_id>/profile/",views.profile,name="profile"),
    path('sign-up', views.SignUpView.as_view(), name = 'signup'),
    path('logout', views.logout_view, name='logout')
]
```

Sl. 3.7. urls.py datoteka

Prema [11], svaka ruta sastoji se od dijela URL-a koji prikazuje stranicu (ako se URL sastoji od dinamičkih podataka oni se upisuju unutar < >), metode unutar *views,py* datoteke, koja obrađuje i vraća odgovor na zahtjev, i imena koji služi za označavanje rute koja se koristiti unutar koda u HTML datoteci.

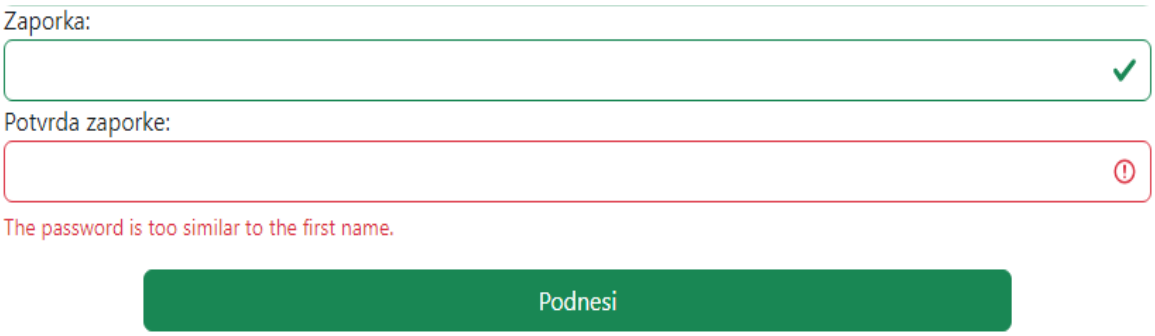
3.6.3. Autentikacija

Postoje dvije vrste korisnika: administrator, koji kontrolira cijelu aplikaciju, donosi odluke o podnesenim zahtjevima te dodaje nove životinje i vrste pasmina, i običan korisnik koji može pregledavati životinje i slati zahtjeve za udomljavanje. Ako jedno polje nije popunjeno, aplikacija označi to polje te registracija neće biti izvršena sve dok ono ne bude popunjeno, slika 3.8. Također se zaporke moraju podudarati i poštovati sljedeća pravila: moraju sadržavati sve vrste znakova (numerički, slova i specijalni znakovi), minimalna duljina zaporke mora biti 8 znakova, zaporka ne smije sadržavati isključivo brojeve ili osobni podatak poput imena i prezimena.



Sl. 3.8. Greška prilikom registracije zbog praznog polja

Iz slike 3.9. može se vidjeti kako je zaporka *Filip1234* previše slična imenu korisnika. Slična greška je javljena ako se zaporke ne podudaraju ili ako je bilo koji od uvjeta za zaporku neispunjen.



Sl. 3.9. Greška prilikom registracije zbog neispravne lozinke

Kako bi se korisnik prijavio u aplikaciju, koriste se adresa e-pošte (engl. *e-mail adress)* i zaporka koja je korištena prilikom registracije korisničkog računa. Obrazac je prikazana na slici 3.10.

Obrazac za registraciju nalazi se unutar datoteke koja se naziva *forms.py*. Unutar te datoteke nalaze se svi obrasci koji su korišteni unutar aplikacije. Kod obrasca za registraciju prikazan je na slici 3.11.

Sl. 3.10. Obrazac za prijavu

```
class CustomUserForm(UserCreationForm):
    date_of_birth = forms.DateField(widget=DatePickerInput(),label='Datum rođenja')
    GENDER_CHOICES = [
        ('male', 'Muško'),
        ('female', 'Žensko'),
    ]
    gender = forms.ChoiceField(choices=GENDER_CHOICES, required=True,label='Spol')
    avatar_url = forms.CharField(max_length=128, required=True,label='URL Slike')
    widgets = {
            'date_of_birth': DateTimePickerInput(),
            }
    password1 = forms.CharField(
        label='Zaporka',
        widget=forms.PasswordInput
    )
    password2 = forms.CharField(
        label='Potvrda zaporke',
        widget=forms.PasswordInput
    )
    class Meta:
        model = User
        fields = ['first_name', 'last_name','oib','address','city', 'date_of_birth',
                  'gender', 'avatar_url', 'email','password1','password2']
        labels={
            'first_name':'Ime',
            'last_name':'Prezime',
            'address':'Adresa',
            'city':'Grad',
            'email':'Adresa e-pošte',
        }
```

Sl. 3.11. Kod obrasca za registraciju unutar *forms.py* datoteke

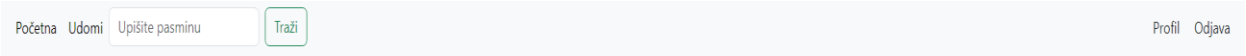
Klasa *CustomUserForm* predstavlja obrazac za registraciju, a ona nasljeđuje *UserCreationForm* kako bi naslijedila sve osnovne funkcionalnosti prilikom registracije korisnika. *UserCreationForm* je standardni obrazac za registraciju korisnika koji već ima definirana polja za registraciju poput *usernamea* (u ovoj aplikaciji je postavljeno na adresu e-pošte) i zaporke. Ostala polja definirana su unutar *CustomUserForm* klase. Klasa *Meta* sadrži tri atributa, od kojih je prvi *model* koji se nalazi unutar *models.py* datoteke i čiji se objekt kreira prilikom popunjavanja obrasca. Drugi, *fields*, predstavlja polja koja su korištena za kreiranje tog objekta. *Labels* je zadnji atribut koji predstavlja oznake koje imenuju polja unutar obrasca. Atribut *widgets* označava objekte koje služe za prilagođavanje polja unutar obrasca. U aplikaciji to se odnosi na birač datuma (engl. *datetime picker)* koji se koristi prilikom biranja datuma rođenja. Obrazac za prijavu nije potrebno definirati zato što se koristi standardni *Djangov* sustav za autentikaciju te on sve definira umjesto programera.

