Razvoj web aplikacije za najam vozila „Take your ride“

Koraca, Deni

Sveučilište u Rijeci
Fakultet informatike
i digitalnih tehnologija

Sveučilišni prijediplomski studij Informatika

Deni Koraca

Razvoj web aplikacije za najam vozila „Take Your Ride“

Završni rad

Mentor: izv. prof. dr. sc. Marija Brkić Bakarić

Rijeka, srpanj 2024.

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija

Rijeka, 3.6.2024.

Zadatak za završni rad

Pristupnik/ica: Deni Koraca

Naziv završnog rada: Razvoj web aplikacije za najam vozila "Take your ride"

Naziv završnog rada na engleskom jeziku: Development of the vehicle rental web application 'Take your ride'

Sadržaj zadatka:
Cilj završnog rada je dati detaljan prikaz razvoja web aplikacije za najam vozila „Take Your Ride“ s podrškom za tri različite vrste korisnika. Neregistrirani korisnici imaju pristup ponudi vozila, registrirani mogu dodatno unajmljivati vozila i mijenjati svoje osobne podatke, a najviša razina ovlasti omogućava i izmjenu podataka o vozilima te dodavanje novih vozila. Za razvoj aplikacije koristit će se razvojni okvir Django.

Mentor/ica
Izv. prof. dr. sc. Marija Brkić Bakarić

Voditelj za završne radove
Izv. prof. dr. sc. Miran Pobar

Zadatak preuzet: 3.6.2024.

Deni Koraca
(potpis pristupnika/ice)

Radmile Matejčić 2,
51000 Rijeka, Hrvatska

T: +385 (0)51/584-700
E: ured@inf.uniri.hr
www.inf.uniri.hr

OIB: 64218323816
IBAN: HR1524020061400006966

Sažetak

U ovom završnom radu prikazana je izrada web aplikacije za najam vozila „Take Your Ride“ korištenjem razvojnog okvira Django. Detaljno su pojašnjeni svi segmenti Djangovog razvojnog okvira te dodatno implementirani segmenti poput *tempus_dominus*, *cripsy_forms* i *crispy_bootstrap5*. Aplikacija je namijenjena za tri vrste korisnika: neregistrirani, registrirani i administrator. Neregistrirani korisnici mogu samo pregledati ponudu automobila, registrirani korisnici mogu i unajmljivati automobile rezervacijom termina te imaju mogućnost izmjene svojih podataka. Administrator s najvišom razinom ovlasti može dodatno izmjenjivati podatke o vozilima ili dodavati nova vozila.

Ključne riječi: razvojni okvir Django; Django projekt; python; web aplikacija; aplikacija za najam vozila

SADRŽAJ

1. Uvod

U današnje vrijeme sve je veća potreba za digitalizacijom kod malih obrta, kao i kod većih korporacija zbog lakše mogućnosti praćenja, uređivanja, brisanja i filtriranja pojedinih proizvoda ili usluga.

Kroz ovaj završni rad prikazan je proces izrade aplikacije za najam vozila „Take Your Ride“ u Django razvojnom okviru. Osnovna karakteristika ovog alata je njegova jednostavnost koja omogućava brzi razvoj sigurnih i skalabilnih web aplikacija zbog svojih ugrađenih funkcionalnosti i modularne strukture. Za izradu programskog koda koristi se alat Visual Studio Code te programski jezik Python.

Ukoliko se tvrtka ili obrt odluči za korištenje web aplikacije za najam vozila to može donijeti mnoge prednosti za nju, ali i za njene klijente, odnosno korisnike. Sam proces iznajmljivanja prijevoznog sredstva je automatiziran što znači da korisnik može samostalno iznajmiti vozilo bez da kontaktira djelatnike tvrtke. Na taj način korisnik ne mora čekati u redu, a djelatnici tvrtke imaju manje posla oko bilježenja rezervacije te dolazi do uštede vremena kod korisnika (registriranih korisnika) i djelatnika (administrativnih korisnika). Još jedna od prednosti je brzo i jednostavno rezerviranje vozila bez potrebe za telefonskim ili elektroničkim razgovorom putem e-pošte. Na ovaj način se poboljšava korisničko iskustvo obzirom da korisnici imaju jednostavan pristup aplikaciji i dostupnost informacija. Što se tiče dostupnih informacija, moguće je dobiti informacije o dostupnosti vozila, nazivu i modelu vozila, godini izdanja, registracijskoj oznaci, boji te vidjeti fotografiju vozila. Klijenti mogu unutar web aplikacije pronaći informacije o tvrtki koja se bavi iznajmljivanjem vozila, njezinoj misiji, vrijednostima i timu. Postoje 3 vrste korisnika: neregistrirani, registrirani korisnici i administrator. Svaka od vrste korisnika ima svoje određene funkcionalnosti. Aplikacija je izrađena tako da neregistrirani korisnici mogu samo saznati više informacija o tvrtki i pogledati objavljena vozila koje tvrtka nudi. Tek nakon što se korisnik registrira, korisnik dobiva mogućnost unajmiti vozilo. Treća mogućnost je registracija kao administrator te se tada dobiva mogućost dodavanja novog vozila, izmjena podataka vozila ili brisanja vozila.

Osim što se ovim načinom automatizacije povećava produktivnost, moguće je smanjiti i financijske troškove obzirom da je moguće zaposliti manji broj radnika, te nije potrebna fizička lokacija ureda u koji klijenti dolaze. Također, ovakve usluge su klijentu na raspolaganju 24 sata dnevno svih 7 dana u tjednu. Smanjenje administrativnih poslova dovodi do zadovoljstva s obje strane, tj. korisnika i zaposlenika, što je i najvažnija stvar u svakom poslovanju kako bi ono bilo uspješno.

2. Django

Django je jedan od najpopularnijih besplatnih web razvojnih okvira otvorenog koda koji se temelji na Python programskom jeziku. Poznat po svojoj jednostavnosti, brzini i fleksibilnosti što ga čini omiljenim izborom među mnogim programerima. Nastao je 2003. godine od strane Adrian Holovaty i Simon Willison, ali je tek 2005. godine predstavljen javnosti. Naziv je dobio po Djangou, poznatom belgijskom gitaristu koji je prepoznatljiv po svojoj brzini i eleganciji. Django razvojni okvir se izdvaja zbog više razloga koji olakšavaju rad programera. Prvi od njih je objektno-relacijsko mapiranje (engl. *Object-Relational Mapping,* ORM) koje omogućava komunikaciju s bazom podataka iz Python-a bez korištenja SQL upita. Druga prednost Djangoa je to što nudi administracijsko sučelje (engl. *Admin Interface*) koje automatski generira administratorski panel za lakšu manipulaciju podacima. Također, Django pruža dobar i jednostavan sistem za rad s obrascima i validacijom unosa te ugrađene protokole za zaštitu kao što su XSS i CSRF. Skriptiranje na više web sjedišta ili samo XSS napadi (engl. *Cross site scripting*) omogućuju napadaču ubrizgavanje skripti na strani klijenta na web stranice koje pregledavaju drugi korisnici. To se obično postiže pohranjivanjem zlonamjernih skripti u bazu podataka koje će biti dohvaćene i prikazane drugim korisnicima, ili tako što se korisnike navede da kliknu na poveznicu koja će uzrokovati izvršenje JavaScript-a napadača u pregledniku korisnika. Međutim, XSS napadi mogu poticati iz bilo kojeg nepouzdanog izvora podataka, poput kolačića ili web servisa, kada podaci nisu dovoljno očišćeni prije uključivanja na stranicu. Korištenje Django predložaka štiti od većine XSS napada. Ipak, važno je razumjeti koje zaštite pruža i njegova ograničenja [1].

Django pruža mogućnost ponovne upotrebe komponenti (engl. *Don't Repeat Yourself,* DRY) te dolazi sa značajkama spremnim za korištenje kao što su sustav za prijavu (engl. *Login*), povezivanje na bazu podataka te CRUD operacije. Što se tiče CRUD operacija, za njih je karakteristično to što nude operacije vezane uz kreiranje (engl. *Create*), čitanje (engl. *Read*), ažuriranje (engl. *Update*) te brisanje (engl. *Delete*).

Django slijedi model-pogled-predložak (engl. *Model View Template, MVT*) uzorak dizajna. Model označava podatke koje želite predstaviti, obično se radi o podacima iz baze podataka. Pogled predstavlja rukovatelja zahtjevima koji vraća relevantni predložak i sadržaj na temelju zahtjeva korisnika. Predložak predstavlja tekstualnu datoteku (poput HTML datoteke) koja sadrži izgled web stranice s logikom prikaza podataka [2].

Mnoge velike kompanije poput Instagram-a, Facebook-a, Pinterest-a i Disqusa koriste Django, što dodatno govori o pouzdanosti i sposobnosti podržavanja velikih projekata s velikim brojem zahtjeva. Razvojni okvir Django zbog svoje fleksibilnosti, jednostavnosti i moćnih ugrađenih alata se smatra neizostavnim alatom svakog web programera. Još jedna o zanimljivih činjenica o razvojnom okviru Django je da njegova zajednica broji više od 1200 programera te na GitHub-u ima više od dvadeset i osam tisuća zvjezdica, odnosno više od tisuću i petsto korisnika prati njegova ažuriranja [3].

3. Priprema radnog okruženja i konfiguracijske komponente

U nastavku je opisano postavljanje radnog okruženja, stvaranje modela i glavne konfiguracijske komponente

3.1. Postavljanje radnog okruženja i stvaranje projekta

Za izradu web aplikacije koristio se Visual Sudio Code i programski jezik Python te Git Bash terminal u samom Visual Studio Code-u, u kojem je potrebno instalirati Django pomoću naredbe *pip install django*.

Potom se kreira Django projekt pomoću naredbe *django-admin startproject mysite*. Za kreiranje aplikacije koristi se naredba *./manage.py startapp main*. Potrebno je povezati projekt s aplikacijom u datoteci settings.py na način da se doda linija *main.apps.MainConfig*, pod INSTALLED_APPS. Za pokretanje aplikacije koristi se naredba *./manage.py runserver*.

3.2. Glavne konfiguracijske komponente

Razvojni okvir Django ima nekoliko ključnih komponenti koje definiraju strukturu i samu funkcionalnost web aplikacije. Neke od glavnih konfiguracijskih komponenata su pogledi (engl. *Views*), URL obrasci (engl. *Urls*), modeli (engl. *Models*) i predlošci (engl. *Templates*). Pogledi su klase pomoću kojih se obrađuju HTTP zahtjevi te vraćaju odgovarajući odgovor. URL obrasci služe za usmjeravanje zahtjeva na odgovarajuće poglede. Komponenta modeli predstavlja strukturu podataka pomoću koje je omogućena interakcija s bazom podataka koristeći Django objektno-relacijsko mapiranje (engl. *Object-Relational Mapping*, ORM) i komponente predložaka koje sadrže HTML datoteku za definiranje izgleda web stranice.

3.3. Migracija podataka

Kako bi podaci unutar web aplikacije ispravno funkcionirali, potrebno je redovito ažuriranje baze podataka što uključuje dodavanje novih podataka, ažuriranje podataka te brisanje starih podataka koji više nisu potrebni unutar sustava. Iz tog razloga se provode migracije nad bazom podataka.

Kako bi se proveo postupak migracije podataka unutar razvojnog okvira Django, potrebno je generirati datoteke migracija na osnovu promjena u modelima koji su definirani unutar datoteke *models.py.* Migracije se kreiraju pomoću naredbe *python manage.py makemigrations* unutar terminala. Nakon što se pokrene prethodno spomenuta naredba, pregledaju se sve promjene napravljene unutar modela te se kreira migracijska datoteka koja opisuje sve te promjene. Ovaj dokument se kasnije koristi za primjenu tih promjena na bazi podataka.

Iduća naredba, *migrate*, se koristi za primjenu migracija na bazu podataka. Ova naredba izvršava SQL kod koji je generiran u migracijskim datotekama, na način da se promjene definirane u modelima reflektiraju u bazi podataka. Naredba *python manange.py migrate,* koja se, također pokreće unutar terminala te primjenjuje sve neprimijenjene migracije kojim su kreirane, što omogućava da baza podataka bude sinkronizirana s Django modelima.

4. Izrada aplikacije

U nastavku je opisan postupak izrade aplikacije.

4.1. Postavke

Datoteka *Settings.py* služi za konfiguraciju Django projekta. Zbog ispravnog funkcioniranja aplikacije datoteka se izmijenila i prilagodila. Na taj način se omogućilo korištenje biblioteka unutar projekta. U INSTALLED_APPS potrebno je dodati *main.apps.MainConfig*, *tempus_dominus*, *crispy_forms* i *crispy_bootstrap5*.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main.apps.MainConfig',
    'tempus_dominus',
    'crispy_forms',
    'crispy_bootstrap5',
]
```

Za povezivanje projekta s aplikacijom korišten je *main.apps.MainConfig*. Pomoću *tempus_dominus* omogućene su brojne mogućnosti za unos datuma i vremena. Prethodno spomenuti dodatak pruža lakše i estetski privlačnije oblikovanje Django obrazaca pomoću *crispy_forms*. *Crispy_bootstrap5* omogućava korištenje Bootstrap 5 razvojnog okvira u kombinaciji s *cripsy_forms* što omogućava da obrasci izgledaju personalizirano te budu funkcionalni na svim rezolucijama.

Za konfiguraciju *django-crispy-forms* biblioteke koriste Bootstrap 5 stilski paket. Najprije se postavlja koji stilski paketi su dopušteni za korištenje *django-cripsy-forms*. Također postavlja se stilski paket koji će biti postavljen kao zadan za stilizaciju obrazaca. U oba slučaja unesen je bootstrap5.

```
CRISPY_ALLOWED_TEMPLATE_PACKS = "bootstrap5"
CRISPY_TEMPLATE_PACK = "bootstrap5"
```

Za upravljanje medijskim datotekama postavlja se MEDIA_URL i MEDIA_ROOT kojim se postavlja direktorij za čuvanje medijskih datoteka. Putanju do statičkog direktorija *static* osiguralo se pomoću STATICFILES_DIRS u kojoj se nalaze datoteke poput *style.css* za dizajn html datoteka.

4.2. Django URL-ovi

Unutar main direktorija, datoteka *urls.py* služi za konfiguraciju URL obrazaca za Django aplikaciju te omogućava poslužitelju da servira medijske datoteke tijekom razvoja. Implementiran je admin model uz *django.contrib* kojim se osigurava moguć pristup admin sučelju. Uvoze se *path* i *include* funkcije iz *django.url* biblioteke. Funkcija *path* koristi se za definiranje url obrazaca, dok funkcija *include* omogućuje uključivanje drugih URL

konfiguracija. Iz modela *django.conf* uvozi se *settings* koji omogućuje pristup svim postavkama definiranim u *settings.py* datoteci. Kako bi se omogućilo korištenje statičkih i medijskih datoteka tijekom razvoja dodaje se *static* iz *django.conf.urls.static* modela.

```
from django.contrib import admin
from django.urls import path, include
from django.conf import settings
from django.conf.urls.static import static
```

Za definiranje URL putanje koristi se *urlpatterns* u kojoj se pomoću funkcije *path*, dodaje putanja za admin sučelje. Također, dodaje se putanja za aplikaciju *main* koja postaje korijenska domena aplikacije.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
]
```

Posljednjim dijelom datoteke dodaju se URL obrasci za posluživanje medijskih datoteka tijekom razvoja aplikacije. Funkcija *if settings.DEBUG* osigurava da se medijske datoteke poslužuju samo kada je *DEBAG* postavljen na *True* što nam govori da je u tijeku razvoja. Ukoliko je aplikacija u tijeku razvoja dogovara se URL obrazac za posluživanje medijskih datoteka.

```
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

4.3. Modeli

U nastavku je opisana datoteka *Models.py* te modeli koji postoje unutar aplikacije.

Kako bi komponenta modeli radila, potrebno je na početku implementirati različite module i funkcije koje omogućuju rad komponente. Prvi implementirani modul je *from django.db import models* koji omogućuje kreiranje modela, odnosno predstavlja strukturu podataka u bazi podataka. Sljedeći implementirani model je *from django.utils.translation import gettext_lazy as _* pomoću kojeg se omogućuje da označeni tekstovi budu prevedeni „lijeno“, to jest da se ne prevedu odmah nego tek kada se koriste. Treći implementirani modul je *from django.contrib.auth.models import User* koji uvozi Djangov model korisnika koji se koristi za autentifikaciju i autorizaciju korisnika unutar web aplikacije. Posljednji modul *from django import forms* pruža mogućnost kreiranja i obrade HTML obrazaca. Ovaj modul omogućuje da se klase i funkcije za definiranje i validaciju obrazaca koriste unutar aplikacije.

U nastavku je definiran svaki od modela s odgovarajućim atributima i vrstama polja. Razlikuju se tri vrste modela: model vozila, model rezervacije vozila i model korisnika.

4.3.1. Model vozila

Prvi definirani model predstavlja svako pojedinačno vozilo u aplikaciji. Prvi atribut je lista MAKE_CHOICES u kojoj se može odabrati automobilska marka poput Toyote, Ford-a, Honde… Nadalje, na isti način je implementirana boja vozila te posljednja lista za kategorije vozila gdje je moguće izabrati između malog, srednjeg ili velikog automobila te kombija.

```
MAKE_CHOICES = [
    ('toyota', 'Toyota'),
```

```
    ('ford', 'Ford'),
    ('bmw', 'BMW'),
    ('audi', 'Audi'),
    ('mercedes', 'Mercedes-Benz'),
    ('honda', 'Honda'),
    ('chevrolet', 'Chevrolet'),
    ('nissan', 'Nissan'),
    ('volkswagen', 'Volkswagen'),
    ('hyundai', 'Hyundai'),
    ]
```

Potom su definirana polja modela. Parametar *make* je vrste CharField koja označava pohranjivanje kratkog tekstualnog podatka kojem je određeno da je maksimalna duljina 20 pomoću parametra *max_length*. Parametar *choices* povlači prije definirane MAKE_CHOICES koja sadrži ponuđene vrijednosti. Sljedeće polje *model*, definirano je za upis modela vozila također kao i u polje *make* koji koristi CharField s jedinim parametrom *max_length* postavljenog na 50 znakova. Treće polje *year* omogućuje unos prirodnih brojeva. Ovo polje nema nikakvih parametra. Kako bi postojala mogućnost unosa registarske pločice, dodano je polje *license_plate* koje je CharField vrste s parametrom *max_length* s vrijednosti 10 znakova. Sljedeće polje je *color* koje je definirano na isti način kao i polje *make* s jednom razlikom da polje *color* povlači COLOR_CHOICES. Za dodavanje slike vozila dodano je polje *image* vrste ImageField koja omogućuje dodavanje slike,. Parametrom *upload_to* definiran je direktorij u kojem se pohranjuje slike te sljedećim parametrom *blank* omogućuje se kreiranje novog vozila bez slike. Polje *is_available* označava dostupnost vozila, vrsta polja je BooleanField koji ima samo dvije vrijednosti (True ili False). Parametrom se postavila zadana vrijednost na *True*. Cijena iznajmljivanja po danu je sačuvana u polju *price_per_day* kao DecimalField sa maksimalno 10 znamenki. Kategorija vozila određena je poljem *category*, koje koristi definirani izbor CATEGORY_CHOICES, a zadana vrijednost definirana je na vrijednost *mali*. Na kraju modela vozila nalazi se metoda *str* koja vraća tekstualni zapis modela. Ova metoda koristi izborni sistem Django kako bi se vratila čitljiva verzija vozila kojom se naknadno prikazuje model vozila.

```
make = models.CharField(max_length=20, choices=MAKE_CHOICES)
model = models.CharField(max_length=50)
year = models.IntegerField()
license_plate = models.CharField(max_length=10)
color = models.CharField(max_length=20, choices=COLOR_CHOICES)
image = models.ImageField(upload_to='vehicle_images/', blank=True, null=True)
is_available = models.BooleanField(default=True)
price_per_day = models.DecimalField(max_digits=10, decimal_places=2, default=0)
category = models.CharField(max_length=20, choices=CATEGORY_CHOICES, default='mali')
```

4.3.2. Model rezervacije vozila

Model *VehicleReservation* predstavlja model za rezervaciju vozila. Sastoji se od 4 polja: vozilo (engl. *Vehicle*), korisnik (engl. *User*), početno vrijeme (engl. *Start_time*) i krajnje vrijeme (engl. *End_time*). Polje vozila je vanjski ključ (engl. *ForeignKey*) koji se odnosi na model vozila, što znači da svaka rezervacija mora biti povezana s jednim vozilom. Ukoliko se vozilo izbriše, brišu se sve njegove rezervacije. To se postiglo pomoću funkcije *n_delete=models.CASCADE*. Polje korisnik također je *ForeignKey* na model korisnika. Na isti način je osigurano ukoliko se izbriše korisnik, onda se brišu sve njegove rezervacije. Polja

početnog i krajnjeg datuma su DateTimeField-ovi koji osiguravaju unos datuma početka i kraja rezervacije.