4. FUNKCIONALNOST APLIKACIJE

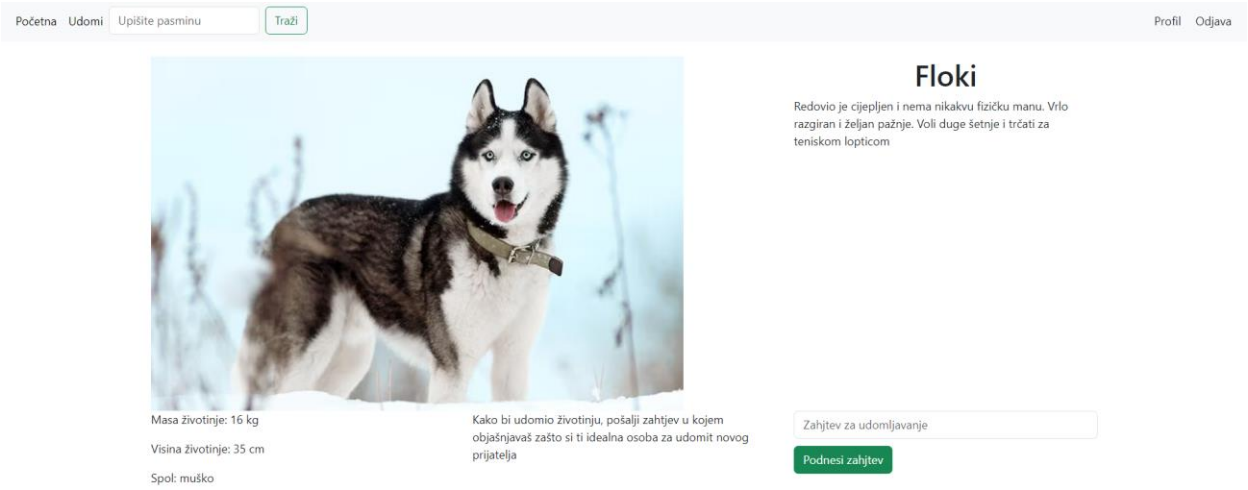
Unutar ovog poglavlja objašnjene su sve funkcionalnosti unutar aplikacije.

4.1. Slanje zahtjeva

Prilikom prijave korisnika u aplikaciju, dogode se neke promjene u izgledu same aplikacije. Umjesto poveznica za prijavu i registraciju, sada se nalaze poveznice *Profil* i *Odjava* (*Profil* koji odvodi na vlastiti profil, a *Odjava* koja služi za odjavu s računa), prikazano na slici 4.1.



Sl. 4.1. Profil i Odjava u navigacijskoj traci



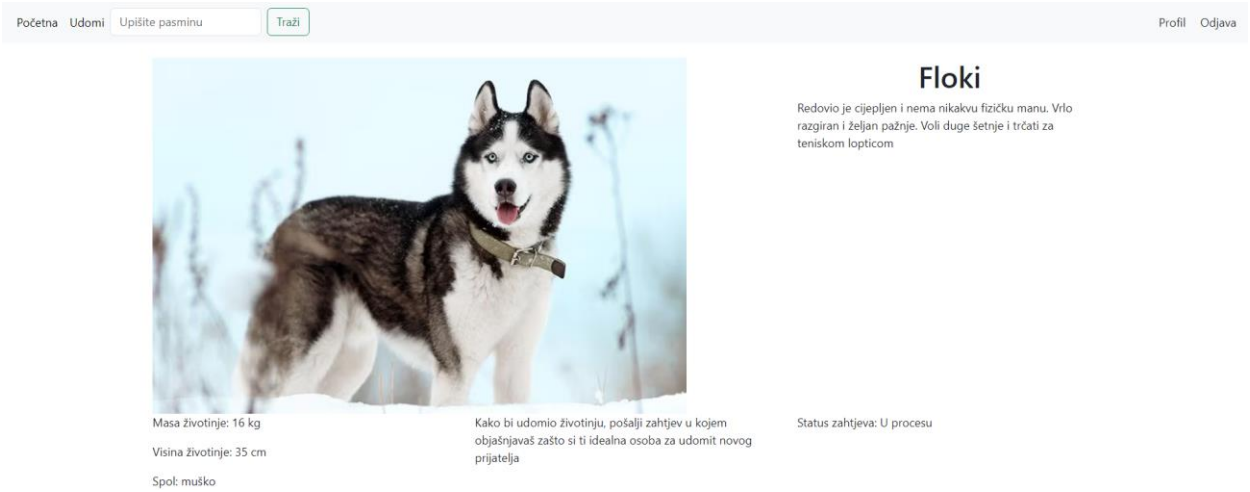
Sl. 4.2. Detalji životinja s korisničkog pogleda

Kada je korisnik prijavljen u svoj račun, može poslati zahtjev za udomljavanje životinje. Dok je korisnik prijavljen u aplikaciju, stranica na kojoj su detalji životinje malo promjeni izgled (gumb *Podnesi zahtjev* se može kliknuti i tekst je drugačiji, prikazano na slici 4.2.). Kako bi se zahtjev poslao potrebno je nešto napisati u postavljeno polje za unos teksta (engl. *text-box*) i stisnuti na gumb. Prilikom pritiska na gumb za zahtjev, poziva se metoda unutar *views.py* datoteke prikazana na slici 4.3.