```
class VehicleReservation(models.Model):
    vehicle = models.ForeignKey(Vehicle, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()

    def __str__(self):
        return f"{self.vehicle} reserved by {self.user} from {self.start_time} to
{self.end_time}"
```

Metoda str vraća čitljiv prikaz rezervacije koji uključuje podate o vozilu i korisniku te vremensko razdoblje rezervacije.

4.3.3. Model korisnika

Model *Korisnika* predstavlja dodatne informacije o modelu *User*. Sadrži pet polja: *ime_korisnika, prezime_korisnika, adresa_korisnika, mobitel_korisnika,* i *email_korisnika*. Svi podaci su tipa CharField osim *email_korisnika* koji je EmailField. E-mail korisnika služi za validaciju ispravnog formata e-mail adrese. Polje *user* je povezano s modelom *User*, jedan na jedan vezom koja omogućuje da su samo jedan korisnik i jedan *user* međusobno povezani. Pomoću funkcije *on_delete* osiguralo se da ukoliko se izbriše *user* da se automatski izbriše i *korisnik*. Metoda *str* vraća e-mail korisnika u čitljivom obliku.

```
class Korisnik(models.Model):
    ime_korisnika = models.CharField(max_length=100)
    prezime_korisnika = models.CharField(max_length=100)
    adresa_korisnika = models.CharField(max_length=100)
    mobitel_korisnika = models.CharField(max_length=10)
    email_korisnika = models.EmailField()
    user = models.OneToOneField(User, null=True, on_delete=models.CASCADE)
    def __str__(self):
        return self.email_korisnika
```

4.4. Django Admin

Datoteka *Admin.py* definira administrativno sučelje za modele *Vehicle, VehicleReservation* i *Korisnik* u Django admin panelu, čineći upravljanje modelima jednostavnije i efikasnije. Kako bi *Admin.py* ispravno funkcionirao treba izvršiti uvoz *admin* modula i svih modela iz datoteke *models.py*.

4.4.1. Prikaz modela vozila

Za prikazivanje modela vozila, odnosno modela *Vehicle* u administrativnom sučelju korištena je *@admin.register(Vehicle)* linija koda. Pomoću klase *VehicleAdmin* se prilagođava izgled i funkcionalnost prikaza samog modela. Atributom *list_display* definira se vidljivost *make, model, year, license_plate, color, price_per_day, category* i *is_available*. Dodani su filteri na polja *make, color, category* i *is_available* kako bi se moglo lakše snalaziti u listi vozila. Također, na taj način je omogućeno pretraživanje polja *make, model* i *license_plate* pomoću atributa *search_fields*. Kako bi se poboljšala preglednost i upotrebljivost korišten je atribut *fieldsets* kojim se organiziraju polja u obrascu.

```
@admin.register(Vehicle)
```

```
class VehicleAdmin(admin.ModelAdmin):
list_display = ('make', 'model', 'year', 'license_plate', 'color', 'price_per_day',
'category', 'is_available')
    list_filter = ('make', 'color', 'category', 'is_available')
    search_fields = ('make', 'model', 'license_plate')
    fieldsets = (
        (None, {
            'fields': ('make', 'model', 'year', 'license_plate', 'color', 'image',
'price_per_day', 'category', 'is_available')
        }),
    )
```

4.4.2. Prikaz modela najma vozila

Za prikaz svih najma vozila, odnosno modela *VehicleReservation* korišteno je *@admin.register(VehicleReservation).* Za prilagodbu je dodana klasa *VehicleReservationAdmin*. Klasa sadrži atribut *list_display za prikaz* polja *vehicle, user, start_time* i *end_time*. Pomoću atributa *search_fields* omogućena je pretraga po poljima *vehicle__make, vehicle__model, start_time* i *end_time*.

```
@admin.register(VehicleReservation)
class VehicleReservationAdmin(admin.ModelAdmin):
    list_display = ('vehicle','user', 'start_time', 'end_time')
    search_fields = ('vehicle__make', 'vehicle__model', 'start_time', 'end_time')
```

4.5. Obrasci

Django koristi obrasce kako bi olakšao kreiranje i upravljanje HTML obrascima za unos podataka. Postoje dvije vrste obrazaca. Prvi je *forms.Form* koji predstavlja običan obrazac koji nije povezan s modelom, a drugi je *forms.ModelForm* koji je povezan s određenim modelom. Ovi obrasci omogućavaju automatsko generiranje polja na osnovi modela. Koriste različita polja za unos određenog tipa podataka kao što su CharField, EmailField i DateField. Validacija podataka je automatska te je moguće dodati još dodatne metode za validaciju određenog polja. Korištenjem Django obrazaca, web aplikacija postaje brža, efikasnija i sigurnija.

Na početku koda treba dodati različite biblioteke i module koje omogućuju rad s obrascima. Na samom početku izvršen je uvoz modula *forms* pomoću *from django import forms*. Kako bi se mogli koristiti izrađeni modeli potrebno je implementirati sve modele iz datoteke *models.py* pomoću naredbe *from .models import* *. Treća linija datoteke *from django.contrib.auth.forms import UserCreationForm* služi za kreiranje korisničkih obrazaca, dok redak *from django.contrib.auth.models import User* služi za autentifikaciju i upravljanje korisnicima. Pomoću *from tempus_dominus.widgets import DateTimePicker* implementiran je dodatak *DateTimePicker* koji služi za jednostavniji unos datuma i vremena. Biblioteka *crispy_forms* omogućava unapređenje izgleda pomoću klase *FormHelper* i klase *Submit* za gumbove. Modul *timezone* u Django omogućuje rad s vremenskim zonama, odnosno precizan rad s datumima i vremenima u aplikaciji. Kombinacijom ovih biblioteka i modula razvio se snažan, jednostavan i fleksibilan sistem za unos podataka u Django aplikaciji.

U nastavku su detaljnije objašnjenje četiri vrste obrazaca: obrazac za registraciju korisnika, obrazac za korisnika, obrazac za vozilo te obrazac za iznajmljivanje vozila.

4.5.1. Obrazac za registraciju korisnika

Klasa *RegisterUserForm* predstavlja prilagođen obrazac za registraciju korisnika. Naslijeđena je od Djagno *UserCreationForm* obrasca. Koristi ugrađeni Djangov model *user*, uključena polja su *username, password1* i *password2*. Metoda *__init__* osigurava da svi potrebni elementi za kreiranje korisničkog obrasca budu uključeni i pravilno stilizirani.

```
class RegisterUserForm(UserCreationForm):

	class Meta:
		model = User
		fields = ('username', 'password1', 'password2')


	def __init__(self, *args, **kwargs):
		super(RegisterUserForm, self).__init__(*args, **kwargs)

		self.fields['username'].widget.attrs['class'] = 'form-control'
		self.fields['password1'].widget.attrs['class'] = 'form-control'
		self.fields['password2'].widget.attrs['class'] = 'form-control'
```

4.5.2. Obrazac korisnik

Klasa *KorisnikForm* predstavlja *ModelForm* koji je povezan na model *Korisnik* te omogućuje unos i izmjenu korisničkih podataka. Obrazac koristi polja: *ime_korisnika, prezime_korisnika, adresa_korisnika, mobitel_korisnika* i *email_korisnika*. Pomoću atributa *labels* prilagođava se prikazan tekst polja u aplikaciji. Za svako polje se formiraju prilagođeni dodaci (engl. *Widgeti*) pomoću dodataka atributa. Polja se dodatno oblikuju pomoću CSS klase *form-control* i dodatnim *placeholder* atributom za poboljšanje korisničkog iskustva.

```
class korisnikForm(forms.ModelForm):
    class Meta:
        model = Korisnik
fields = ['ime_korisnika', 'prezime_korisnika', 'adresa_korisnika', 'mobitel_korisnika',
'email_korisnika']
        labels = {
            'ime_korisnika': 'Ime',
            'prezime_korisnika': 'Prezime',
            'adresa_korisnika': 'Adresa',
            'mobitel_korisnika': 'Mobitel',
            'email_korisnika': 'Email',
        }
        widgets = {
 'ime_korisnika': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Ime'}),
'prezime_korisnika': forms.TextInput(attrs={'class': 'form-control', 'placeholder':
'Prezime'}),
'adresa_korisnika': forms.TextInput(attrs={'class': 'form-control', 'placeholder':
'Adresa'}),
'mobitel_korisnika': forms.TextInput(attrs={'class': 'form-control', 'placeholder':
'Mobitel'}),
'email_korisnika': forms.EmailInput(attrs={'class': 'form-control', 'placeholder':
'Email'}),
}
```

4.5.3. Obrazac vozilo

Klasa *VehicleForm* je Django *ModelForm* kojom je omogućen unos i ažuriranje informacija o vozilu. Povezana je modelom *Vehicle* te sadrži polja: *make, model, year, license_plate, color, image, is_available, price_per_day* i *category*. Svako polje koristi specifičan dodatak za prilagodbu izgleda po predlošku na model. Na samom kraju nalazi se metoda *__init__* koja

sadrži inicijalizaciju *FormHelper* iz *crispy_forms* biblioteke. Obrascu se poboljšava izgled i dodaju dodatne mogućnosti. Metoda šalje obrazac na POST te dodaje dugme s tekstom „Spremi“ .