```python
def new_request(request,animal_id):
    animal=get_object_or_404(Animal,pk =animal_id)
    if request.method == "POST" and request.user.is_authenticated:
        text = request.POST.get("text")
        adoptingReq = AdoptingRequest(
            animal_id=animal,
            user_id=request.user,
            status=AdoptingRequest.PENDING,
            description=text,
        )
        adoptingReq.save()
    return HttpResponseRedirect(reverse("app:animalDetails", args=(animal_id,)))
```

Sl. 4.3. Metoda za kreiranje zahtjeva za udomljavanje

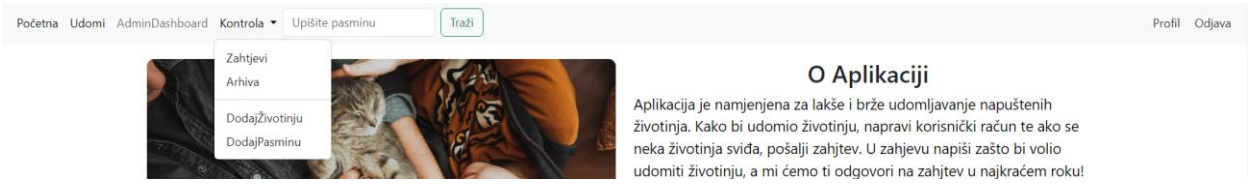
Metoda prvo pronalazi objekt životinje pomoću *Id-a* koji je poslan kao argument metode. Potom se provjerava je li korisnik prijavljen i je li metoda unutar HTTP (engl. *Hypertext Tranfser Protocol*) zahtjeva POST. Ako su uvjeti ispunjeni, kreira se objekt klase *AdoptingRequest* unutar kojeg se postavlja životinja za koju se šalje zahtjev, korisnik koji šalje zahtjev, korisnikov tekst prilikom slanja zahtjeva i status „na čekanju“ (engl. *pending*) na atribute klase *AdoptingRequest*. Korisnik nakon što je poslao zahtjev više ne može slati zahtjev za istu životinju, prikazano na slici 4.4.



Sl. 4.4. Nemogućnost slanja zahtjeva

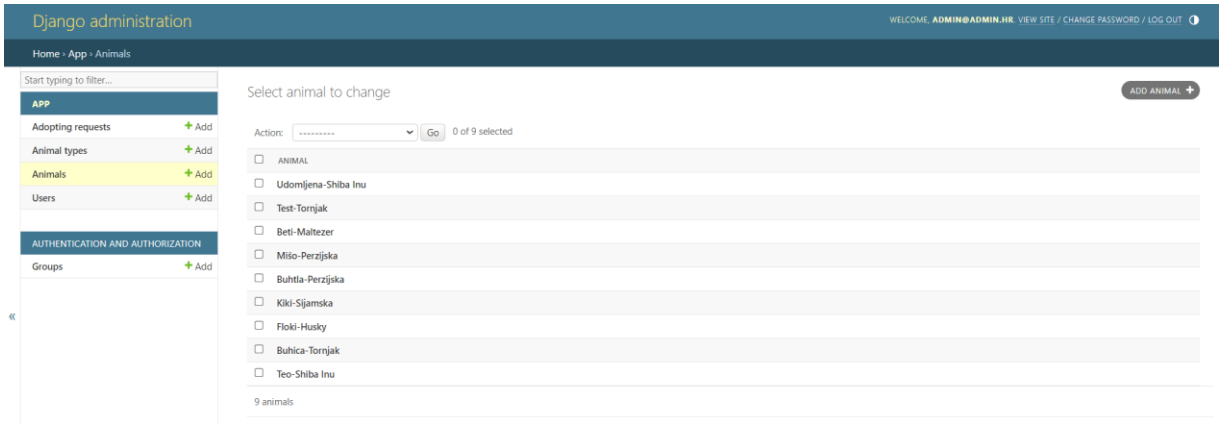
4.2. Donošenje odluke i arhiva

Odluku o udomljavanju životinje isključivo donosi administrator aplikacije. Prilikom prijave administratora u aplikaciju, promjeni se navigacijska traka u odnosu na običnog korisnika na način da su dodani padajući izbornik (engl. *dropdown*) i nova poveznica, prikazano na slici 4.5.

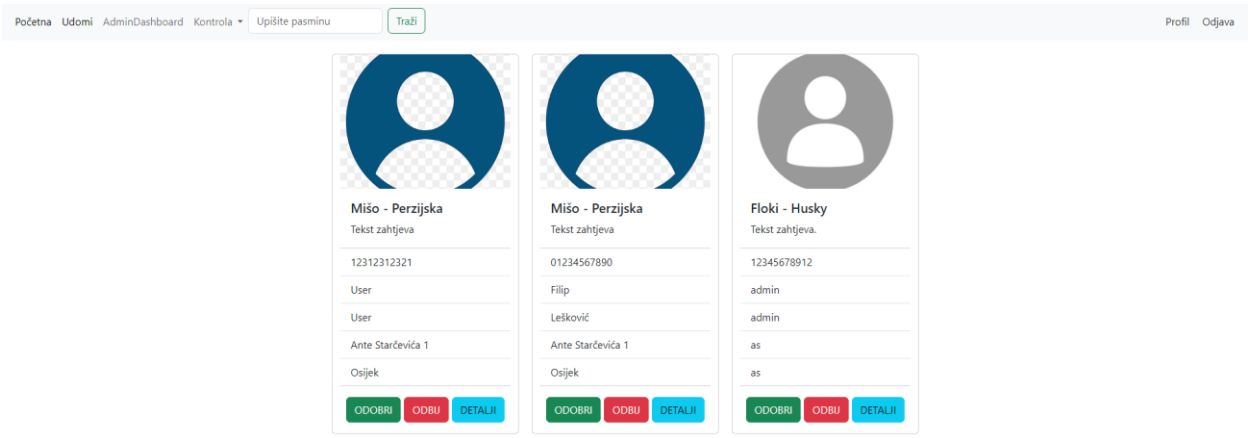


Sl. 4.5. Administratorska navigacijska traka

Na navigacijskoj traci dodana je poveznica *AdminDashboard* koji odvodi administratora na službenu *Djangovu* administratorsku stranicu. Administrator unutar *dashboarda* može dodavati nove podatke, mijenjati već postojeće i brisati ih. *Admin dashboard* prikazan je na slici 4.6. Osim toga, dodan je i padajući izbornik koji prikazuje četiri poveznice. Jedna od njih je poveznica *Zahtjevi* koja vodi na stranicu gdje se donose odluke o udomljavanju, prikazano na slici 4.7. Na toj stranici administratoru su prikazani svi zahtjevi koji su poslani. Zahtjev je prikazan kao kartica na kojoj se nalaze podatci o korisniku, tekst zahtjeva, slika životinje i tri gumba, od kojih dva služe za donošenje odluke. Donošenje odluke temelji se na administratorovom subjektivnom mišljenju. Ako administrator odbije zahtjev za udomljavanje, taj korisnik više ne može slati zahtjev za tu istu životinju, ali može za neku drugu. Ako administrator potvrdi zahtjev, životinja se miče s popisa slobodnih životinja i podatci na početnoj stranici su ažurirani. Metoda koja brine o tome prikazana je na slici 4.8. Gumb *detalji* koji se nalazi na zahtjevu odvodi administratora do detalja životinje za koju je zahtjev poslan.



Sl. 4.6. Admin dashboard



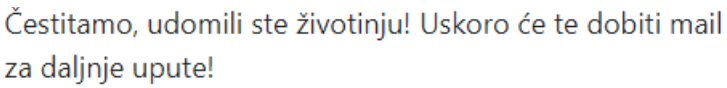
Sl. 4.7. Stranica sa zahtjevima za udomljavanje

```
def request_decision(request,req_id):
    req = get_object_or_404(AdoptingRequest, pk=req_id)
    animal_id = req.animal_id
    if request.method == "POST" and request.user.is_authenticated:
        inputvalue = request.POST.get("decision", None)
        if inputvalue == "APPROVE":
            req.status = AdoptingRequest.APPROVED
            animal_id.is_adopted=True
            animal_id.save()
        else:
            req.status = AdoptingRequest.DECLINED
        req.save()
    return HttpResponseRedirect(reverse("app:requests", args=()))
```

Sl. 4.8. Metoda za donošenje odluke

Prvo se dohvati objekt klase *AdoptinqRequest* pomoću *Id-a* zahtjeva koji je poslan kao argument metode. Opet se provjerava je li korisnik prijavljen i je li metoda unutar HTTP zahtjeva POST. Potom se dohvati vrijednost s gumba koji je pritisnut te se na temelju njegove vrijednosti grana metoda na dva dijela. Ako je zahtjev odobren (engl. *approved*) , postavlja se istinita vrijednost (engl. *true*) na svojstvo *is_adopted* što omogućuje da se životinja više ne prikazuje drugim korisnicima kao slobodna. Osim toga, ovisno o odluci ažurira se i stanje samog zahtjeva. Sve odbijene i odobrene zahtjeve administrator može pronaći u arhivi, slika 4.9. Korisniku se unutar detalja životinje ažurira status njegovog zahtjeva, ako je zahtjev odobren napisana je poruka prikazana na slici 4.10, a ako je odbijen, piše da je zahtjev odbijen.

Sl. 4.9. Arhiva zahtjeva



Sl. 4.10. Poruka o odobravanju zahtjeva

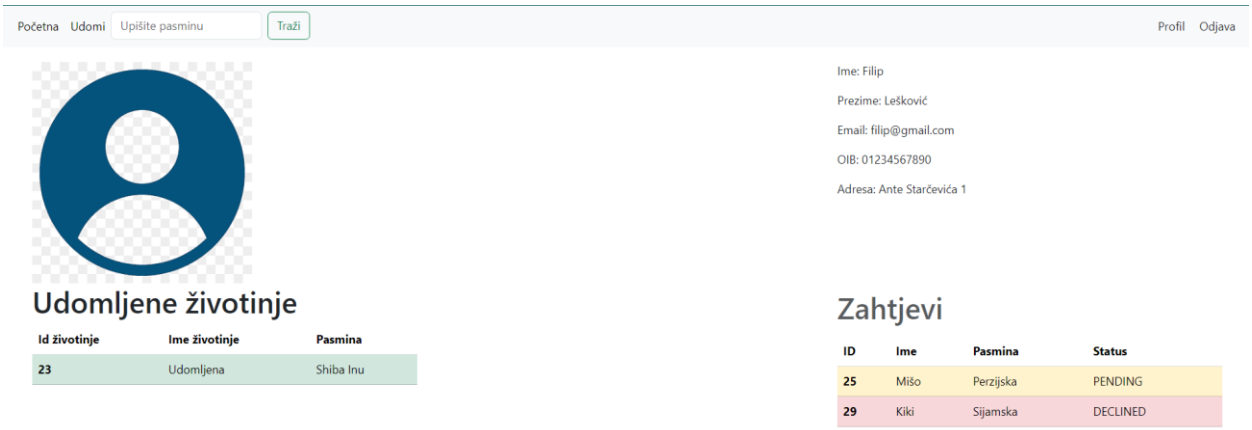
U arhivi nalaze se dvije tablic e, u jednoj su prikazani odobreni zahtjevi (tablica obojena u zeleno), dok su u drugoj prikazani odbijeni zahtjevi(tablica obojena u crveno). Svaki redak tablice predstavlja jedan zahtjev. Tamo se također nalaze i *Oib* korisnika i *Id* životinje te se klikom na njih može direktno doći do detalja životinja i profila korisnika. Prilikom odbijanja i odobravanja zahtjeva, tablice su redovito ažurirane pomoću metode prikazane na slici 4.11.