```
class VehicleForm(forms.ModelForm):
    class Meta:
        model = Vehicle
 fields = ['make', 'model', 'year', 'license_plate', 'color', 'image', 'is_available',
'price_per_day', 'category']
        widgets = {
            'make': forms.Select(attrs={'class': 'form-control'}),
            'model': forms.TextInput(attrs={'class': 'form-control'}),
            'year': forms.NumberInput(attrs={'class': 'form-control'}),
            'license_plate': forms.TextInput(attrs={'class': 'form-control'}),
            'color': forms.Select(attrs={'class': 'form-control'}),
            'image': forms.ClearableFileInput(attrs={'class': 'form-control'}),
            'is_available': forms.CheckboxInput(attrs={'class': 'form-check-input'}),
            'price_per_day': forms.NumberInput(attrs={'class': 'form-control'}),
            'category': forms.Select(attrs={'class': 'form-control'}),
        }

    def __init__(self, *args, **kwargs):
        super(VehicleForm, self).__init__(*args, **kwargs)
        self.helper = FormHelper()
        self.helper.form_method = 'post'
        self.helper.add_input(Submit('submit', 'Spremi'))
```

4.5.4. Obrazacza iznajmljivanje vozila

Obrazac za iznajmljivanje vozila, odnosno klasa *RentalForm* je prilagođena Djagno *ModelForm* pomoću koje se može dodati rezervacija vozila. Obrazac je povezan s modelom *VehicleReservation*. Uključena su polja: *vehicle, start_time* i *end_time*. Obrazac sadrži prilagođene dodatke koji uključuju *DateTimePicker* iz *tempus_dominus* biblioteke za unos datuma. U klasi *Meta* definirano je da polje *vehicle* bude skriveno pomoću atributa *HiddenInput*, dok polja *start_time* i *end_time* koriste *DateTimePicker* sa opcijama sprečavanja odabira prošlih datuma i omogućava prikaz kalendara. Metoda *__init__* prilagođava inicijalizaciju obrasca tako da prihvaća opcionalni *initial_vehicle* argument koji postavlja početnu vrijednost za polje *vehicle*. Također metoda *clean_start_time* ponovno provjerava da datum nije u prošlosti.

```
def clean_start_time(self):
        start_time = self.cleaned_data['start_time']
        if start_time < timezone.now():
            raise forms.ValidationError("Datum početka ne može biti u prošlosti.")
        return start_time
```

4.6. Navigacijska traka

Navigacijska traka važan je dio svake web aplikacije te omogućuje korisniku kretanje između stranica. Navigacijska traka web aplikacije ima tamnu pozadinu pomoću *navbar-dark bg-dark* te se širina automatski prilagođava širini ekrana zahvaljujući *navbar-expand-lg* i *navbar-toggler*, koje omogućavaju preklapanje navigacije na manjim ekranima, to jest pojavljuje se hamburger izbornik. Unutar elementa div s klasom *container* nalazi se logo i naziv aplikacije „Take Your Ride“ koji je povezan s poveznicom *{% url 'main:home' %}* na početnu stranicu (Slika 3.). S desne strane navigacijske trake nalaze se sve web stranice aplikacije. Navigacijska

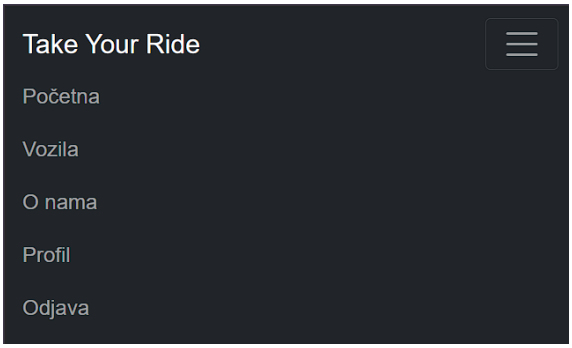
traka sadrži poveznicu na *Početnu stranicu, stranicu Vozila, O nama, Dodaj Auto, Profil, Odjava, Prijava* i *Registracija*. Ukoliko korisnik nije prijavljen, korisniku se prikazuju samo *Početna stranica, stranica Vozila, O nama, Prijava* i *Registracija* (Slika 3.). Ukoliko se korisnik prijavi ili registrira, stranice *Prijava* i *Registracija* nestaju te se pojavljuju stranice *Profil* i *Odjava* (Slika 1.). Ukoliko se korisnik prijavi kao administrator pojavljuje se dodatno stranica *Dodaj automobil*.

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="{% url 'main:home' %}">Take Your Ride</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:home' %}">Početna</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:vehicles' %}">Vozila</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:about' %}">O nama</a>
                    </li>
                    {% if user.is_staff %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:add_vehicle' %}">Dodaj
automobil</a>
                    </li>
                    {% endif %}
                    {% if user.is_authenticated %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:profile' %}">Profil</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:logout' %}">Odjava</a>
                    </li>
                    {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:login' %}">Prijava</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'main:register'
%}">Registracija</a>
                    </li>
                    {% endif %}
                </ul>
            </div>
        </div>
    </nav>
```

Slika 1. prikazuje navigacijsku traku za ekrane veće širine koja na lijevoj strani ima naziv aplikacije, odnosno prikazuje se Take Your Ride. Na desnoj strani su poveznice koje vode na preostale stranice kao što su *Početna, Vozila, O nama, Profil* te *Odjava*. Ovakva vrsta navigacijske trake se prikazuje ukoliko je korisnik već prijavljen u web aplikaciji. U suprotnom, navigacijska traka izgleda malo drugačije. Primjerice, umjesto *Profil,* piše *Prijava* te umjesto *Odjava* piše *Registracija.*

Slika 1. Navigacija za ekrane veće širine

Na Slici 2. prikazana je navigacijska traka, odnosno hamburger izbornik, za ekrane manje širine kao što su mobilni uređaji. Na samom vrhu se nalazi naziv web aplikacije, nakon čega slijede poveznice na druge stranice kao što su *Početna, Vozila, O nama, Profil* te *Odjava.* Kao i kod navigacijske trake za ekrane veće širine, ukoliko korisnik nije prijavljen u aplikaciju, onda se prikazuju drugačije poveznice.



Slika 2. Navigacija za ekrane manje širine

4.7. Django predložak

U datoteci *views.py* klasa *LandingPageView* koristi generičku *ListView* klasu za prikaz objekata iz modela *Vehicle*. U klasi se poziva *landing.html* pomoću *template_name* kojim se postavlja sučelje stranice. Metodom *get* dohvaćaju se svi objekti iz modela *Vehicle* te ih se prosljeđuje u sučelje stranice kroz varijablu *kate* kroz koju je omogućen prikaz svih objekata na landing stranici. URL konfiguracija za ovu stranicu definirana je pomoću *path(", views.LandingPageView.as_view(), name='landing'),* kojom se postavlja URL putanja na prazno što govori da se radi o primarnoj stranici. Iz datoteke *views* se dohvaća klasa *LandingPageView* te metodom *as_view()* pretvara u funkciju koju Django može pozvati prilikom obrade URL zahtjeva.

```
class LandingPageView(ListView):
    model = Vehicle
    template_name = 'landing.html'

    def get(self, request, *args, **kwargs):
        kate = Vehicle.objects.all()
        return render(request, 'landing.html', {'kate': kate})
```

4.7.1. Landing.html

Stranica *landing.html* u sekciji zaglavlje (engl. *Head*) sadrži *{% load static %}* za učitavanje statičkih datoteka. *<!DOCTYPE html>* definira se radi o dokumentu HTML5 te se postavlja jezik dokumenta na hrvatski pomoću *<html lang="hr">*. Za definiranje kodiranja znakova kao UTF-8 standardu dodaje se *<meta charset="UTF-8">*. Kako bi se postigla responzivnost stranice treba se omogućit prilagođavanje za različite uređaje. Iz tog razloga se dodaje *<meta name="viewport" content="width=device-width, initial-scale=1.0">*. Naslov se izmjenjuje

na svakoj stranici gdje se koristi što se postiglo na način da se unutar atributa title dodaje atribut *block title* koji to omogućuje. U posljednjem dijelu zaglavlja implementiran je Bootstrap 5 razvojni okvir pomoću *<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css">* te se pomoću *<link rel="stylesheet" type="text/css" href="{% static 'style.css' %}">* uključuje CSS datoteka iz statičkih resursa.

```
{% load static %}
<!DOCTYPE html>
<html lang="hr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{% block title %}TakeYourRide{% endblock %}</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
    <link rel="stylesheet" type="text/css" href="{% static 'style.css' %}">
</head>
```

U sesiji tijelo (engl. *Body*) na samom početku dodaje se navigacijska traka. Glavni dio dokumenta je rezervirano mjesto za dinamički sadržaj koji je definiran u drugim dokumentima koji nasljeđuju ovaj predložak, što se postiglo pomoću *{% block content %}{% endblock %}*.

Podnožje stranice je definirano *<footer class="text-center py-4">* s centriranim tekstom.

```
<footer class="text-center py-4">
        <p>&copy; 2024 Web stranice Deni Koraca. Sva prava pridržana.</p>
    </footer>
```

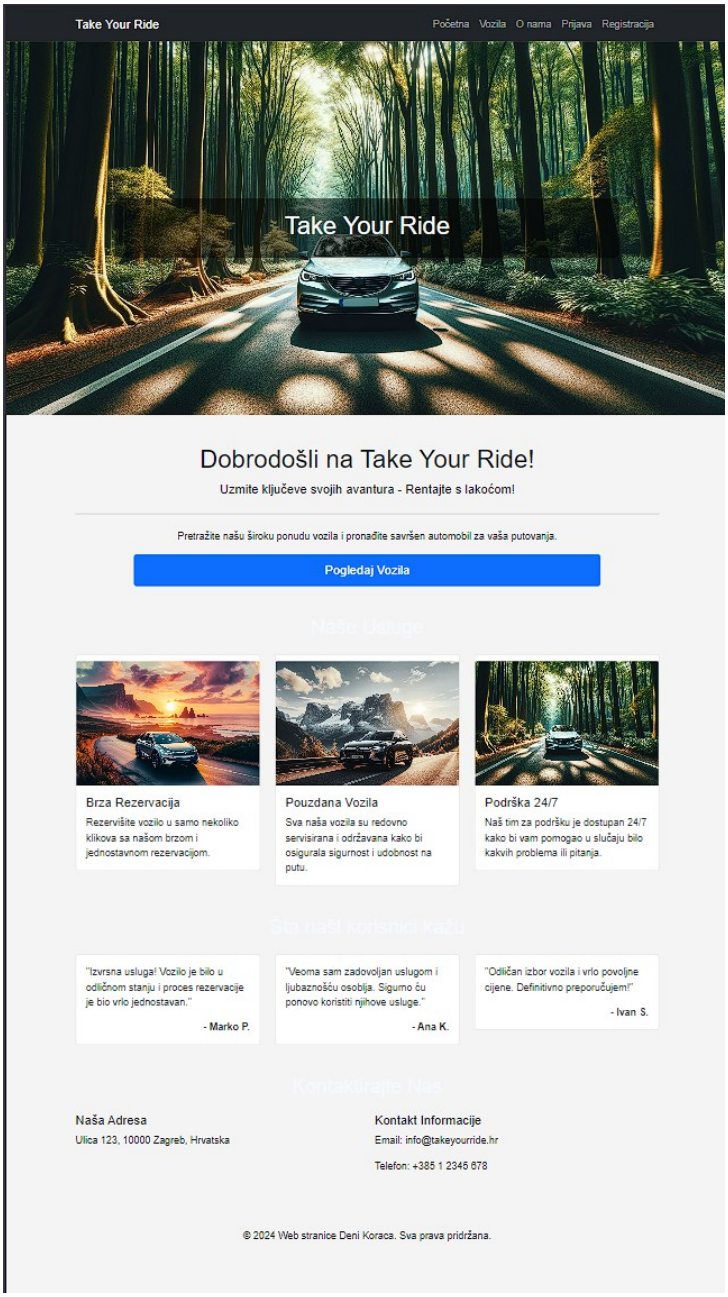
Podnožje sadrži tekst u kojem se naznačuje da ju je Deni Koraca izradio. Na samom kraju HTML datoteke nalazi se skripta *<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js">* koja uključuje Bootstrap 5 JavaScript *bundle* za funkcionalnosti kao što su padajući izbornici i modeli.