```
def archive(request):
    approved_requests=AdoptingRequest.objects.filter(status=AdoptingRequest.APPROVED)
    declined_requests=AdoptingRequest.objects.filter(status=AdoptingRequest.DECLINED)
    context={
        "approved":approved_requests,
        "declined":declined_requests,
    }
    return render(request, "app/archive.html", context)
```

Sl. 4.11. Metoda za filtriranje podataka u arhivi

Metoda se poziva svaki put kada se posjeti stranica *Arhiva*. Unutar metode filtrirani su odbijeni i odobreni zahtjevi pomoću *objects.filter* metode. Unutar te metode zahtjevi su filtrirani po statusu zahtjeva. Kada su zahtjevi dohvaćeni, prikaže se stranica *Arhiva*, a podatci su pomoću *contexta* dostupni unutar HTML datoteke gdje se dalje koriste za oblikovanje stranice. *Context* je predan kao argument metode *render* što omogućuje dostupnost *contexta* HTML datoteci. Također pojedinačna arhiva za svakog korisnika postoji i unutar njegovog profila. Na profilu korisnika nalaze se neke osnovne informacije o korisniku. Također su prikazane sve životinje koje je

korisnik udomio te su prikazani svi zahtjevi koji su odbijeni i koji još čekaju na odluku. Stranica profila korisnik prikazana je na slici 4.12. Svaki korisnik može do svog profila doći klikom na poveznicu *Profil* u navigacijskoj traci.



Sl. 4.12. Profil

Prilikom pritiska na bilo koji redak unutar tablica, korisnik se preusmjeri do detalja životinje za koju je poslao zahtjev. Ova funkcionalnost je odrađena pomoću JavaScript skripte tako što je na svaki redak postavljen slušatelj događaja (engl. *EventListener*) prikazano na slici 4.13

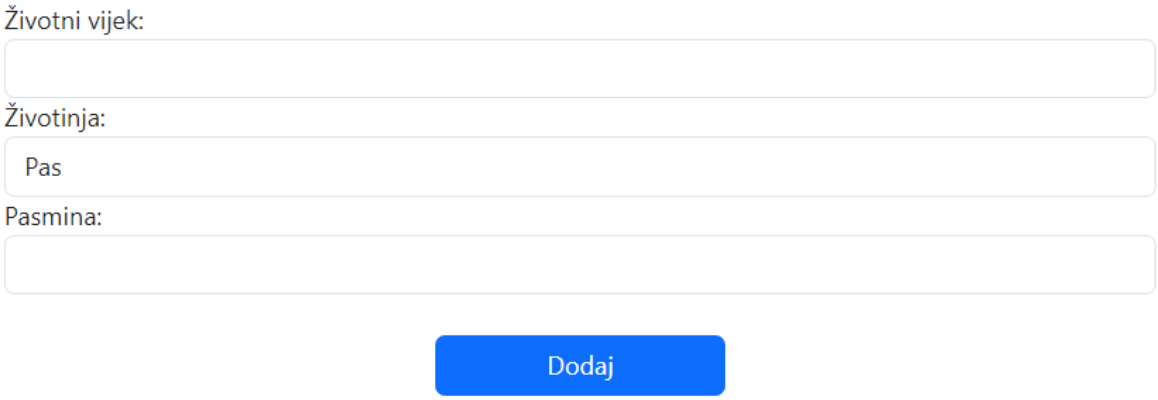
```
<script>
    document.addEventListener("DOMContentLoaded", function () {
      const rows = document.querySelectorAll("tr[data-href]");
      rows.forEach((row) => {
        row.addEventListener("click", function () {
          window.location.href = row.getAttribute("data-href");
        });
      });
    });
  </script>
```

Sl. 4.13. Skripta za redove unutar tablica

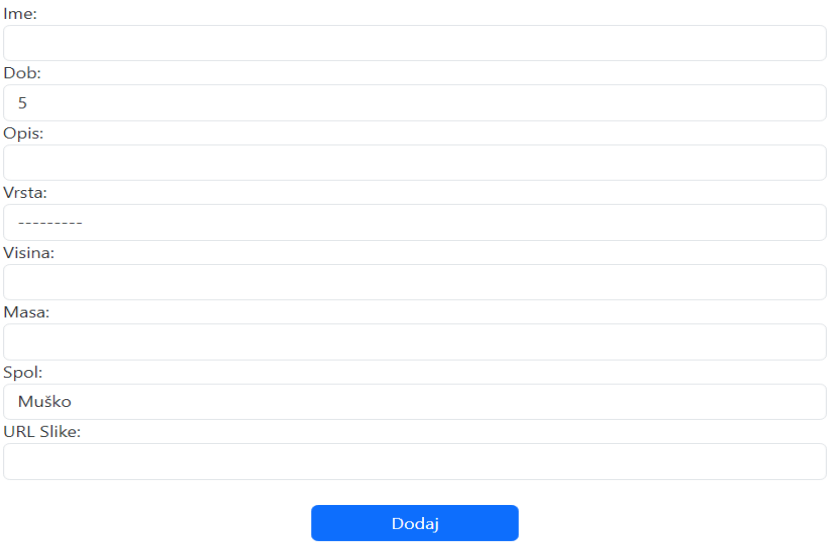
Unutar skripte dohvaćeni su svi elementi unutar stranice koji sadrže atribut *tr[data-href]*. Prilikom klika na određeni redak, dohvaća se vrijednost atributa pod tim retkom i pomoću metode *window.location.href* preusmjerava se korisnik do detalja životinje pod tim retkom.

4.3. Dodavanje životinja i pasmina

Administrator može na dva načina dodavati nove životinje i pasmine. Jedan od njih je preko *admin dashboarda*, a drugi je pomoću obrazaca do kojih se može doći preko poveznica koje se prikazuju na navigacijskoj traci administratora. Obrasci se nalaze u padajućem izborniku *Kontrola* na poveznicama *DodajPasminu* i *DodajŽivotinju* te se one otvaraju na zasebnim stranicama. Obrazac za dodavanje pasmine prikazan je na slici 4.14, dok je obrazac za dodavanje životinja prikazan na slici 4.15.



Sl. 4.14.Obrazac za dodavanje pasmine



Sl. 4.15. Obrazac za dodavanje životinje

Ta dva obrasca također se nalaze se unutar *forms.py* datoteke. Svaki obrazac predstavlja klase čiji je objekt dostupan pomoću *contexta* u *add.html* predlošku (engl. *template*) koji se otvara prilikom klika na jedan od dva guma. Slika 4.16. prikazuju te dva obrasca, odnosno te dvije definirane klase.

```
class AnimalForm(ModelForm):
    GENDER_CHOICES = [
        ('muško', 'Muško'),
        ('žensko', 'Žensko'),
    ]
    gender = forms.ChoiceField(choices=GENDER_CHOICES, required=True,label='Spol')
    avatar_url = forms.CharField(max_length=128, required=True,label='URL Slike')
    class Meta:
        model = Animal
        fields = ['name', 'age', 'description', 'type',
                  'height', 'weight','gender','avatar_url']
        labels = {
            'name': 'Ime',
            'age': 'Dob',
            'description': 'Opis',
            'type': 'Vrsta',
            'height': 'Visina',
            'weight': 'Masa',
        }

class AnimalTypeForm(ModelForm):
    ANIMAl_CHOICES=[
        ('DOG','Pas'),
        ('CAT', 'Mačka'),
    ]
    type=forms.ChoiceField(choices=ANIMAl_CHOICES,required=True,label='Životinja')
    class Meta:
        model=AnimalType
        fields =['lifespan','type','breed']
        labels={
            'lifespan':'Životni vijek',
            'breed':'Pasmina'
        }
```

Sl. 4.16. Kod obrazaca za dodavanje pasmina i životinja unutar forms.py

AnimalForm predstavlja obrazac za dodavanje novih životinja, dok *AnimalTypeForm* predstavlja dodavanje novih pasmina. Na atribut *model* postavlja se model unutar baze (*models.py* datoteke) za kojeg će se stvoriti obrazac. Atribut *fields* predstavlja atribute unutar modela koji će biti dostupni obrascu.

Metode koje brinu o ispravnom radu obrazaca nalaze se unutar *views.py* datoteke i prikazane na slici 4.17. Podijeljene su u dva dijela od koji se prvi dio izvršava ako je ispunjena provjera je li korisnik prijavljen i je li metoda unutar HTTP zahtjeva POST. U prvom dijelu se prikupljaju podatci koji su ispunjeni unutar obrasca i spremaju se u objekt određene klase ovisno o tome kreira li se pasmina ili životinja. Kada se stvori objekt, pomoću metode *save()* se sprema unutar baze podataka. Kada se objekt spremi, metoda preusmjerava korisnika na početnu stranicu aplikacije pomoću metode *reverse*, koja se nalazi unutar *HttpResponseRedirect* klase. Ako nije ispunjen prvi

uvjet, onda se izvršava drugi dio metode koji služi za otvaranje praznog obrasca, odnosno stvaranje objekta obrasca koji se pomoću *contexta* šalje u predložak *add.html* koji se onda potom prikaže na ekranu. Obje metode, *addAnimal* i *addBreed*, su slične, samo se pozivaju različiti konstruktori prilikom stvaranja obrasca.