4.8. Django pogledi

4.8.1. Početna stranica

Početna stranica web aplikacije Take Your Ride koristi Bootstrap5 za stilizaciju sučelja. Na samom početku *home.html* datoteke *{% load static %}* omogućava učitavanje statičkih datoteka pomoću Django predloška. U samom dijelu zaglavlja (engl. *Head*) implementiran je Bootstrap css pomoću poveznica i prilagođeni *stile.css* koji se nalazi u *static* direktoriju. Na vrhu web stranice nalazi se navigacijska traka. Sama atraktivnost stranice postigla se pomoću dinamičkog transparenta na kojem se nalazi veliki naslov „Take Your Ride“, dok se u pozadini preko cijele stranice izmjenjuju 3 slike što se postiglo pomoću JavaScript-a na samom dnu programskog koda koji svakih 5 sekundi automatski izmjenjuje slike. Funkcija *showNextBanner* uklanja klasu *active* s trenutne slike i postavlja na sljedeću sliku. Sljedeći dio tekstualni blok s klasom *text-centar* pozdravlja korisnike s naslovnim tekstom „Dobro došli na Take Your Ride“ te podnaslovom „Uzmite ključeve svojih avantura - Rentajte s lakoćom!" koji predstavlja moto tvrtke. Nakon podnaslova, nalazi se tekst s gumbom za prelazak na stranicu s

automobilima. Treća sekcija prikazuje naše usluge pomoću tri odlomka gdje svaki sadrži sliku, naslov i popratni opis. Prvi odlomak opisuje brzu rezervaciju vozila, drugi naglašava pouzdanost i redovito održavanje vozila, dok posljednji odlomak ističe 24/7 korisničku podršku. Sve slike nalaze se u statičkim resursima u direktoriju *media* te ih se poziva naredbom *{% static 'media/pozadina.webp' %}*. Četvrta sekcija sadrži tri pozitivne recenzije zadovoljnih korisnika odvojenih u karticama. Kartice koriste Bootstrap klasu *card* za stilizaciju. Posljednja sekcija sadrži korisne podatke o tvrtki te je podijeljena u 2 stupca gdje lijevi stupac sadrži adresu tvrtke, a desni kontakt informacije poput elektroničke pošte i broj mobitela. Unutar priloga se nalazi programski kod za *home.html* datoteku. Na Slici 3. prikazana je početna stranica web aplikacije Take Your Ride. Na samom vrhu moguće je vidjeti navigacijsku traku s poveznicama na druge stranice. Zatim slijedi fotografija s nazivom aplikacije. Iduća sekcija prikazuje poruku dobrodošlice i moto tvrtke. Nudi se mogućnost pregleda ponude vozila. Navedene su tri prednosti ove web aplikacije kao što su: brza rezervacija, pouzdana vozila te podrška 24/7. Također, prikazane su recenzije nekih od korisnika. Na dnu, odnosno u podnožju, se nalazi kontakt koji uključuje adresu, broj telefona i adresu elektroničke pošte.



Slika 3. Naslovna stranica

4.8.2. O nama

Izrađena je stranica *about.html* koja nudi informacije o nama, odnosno našoj misiji, vrijednostima i više detalja o timu. Ovaj dokument koristi osnovni predložak *landing.html* kao osnovu za stranicu - *{% extends "landing.html" %}*. Slijedeći korak je učitavanje statičkih dijelova kao što su slike, CSS, JavaScript i slično pomoću *{% load static %}*. Potom se definira blok za naslov pomoću *{% block title %}O nama - Take Your Ride{% endblock %}*. Glavni dio sadržaja nalazi se u bloku sa sadržajem (engl. *Block content*). Uvodni dio sastoji se od dijela div, h1 i paragrafa.

```
<main class="container my-5">
    <div class="text-center mb-5">
        <h1 class="display-4">O nama</h1>
        <p class="lead">Saznajte više o našoj misiji, vrijednostima i timu.</p>
```

```
        <hr class="my-4">
    </div>
```

Odlomak „Naša misija“ sastoji se od Bootstrap *row* sa marginom na dnu. Također, razlikuju se dva stupca gdje se unutar prvog stupca nalazi tekst, dok se unutar drugog stupca nalazi fotografija.

```
<div class="row mb-5">
    <div class="col-md-6">
        <h2 style="color: blue;">Naša Misija</h2>
        <p>Naša misija je pružiti najbolju moguću uslugu rentanja vozila...</p>
    </div>
    <div class="col-md-6">
        <img src="{% static 'media/pozadina3.webp' %}" class="img-fluid rounded" alt="Naša
Misija">
    </div>
</div>
```

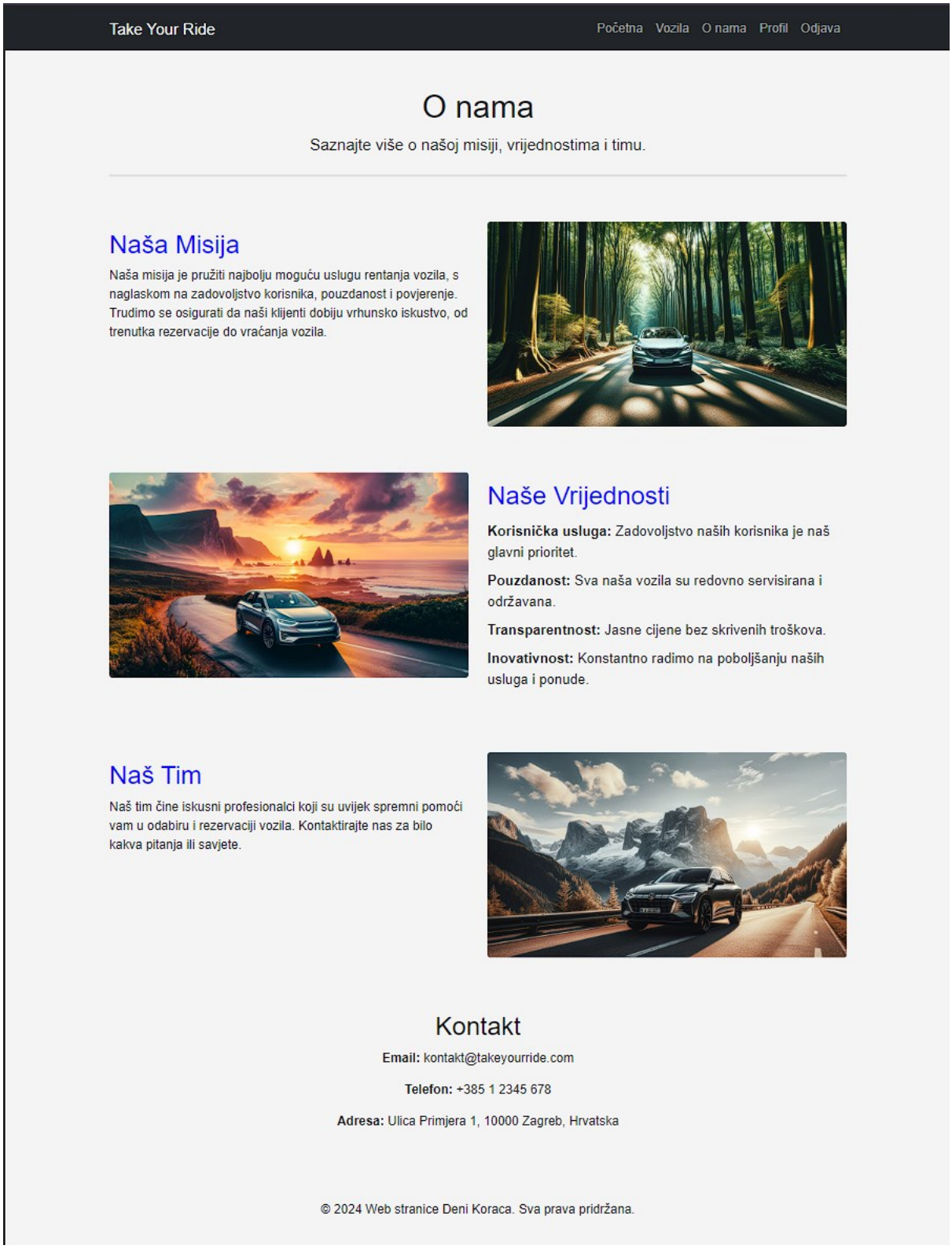
Idući dio „Naše vrijednosti“ ima sličnu strukturu kao prethodno opisani kod. Jedina razlika je u mijenjaju redoslijeda prikaza, odnosno najprije se prikazuje fotografija, pa tekst. To se postiglo s Bootstrap klasama *order-md-2* i *order-md-1.*

Sekcija „Naš tim“ ima istu strukturu kao i sekcija „Naša misija“ samo s odgovarajućim tekstualnim i slikovnim zapisima.