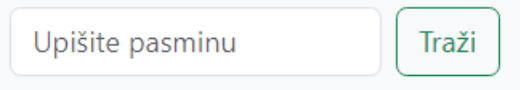
```
def addAnimal(request):
    if request.method == "POST" and request.user.is_authenticated:
        form = AnimalForm(request.POST)
        if form.is_valid():
            saved_animal = form.save(commit=False)
            saved_animal.is_adopted=False
            saved_animal.save()
            return HttpResponseRedirect(reverse("app:index"))
    else:
        form = AnimalForm()
    context = {
        "form": form,
    }
    return render(request, "app/add.html", context)

def addBreed(request):
    if request.method == "POST" and request.user.is_authenticated:
        form = AnimalTypeForm(request.POST)
        if form.is_valid():
            saved_breed = form.save(commit=False)
            saved_breed.save()
            return HttpResponseRedirect(reverse("app:index"))
    else:
        form = AnimalTypeForm()
    context = {
        "form": form,
    }
    return render(request, "app/add.html", context)
```

Sl. 4.17. Metode za dodavanje životinja i pasmina

4.4. Pretraživanje pasmina

Aplikacija omogućuje jednostavno pretraživanje životinja ovisno o tome kojoj pasmini pripadaju. U navigacijskoj traci nalazi se polje za unos i gumb koji pokreće pretraživanje životinja, prikazano na slici 4.18. Metoda koja brine za ispravno funkcioniranje ovog dijela aplikacije prikazana je na slici 4.19.



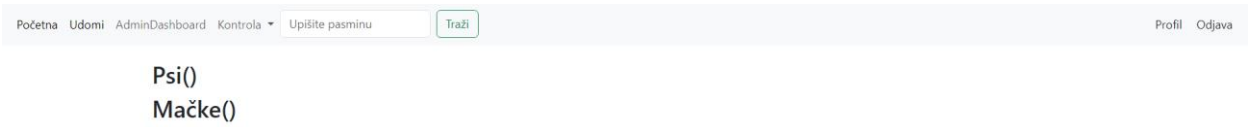
Sl. 4.18. Polje za unos i gumb za pretraživanje

```
def searchBreeds(request):
    if request.method == 'POST':
        searchedBreed = request.POST.get("searchBreed")
        if searchedBreed:
            animalType = AnimalType.objects.filter(breed__icontains=searchedBreed).first()
            if animalType:
                if animalType.type=="DOG":
                    dogs= Animal.objects.filter(type__breed=animalType.breed,is_adopted=False)
                    cats=[]
                else:
                    cats=Animal.objects.filter(type__breed=animalType.breed,is_adopted=False)
                    dogs=[]
                context = {
                    "dogs":dogs,
                    "cats":cats,
                    "user": request.user,
                }
            else:
                context={}
        else:
            context={}
    return render(request, "app/adopt.html", context)
```

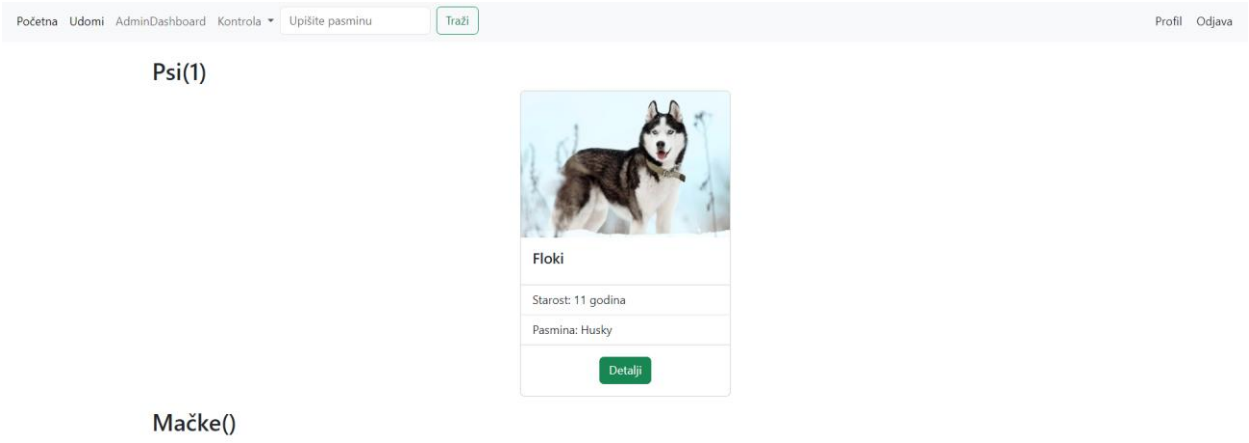
Sl. 4.19. Metoda za pretraživanje pasmina

Unutar metode prvo se provjerava je li metoda unutar HTTP zahtjeva POST. Potom se iz polja uzima vrijednost pomoću *POST.get* metode i provjerava se postoje li unutar baze podataka pasmine koja sadrže *string* koji je unesen prilikom pretraživanja. U slučaju da je pritisnut gumb *Traži*, a pritom nije ništa upisano u polje za unos, automatski se predaje prazan *context* i renderira se stranica koja je prikazana na slici 4.20. Istovremeno može biti pronađeno više pasmina, ali se uzima ona prva pronađena kako bi se pojednostavila metoda. Parametar *breed__icontains* unutar metode *filter* služi za filtriranje rezultata tako što provjerava sadrži li ime pasmine uneseni *string*. Međutim, ako upisani *string* nije dio imena niti jedne pasmine, predaje se prazan *context* kao argument metode *render*. Rezultat neuspjelog pretraživanja prikazan je na slici 4.20. Ako je pronađena pasmina koja je pas, ona se sprema u varijablu *dogs*, a ako je pasmina mačka, sprema se u varijablu *cats*. Metoda vrača metodu *render* koja zapravo renderira *adopt.html* predložak koji se koristi za glavni izbornik životinja odnosno *Udomi* stranicu. To znači da ova metoda zapravo prikazuje samo filtrirane životinje na toj stranici. Na primjer kada se upiše neki *string*, npr. *husk*, aplikacija bi trebala prikazati sve pse koji su pasmine *Husky*, to i prikazuje slika 4.21. Zbog

ponovnog renderiranja, nestane *string* iz polja za upis, ali je aplikacija prepoznala da treba prikazati sve *Huskye*.