Posljednja sekcija „Kontakt informacije“ sadrži jedan red sa jednim stupcem koji zauzima cijelu širinu (svih dvanaest stupaca) *col-md-12*. Također, tekst je centriran, a jedan dio i podebljan (engl. *Bold*).

```
<div class="row mb-5">
    <div class="col-md-12 text-center">
        <h2 style="color: black;">Kontakt</h2>
        <p><strong>Email:</strong> kontakt@takeyourride.com</p>
        <p><strong>Telefon:</strong> +385 1 2345 678</p>
        <p><strong>Adresa:</strong> Ulica Primjera 1, 10000 Zagreb, Hrvatska</p>
    </div>
</div> <div class="row mb-5">
    <div class="col-md-12 text-center">
        <h2 style="color: black;">Kontakt</h2>
        <p><strong>Email:</strong> kontakt@takeyourride.com</p>
        <p><strong>Telefon:</strong> +385 1 2345 678</p>
        <p><strong>Adresa:</strong> Ulica Primjera 1, 10000 Zagreb, Hrvatska</p>
    </div>
</div>
```

Na Slici 4. prikazuje se stranica „O nama“. Na vrhu se nalazi navigacijska traka. Nakon čega slijedi opis misije, vrijednosti i nekoliko informacija o timu. Također, na ovoj stranici nalaze se tri fotografije.



Slika 4. Stranica „O nama“

4.8.3. Prijava korisnika

Izrađen je *login.html* za prijavu korisnika koji koristi odjeljak (engl. *Block)* strukturu za umetanje sadržaja u osnovni predložak, odnosno *landing.html* datoteku. Zatim slijedi odjeljak

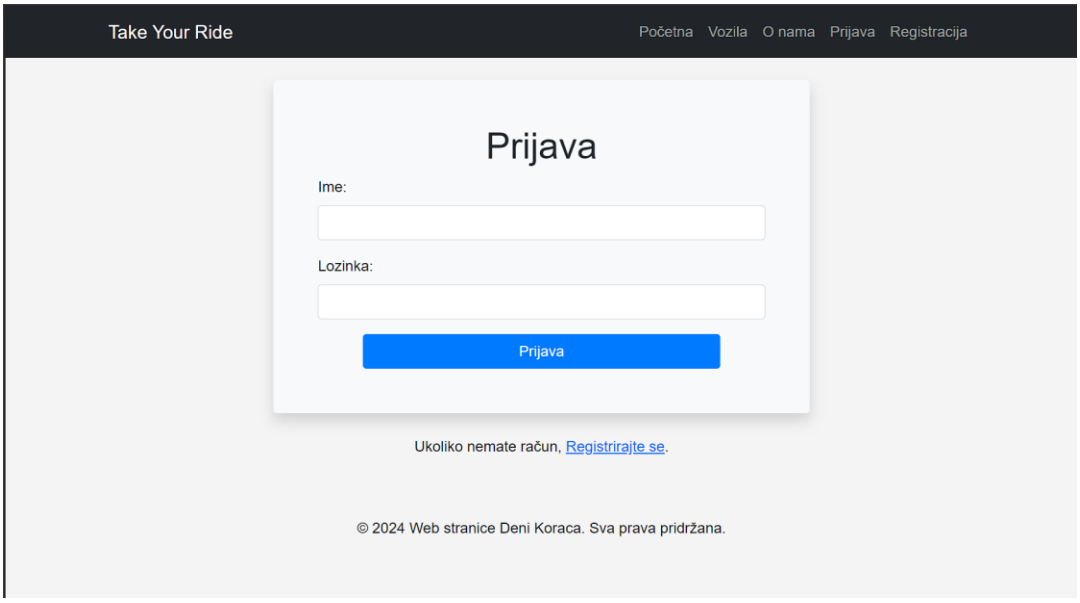
sa sadržajem (engl. *Content*). Kod prikazan u nastavku odnosi se na obrazac za prijavu koji ima postavljeno centrirano poravnanje, naslov *Prijava*¸ polje za unos korisničkog imena, polje za unos lozinke te gumb za slanje obrasca. Za metodu se koristi POST obzirom da se radi o slanju obrasca. Oznaka *{% csrf_token %}* dodaje CSRF token za zaštitu od CSRF napada što je standardna sigurnosna mjera u razvojnom okviru Django. CSRF (engl. *Cross-Site Request Forgery*) predstavlja napad koji tjera autentificirane korisnike da pošalju zahtjev web aplikaciji u kojoj su trenutno autentificirani. Na taj način napadač u ime ovlaštenog korisnika pristupa nekoj aplikaciji [4].

```
<div class="shadow p-5 mb-6 bg-light rounded" style="margin: 0 auto; width: 50%;">
    <div style="text-align: center;">
        <h1>Prijava</h1>
    </div>
    <form method="POST" action="">
        {% csrf_token %}
        <div class="mb-3">
            <label for="exampleInputEmail1" class="form-label">Ime: </label>
            <input type="text" class="form-control" name="username" aria-
describedby="emailHelp">
        </div>
        <div class="mb-3">
            <label for="exampleInputPassword1" class="form-label">Lozinka: </label>
            <input type="password" class="form-control" name="password">
        </div>
        <div style="text-align: center;">
            <button type="submit" class="btn btn-primary">Prijava</button>
        </div>
    </form>
</div>
```

U ovom dijelu koda korištene su CSS klase i stilovi, odnosno Bootstrap. Jedan od primjera je Bootstrap klasa koja dodaje sjenke, margine, svijetlu pozadinu, zaobljene rubove i udaljenost unutar rubova (engl. *Padding*).

Ukoliko korisnik nema korisnički račun, odnosno nije prijavljen, onda je potrebno izvršiti registraciju. Obzirom na prethodno spomenutu mogućnost, korisniku se prikazuje poruka sa poveznicom koja vodi na registraciju korisnika. Poveznica koja se koristi za prosljeđivanje na drugu stranicu koristi oznaku *{% url 'main:register' %}* pomoću koje se generira URL za registraciju. Na posljetku je potrebno zatvoriti blok sa sadržajem pomoću oznake *{% endblock %}*.

Na Slici 5. prikazan je postupak prijave u web aplikaciju. Kako bi se korisnik prijavio, potrebno je unijeti korisničko ime i lozinku te kliknuti na gumb *Prijava.* Ukoliko korisnik nema svoj račun, onda mu se nudi poveznica za registraciju.



Slika 5. Obrazac za prijavu

4.8.4. Registracija korisnika

Kako bi se korisnik mogao prijaviti u aplikaciju potrebno je najprije obaviti postupak registracije korisnika za što se koristi datoteka *register.html*. Ovaj dio koristi Django uzorak jezika (engl. *Django Template Language (DTL))*. Kod koristi „block“ strukturu, odnosno odjeljak za umetanje sadržaja u osnovni predložak, odnosno datoteku *landing.html* što je ostvareno pomoću linije *{% extends "landing.html" %}*. Sve što se nalazi unutar „block“ oznaka će biti umetnuto na odgovarajuće mjesto u *landing.html* datoteci. Idući korak je kreiranje početka bloka, odnosno odjeljka sa sadržajem koji će biti umetnut u „content“ blok osnovnog predloška. Ovaj dio je ostvaren pomoću linije *{% block content %}*.

Idući dio koda odnosi se na provjeru postojanja grešaka u obrascu, odnosno *form.errors.* Ukoliko su greške prisutne, onda se prikazuje upozorenje pomoću Bootstrap klase za oblikovanje poruka upozorenja. Za to se koristi Bootstrap klasa *alert-warning* koja pruža korisniku povratnu informaciju ukoliko je nešto nepravilno popunjeno ili nije uopće popunjeno. Ova klasa ima žuto upozorenje koje može biti zatvorenu pomoću gumba sa klasom *btn-close.* Inače, Bootstrap nudi još neke vrste upozorenja kao što su poruka o uspjehu (engl. *Alert-success*), upozorenje o opasnosti (engl. *Alert-danger),* poruka s informacijama (engl. *Alert-info*) i još mnoga druga upozorenja. Također, svako od navedenih upozorenja ima svoju boju. Primjerice, upozorenje o opasnosti je crvene boje, poruka o uspjehu je zelene boje, dok je poruka s informacijama zeleno-plave boje [5].

Nakon dijela s potencijalnim pogreškama, dolazi dio za registraciju korisnika. Ovaj dio sastoji se od naslova *Register,* CSRF tokena za zaštitu od CSRF napada (prethodno je objašnjeno u odjeljku za prijavu korisnika), prethodno definiranih i objašnjenih metoda – *form1* i *form,* te gumba za slanje obrasca sa vrijednosti *Submit* i Bootstrap klasom *btn btn-primary.* Ova dva obrasca se prikazuju koristeći metodu *as_p* koja svako polje obrasca prikazuje unutar *<p>*

(paragraf) oznaka. Za slanje obrasca koristi se metoda POST. U nastavku je prikazan prethodno opisan programski kod.

```
<div class="shadow p-5 mb-6 bg-light rounded" style="margin: 0 auto; width: 50%;">
  <div style="text-align: center;">
    <h1>Register</h1>
</div>
<form method="POST">
        {% csrf_token %}
        {{ form1.as_p }}
        {{ form.as_p }}
        <br/>
        <div style="text-align: center;">

        <input type="submit" value="Submit" class="btn btn-primary">
        </div>
    </form>
</div>
```

Na samom kraju je potrebno zatvoriti sadržaj bloka koristeći oznaku *{% endbloock %}*.

Što se tiče dizajna, korištene su CSS klase i stilovi. Jedne od njih su Bootstrap klase koje dodaju sjenu, udaljenost od rubova (engl. *Padding*), margine, svijetlu pozadinu i zaobljene rubove. Prethodno spomenute klase su: *shadow, p-5, mb-6, bg-light, rounded*. S druge strane postoje korišteni stilovi koji centriraju tekst i element horizontalno te postavljaju širinu elementa na 50%. Prethodno spomenuti stilovi jesu: *text-align: center; margin: 0 auto; width: 50%.*

4.8.5. Vozila

U nastavku je opisana implementacija pogleda i stranica za renderiranje za pregled svih vozila, pregled detalja o pojedinačnom vozilu, dodavanja vozila, uređivanje vozila te brisanje vozila.

Web stranica služi za prikaz svih vozila koja su dodana u bazu podataka. U datoteci *urls.py* pod klasom *main* definirana je putanja web stranice *vozila path('vehicles/', views.vehicle_list, name='vehicles')*. Putanja koristi funkciju *vehicle_list* iz datoteke *views.py*. Funkcija *vehicle_list* dohvaća vrijednosti *category* i *search_query* pomoću GET zahtjeva. Potom pomoću *vehicles = Vehicle.objects.all()* dohvaća sve objekte koji su iz *Vehicle* modela. Kako bi se prikazala vozila određene kategorije za filtraciju koristi se if izraz gdje se prikazuju sva vozila čija kategorija se podudara s odabranom kategorijom. Pretraživanje se također omogućilo pomoću if izraza gdje se unesena tekstualna vrijednost uspoređuje s markom, modelom i registracijskom oznakom modela. Na samom kraju pomoću *return render(request, 'vehicles.html', {'vehicles': vehicles, 'categories': categories, 'search_query': search_query})* se vraća renderirana stranica koja sadrži filtrirana vozila, dostupne kategorije i pojam koji se pretraživao.

4.8.5.1. Vehicles.html

Ovaj dokument je proširenje za *landing.html* što se omogućuje pomoću *{% extends 'landing.html' %}*. Kako bi se omogućilo korištenje Django *cripsy* predloška za dizajn, na početku dokumenta se učitava *{% load crispy_forms_tags %}*. Postavlja se naslov „Vozila“ u bloku naslov (engl. *Title*). U bloku sadržaja (engl. *Content*) na početku nalazi se Bootstrap klasa *container my-5* koja udaljava od osnovnog predloška (*landing.html*). Potom dolazi naslov stranice „Dostupna vozila“ pomoću H1 elementa s klasom *text-center mb-4* koja centrira naslov

na sredinu zaslona i postavlja marginu ispod elementa. Nakon naslova pomoću elementa obrasca se poziva metoda get. U jednom retku nalaze se izbornik za kategorije, mjesto za pretraživanje, gumb pretraži te gumb za resetiranje filtera. Izbornik je početno postavljen na „Sve kategorije“ te se po želji korisnika može odabrati određena kategorija u padajućem izborniku koji prikazuje sve postojeće kategorije. Tekstualno polje za pretraživanje, pomoću GET zahtjeva *{request.GET.search_query}* omogućuje korisniku da unese vrijednost te pretraži vozilo po svojim kriterijima. U istom retku nalaze se još dva gumba svaki u svom stupcu. Gumb filtriraj koji aktivira pretraživanje po zadanim kriterijima te gumb resetiraj koji vraća filtere na početnu vrijednost tako da nema niti jednog kriterija.

```
<form method="get" class="mb-4">
        <div class="row">
            <div class="col-md-4 mb-2">
                <select name="category" class="form-control">
                    <option value="">Sve kategorije</option>
                    {% for key, value in categories %}
                        <option value="{{ key }}" {% if request.GET.category == key
%}selected{% endif %}>{{ value }}</option>
                    {% endfor %}
                </select>
            </div>

            <div class="col-md-4 mb-2">
                <input type="text" name="search_query" class="form-control"
placeholder="Pretraži vozila" value="{{ request.GET.search_query }}">
            </div>
            <div class="col-md-2 mb-2">
                <button type="submit" class="btn btn-primary btn-block">Filtriraj</button>
            </div>
            <div class="col-md-2 mb-2">
                <a href="{% url 'main:vehicles' %}" class="btn btn-secondary btn-
block">Resetiraj</a>
            </div>
        </div>
    </form>
```

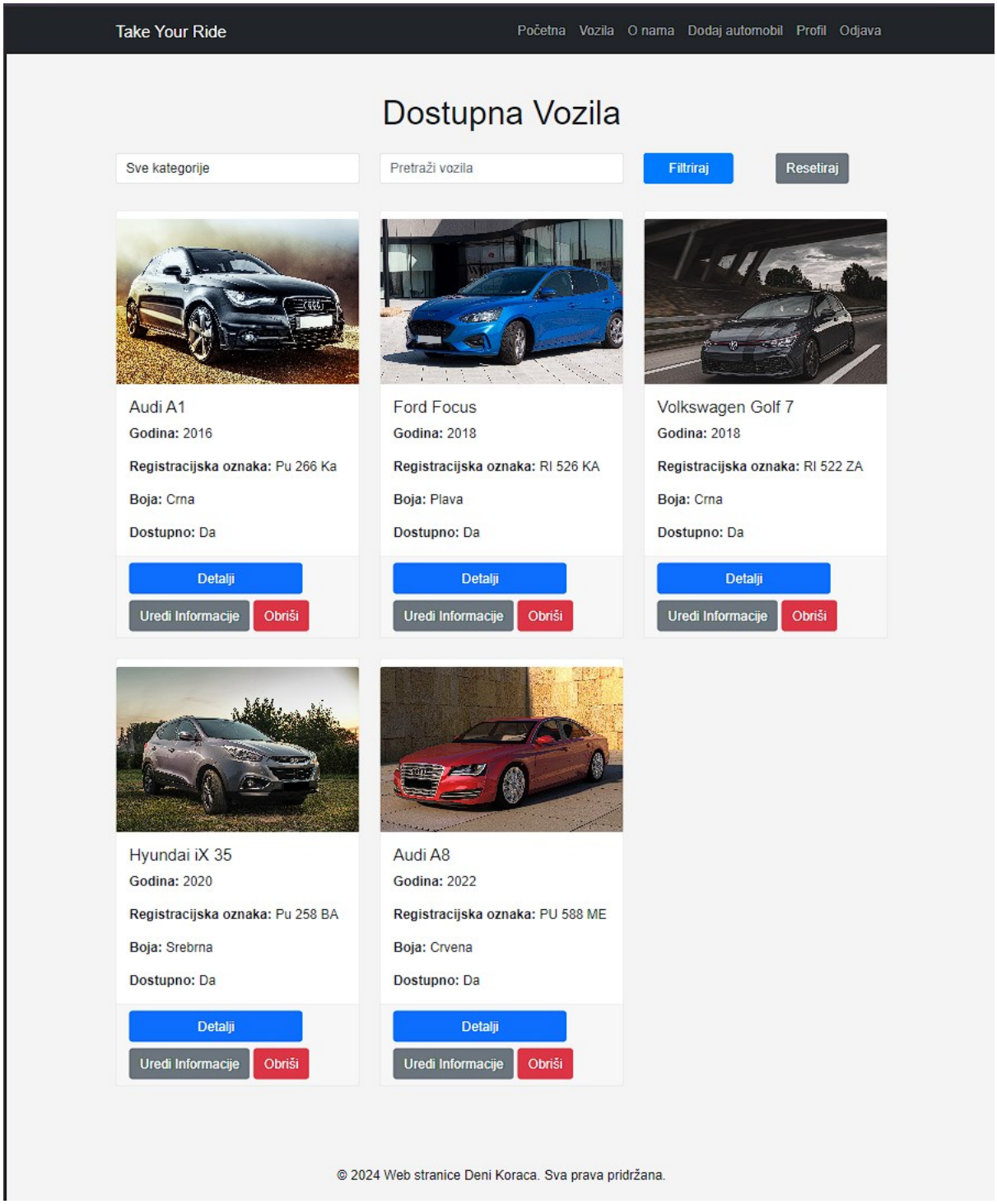
Drugi dio koda služi za prikaz vozila. Sadrži div element s klasom *row* koja s klasom *col-md-4 mb-4* osigurava da svaka kartica na kojoj se nalazi neko vozilo zauzima jednu trećinu širine ekrana. Svi objekti se dohvaćaju pomoću for petlje *{% for vehicle in vehicles %}*. Svako vozilo se nalazi na zasebnoj kartici. Svaka kartica sadrži sliku vozila te osnovne informacije poput naziva marke i modela, godine proizvodnje, registracijske oznake vozila te je li ono dostupno. Također svaka kartica sadrži tri gumba. Gumb detalji pomoću kojeg se prelazi na stranicu s svim informacijama o vozilu koji se može također ispuniti ukoliko je korisnik prijavljen. U slučaju da korisnik nije prijavljen gumb ga vodi na stranicu za prijavu. Ako je prijavljen kao administrator prikazuju mu se dodatna dva gumba. Jedan gumb je za uređivanje podataka, a drugi je namijenjen brisanju vozila. Gumb za brisanje vozila koristi *csrf_token* i dijaloge za potvrdu brisanja.

```
    <div class="row">
        {% for vehicle in vehicles %}
            <div class="col-md-4 mb-4">
                <div class="card h-100">
                    {% if vehicle.image %}
                        <img src="{{ vehicle.image.url }}" class="card-img-top" alt="Slika
vozila {{ vehicle.get_make_display }} {{ vehicle.model }}">
                    {% endif %}
```

```
                    <div class="card-body">
                        <h5 class="card-title">{{ vehicle.get_make_display }} {{
vehicle.model }}</h5>
                        <p class="card-text"><strong>Godina:</strong> {{ vehicle.year
}}</p>
                        <p class="card-text"><strong>Registracijska oznaka:</strong> {{
vehicle.license_plate }}</p>
                        <p class="card-text"><strong>Boja:</strong> {{
vehicle.get_color_display }}</p>
                        <p class="card-text"><strong>Dostupno:</strong> {{
vehicle.is_available|yesno:"Da,Ne" }}</p>
                    </div>
                    <div class="card-footer">
                        {% if user.is_authenticated %}
                            <a href="{% url 'main:vehicle_detail' vehicle.id %}" class="btn
btn-primary btn-block">Detalji</a>
                            {% if user.is_staff %}
                                <a href="{% url 'main:edit_vehicle' vehicle.id %}"
class="btn btn-secondary btn-block mt-2">Uredi Informacije</a>

                                <form action="{% url 'main:delete_vehicle' vehicle.id %}"
method="post" class="d-inline">
                                    {% csrf_token %}
                                    <button type="submit" class="btn btn-danger btn-block
mt-2" onclick="return confirm('Jeste li sigurni da želite obrisati ovo
vozilo?');">Obriši</button>
                                </form>
                                {% endif %}
                        {% else %}
                            <a href="{% url 'main:login' %}?next={% url
'main:vehicle_detail' vehicle.id %}" class="btn btn-primary btn-block">Detalji</a>
                            <a>Potrebna je prijava</a>
                        {% endif %}
                    </div>
                </div>
            </div>
        {% endfor %}
    </div>
```

Na Slici 6. prikazana je lista svih dostupnih vozila. Na samom vrhu se nalazi navigacijska traka, nakon čega slijedi naslov stranice, odnosno „Dostupna vozila“. Korisnik ima mogućnost filtrirati vozila po kategorijama ili ručno unijeti pojam po čemu želi pretražiti. Također, postoji gumb *Resetiraj* koji uklanja sve prethodno postavljene filtere. Zatim slijede vozila gdje se mogu vidjeti osnovne informacije kao što su naziv vozila, godina proizvodnje, registracijska oznaka, boja te dostupnost vozila. Također, postoji gumb za pogledati detalje vozila, urediti postojeće informacije točno određenog vozila ili obrisati pojedino vozilo.