Sl. 4.20. Test neuspješnog pretraživanja pasmine



Sl. 4.21. Test uspješnog pretraživanja pasmine

5. ZAKLJUČAK

Iako postoje već brojne aplikacije na temu ovog završnog rada, kako u Hrvatskoj, tako i u svijetu, svaka je drugačija svoj način. Razlike mogu biti u načinu udomljavanja ili o različitoj količini vrsta životinja koje posjeduju. Tako i tehnički gledano, svaka aplikacija je napravljena na drugačiji način, koriste se različite tehnologije, različiti pristupi problemu i slično.

U ovom završnom radu prikazana je aplikacija koja omogućuje *online* udomljavanje životinja. Prikazan je izgled stranice te korisničke i administratorske funkcionalnosti za korištenje ove aplikacije. Također objašnjeno je kako se ispravno registrirati kako bi mogli daljnje koristiti aplikaciju. Osim toga prikazane su i ostale funkcionalnosti koje služe za vođenje evidencije životinja i pretraživanje životinja koje omogućuje ciljano i brže udomljavanje.

Uz dodatna proširenja, aplikaciju mogu koristiti azili za udomljavanje životinja kao *online* opciju za udomljavanje. Proširenja poput slanje e-pošte o detaljima za daljnje postupanje nakon potvrde o udomljavanju, kolekcije slika pojedinih životinja i poboljšana kategorizacija uveliko bi olakšalo i povećalo kvalitetu samog procesa.

LITERATURA

[1] DogsTrust: Dogs Rehoming & Dog Rescue Charity dostupno na

https://www.dogstrust.org.uk/rehoming/how-to-adopt [16.7.2024.]

[2] Sklonište za životinje, Čakovec / ZEU Prijatelj životinja i prirode dostupno na https://www.prijatelji-zivotinja.org/hr/udomi-psa/proces-udomljavanja [16.7.2024.]

[3] Projekt Adopt Žarkovica Dogs | Udomi, volontiraj, proširi glas… dostupno na https://www.adopt-zarkovica.eu/projekt/ [16.7.2024.]

[4] Snoopy Udruga za zaštitu životinja dostupno na https://www.snoopy.hr/kako-udomiti-psa/ [5.8.2024.]

[5] PetRescue- Create happines. Save lives. – PetRescue dostupno na

https://www.petrescue.com.au/ [5.8.2024.]

[6] MDN Web Docs - Mozzila dostupno na

https://developer.mozilla.org/en-US/docs/Web/HTML [21.7.2024.]

[7] MDN Web Docs - Mozzila dostupno na

https://developer.mozilla.org/en-US/docs/Web/CSS [21.7.2024]

[8] TechTarget: Purchase Intent Dana for Enterprise Tech Sales.. dostupno na https://www.techtarget.com/whatis/definition/bootstrap#:~:text=Bootstrap%20is%20a%20free%2C%20open,of%20syntax%20for%20template%20designs. [25.7.2024.]

[9] MDN Web Docs - Mozzila dostupno na

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction#maintainable [25.7.2024.]

[10] dbt Developer Hub dostupno na https://docs.getdbt.com/terms/dry [25.7.2024.]

[11] DjangoProject, DRY- dostupno na https://docs.djangoproject.com/en/5.1/ref/urls/ [21.8.2024.]

SAŽETAK

U ovom završnom radu predstavljena je aplikacija za udomljavanje životinja. U teorijskom dijelu predstavljena su već postojeća rješenja koje se nalaze na internetu te je objašnjen njihov proces udomljavanja životinja. Potom su objašnjene tehnologije koje su korištene tijekom pravljenja same aplikacije, poput Djanga, HTML i Bootstrapa. U praktičnom dijelu prikazani su svi dijelovi aplikacije koji su nužni za ispravno korištenje iste. Također prikazuju se i moguće pogreške prilikom registracije novih korisnika zbog nepoštivanja pravila registracije. Prikazane su i objašnjene sve korisničke funkcionalnosti poput slanja zahtjeva za udomljavanje, pregled profila i praćenje statusa zahtjeva. Također su objašnjene i prikazane sve administratorske funkcionalnosti poput dodavanja životinja, pasmina, donošenja odluka o zahtjevu i praćenje evidencije životinja. Sve funkcionalnosti popraćene su kodom, uz njegovo detaljno objašnjenje.

Ključne riječi: *Bootstrap, Django,* udomljavanje, životinje, web aplikacija

ABSTRACT

Web application for adopting animals

In this thesis, an application for adopting animals is presented. In the theoretical part of the thesis, already existing solutions found on the Internet are presented and their process of adopting animals is explained. The technologies used during the creation of the application itself, such as Django, HTML and Bootstrap were explained. The practical part of the thesis shows all the parts of the application that are necessary for its correct use. Possible errors during registering new users due to non-compliance with the registration rules are also displayed. All user functionalities are shown and explained, such as sending requests for adoption, viewing profiles and monitoring the status of requests. All administrative functions such as adding animals, breeds, making decisions on requests and monitoring animal records are also explained and shown. All functionalities are accompanied by code with a detailed explanation.

Keywords: Bootstrap, Django, adopting, animals, web application

PRILOG

- GitHub link od aplikacije: https://github.com/filipleskovic/Udomljavanje-zivotinja