Slika 6. Popis svih vozila

4.8.6. Detalji vozila

U datoteci *urls.py* definira se web adresa stranice. Adresa se sastoji od *vehicle* i *id* objekta vozila kako bi se znalo za koje točno vozilo se prikazuju detalji. Iz datoteke *Views.py* poziva se funkcija *vehicle_detail* koja prije početka ima dodatnu sigurnost pomoću dekoratora koji pazi da korisnik treba biti prijavljen. Nakon toga dohvaća podatke o vozilu. Ukoliko nema

vozila s određenim ID-em, vraća se greška 404. Potom se provjera je li vozilo dostupno. Ukoliko nije prikazuje se poruka greške i ponovno se učitava stranica s popisom vozila.

```
if not vehicle.is_available:
    messages.error(request, "Vozilo trenutno nije dostupno za iznajmljivanje.")
    return redirect('main:vehicles')
```

Potom se provjerava je li korisnik poslao POST metodu. Ukoliko je, znači da je korisnik poslao obrazac za iznajmljivanje. Obrazac automatski popunjava podatke o kojem vozilu se radi i koji korisnik je napravio rezervaciju. Unutra provjere postoji if petlja kojom se provjera je li obrazac ispravan. Potom se provjerava ima li neka druga rezervacija u tom razdoblju. U slučaju da ima prikazuje se poruka o pogrešci.

```
form = RentalForm(request.POST, initial_vehicle=vehicle)
        if form.is_valid():
            new_reservation = form.save(commit=False)
            new_reservation.user = request.user
            new_reservation.vehicle = vehicle
            overlapping_reservations = VehicleReservation.objects.filter(
                vehicle=vehicle,
                start_time__lt=new_reservation.end_time,
                end_time__gt=new_reservation.start_time
            ) | VehicleReservation.objects.filter(
                vehicle=vehicle,
                end_time=new_reservation.start_time
            ) | VehicleReservation.objects.filter(
                vehicle=vehicle,
                start_time=new_reservation.end_time
            )
            if overlapping_reservations.exists():
                messages.error(request, "Auto je već rezerviran za odabrani period.")
            else:
                new_reservation.save()
                messages.success(request, "Vaša rezervacija je uspješno kreirana.")
                return redirect('main:vehicle_detail', vehicle_id=vehicle.pk)
```

Ako nema nikakvih preklapanja i obrazac je ispravan, rezervacija se sprema i prikazuje se poruka o uspješnosti rezervacije. Ukoliko obrazac ima neku grešku, prikazuje se pogreška obrasca. Pogled renderira *vehicle_detail.html* web stranica koja pruža informacije o vozilu i obrazac kojim je moguće izvršiti rezervaciju.

4.8.6.1. Vehicle_detail.html

Struktura datoteke *vehicle_detail.html* sastoji se od dijela zaglavlja gdje se navode meta oznake, naslov i stilske datoteke (*style.css*). Zatim unutar dijela tijela (engl. *Body*) se nalazi navigacijska traka te blok sa glavnim sadržajem (engl. *Block content*). Unutar ovog bloka moguće je vidjeti detalje o vozilu.

```
<div class="container my-5">
    <div class="row">
        <div class="col-md-8 offset-md-2">
            <div class="card mb-4">
                <div class="card-header bg-primary text-white">
                    <h1>{{ vehicle.get_make_display }} - {{ vehicle.model }}</h1>
                </div>
                <div class="card-body">
                    <p><strong>Godina proizvodnje:</strong> {{ vehicle.year }}</p>
                    <p><strong>Registracijska oznaka:</strong> {{ vehicle.license_plate
}}</p>
                    <p><strong>Boja:</strong> {{ vehicle.get_color_display }}</p>
```

```
                    <p><strong>Kategorija:</strong> {{ vehicle.get_category_display }}</p>
                    <p><strong>Cijena po danu:</strong> {{ vehicle.price_per_day }}</p>
                    <p><strong>Dostupno:</strong> {{ vehicle.is_available|yesno:"Da,Ne"
}}</p>
                    {% if vehicle.image %}
                        <img src="{{ vehicle.image.url }}" alt="Slika vozila {{
vehicle.get_make_display }} {{ vehicle.model }}" class="img-fluid">
                    {% endif %}
                </div>
            </div>
```

Zatim slijedi obrazac za iznajmljivanje koji koristi CSRF token zbog sigurnosti. Unutar obrasca je moguće odabrati datum i vrijeme za početak i kraj najma koristeći Tempus Dominus. Na kraju se nalazi izračun ukupne cijene na temelju odabranih datuma i cijene po danu.

```
<form id="rental-form" method="post" action="{% url 'main:vehicle_detail' vehicle.pk %}">
                        {% csrf_token %}
                        {{ form.vehicle }}
                        <div class="mb-3">
                            {{ form.start_time.label_tag }}
                            <div class="input-group date" id="start_time_picker" data-
target-input="nearest">
                                <input type="text" id="start_time" name="start_time"
class="form-control datetimepicker-input" data-target="#start_time_picker"/>
                                <div class="input-group-append" data-
target="#start_time_picker" data-toggle="datetimepicker">
                                    <div class="input-group-text"><i class="fa fa-
calendar"></i></div>
                                </div>
                            </div>
                        </div>
                        <div class="mb-3">
                            {{ form.end_time.label_tag }}
                            <div class="input-group date" id="end_time_picker" data-target-
input="nearest">
                                <input type="text" id="end_time" name="end_time"
class="form-control datetimepicker-input" data-target="#end_time_picker"/>
                                <div class="input-group-append" data-
target="#end_time_picker" data-toggle="datetimepicker">
                                    <div class="input-group-text"><i class="fa fa-
calendar"></i></div>
                                </div>
                            </div>
                        </div>
                        <p><strong>Ukupna cijena:</strong> <span
id="total_price">0.00</span> €</p>
                        <button type="submit" class="btn btn-primary">Rezerviši</button>
                    </form>
```

Može se pojaviti odgovarajuća obavijest ukoliko vozilo nije dostupno kao što je prikazano u slijedećem kodu.

```
{% else %}
            <div class="alert alert-danger mt-4">
                <strong>Napomena:</strong> Ovo vozilo trenutno nije dostupno za
iznajmljivanje.
            </div>
            {% endif %}

            {% if messages %}
                <div class="mt-4">
                    {% for message in messages %}
                        <div class="alert alert-{{ message.tags }} alert-dismissible fade
show" role="alert">
```

```
                            {{ message }}
                            <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
                        </div>
                    {% endfor %}
                </div>
            {% endif %}
```

Kod u nastavku prikazuje kalendar rezervacija.

```
<div class="card mt-4">
                <div class="card-header bg-primary text-white">
                    <h2 style="color: white;">Kalendar Rezervacija</h2>
                </div>
                <div class="card-body">
                    <div id="calendar" style="color: blue;" ></div>
                </div>
            </div>
```

Na kraju HTML koda nalazi se JavaScript skripta. Skripta je podijeljena u pet funkcija namijenjenih za: inicijalizaciju *DataTime Picker*, sinkronizaciju početnog i završnog datuma, izračun ukupne cijene, inicijalizaciju *FullCalendar*-a i automatskog zatvaranja upozorenja. Prva funkcija služi za inicijalizaciju *DataTime Pickera*. U funkciji se postavlja format datuma na 'YYYY-MM-DD' i minimalni datum na današnji dan za početni datum. Dok se za završni datum također postavlja isti format te se onemogućuje postavljanje današnjeg dana.

```
$(function () {
    $('#start_time_picker').datetimepicker({
        format: 'YYYY-MM-DD',
        minDate: moment().startOf('day')  // Postavlja minimalni datum na danas
    });

    $('#end_time_picker').datetimepicker({
        format: 'YYYY-MM-DD',
        useCurrent: false  // Važno! Vidi problem #1075
    })
```

Kako bi se spriječile nekakve nelogičnosti, potrebno je osigurati da nakon postavljanja početnog datuma krajnji datum mora biti poslije početnog. Na isti način nakon postavljanja krajnjeg datuma osigurava se da početni datum mora biti prije krajnjeg datuma.

```
$("#start_time_picker").on("change.datetimepicker", function (e) {
    $('#end_time_picker').datetimepicker('minDate', e.date);
    calculateTotalPrice();
});

$("#end_time_picker").on("change.datetimepicker", function (e) {
    $('#start_time_picker').datetimepicker('maxDate', e.date);
    calculateTotalPrice();
});
```

Treća funkcija izračunava ukupnu cijenu za određeno vrijeme koje je odabrano. Funkcija povlači početni datum i krajnji. Ukoliko uspije povući oba datuma (znači da su oba datuma unesena), onda se izračunava kolika je razlika između oba dana te se ona množi se cijenom vozila po danu. Na kraju se ažuriraju ukupne cijene na stranici.

```
function calculateTotalPrice() {
        var startDate = $('#start_time').val();
        var endDate = $('#end_time').val();
        if (startDate && endDate) {
            var start = moment(startDate, 'YYYY-MM-DD');
```

```
            var end = moment(endDate, 'YYYY-MM-DD');
            var days = end.diff(start, 'days') + 1;  // Dodajemo 1 da uključimo i zadnji
dan
            var pricePerDay = {{ vehicle.price_per_day }};
            var totalPrice = days * pricePerDay;
            $('#total_price').text(totalPrice.toFixed(2));
        } else {
            $('#total_price').text('0.00');
        }
    }
```

U četvrtom dijelu omogućuje se prikaz *FullCalendar-a* te se postavlja izbornik na samom vrhu kalendara pomoću atributa *header* u kojem se dodaje naslov te gumbovi za navigaciju unutar kalendara. S desne strane se nalazi koliko detaljno prikazuje kalendar, odnosno može se birati između mjeseca, tjedna ili dana. Pomoću atributa *events* povlače se postojeće rezervacije. Kako bi se prikazale rezervacije koristi se *eventRender* funkcija koja prilagođava prikaz događaja unutar kalendara. Također, omogućuje se prikaz osobe koja je uradila rezervaciju samo za administratore.

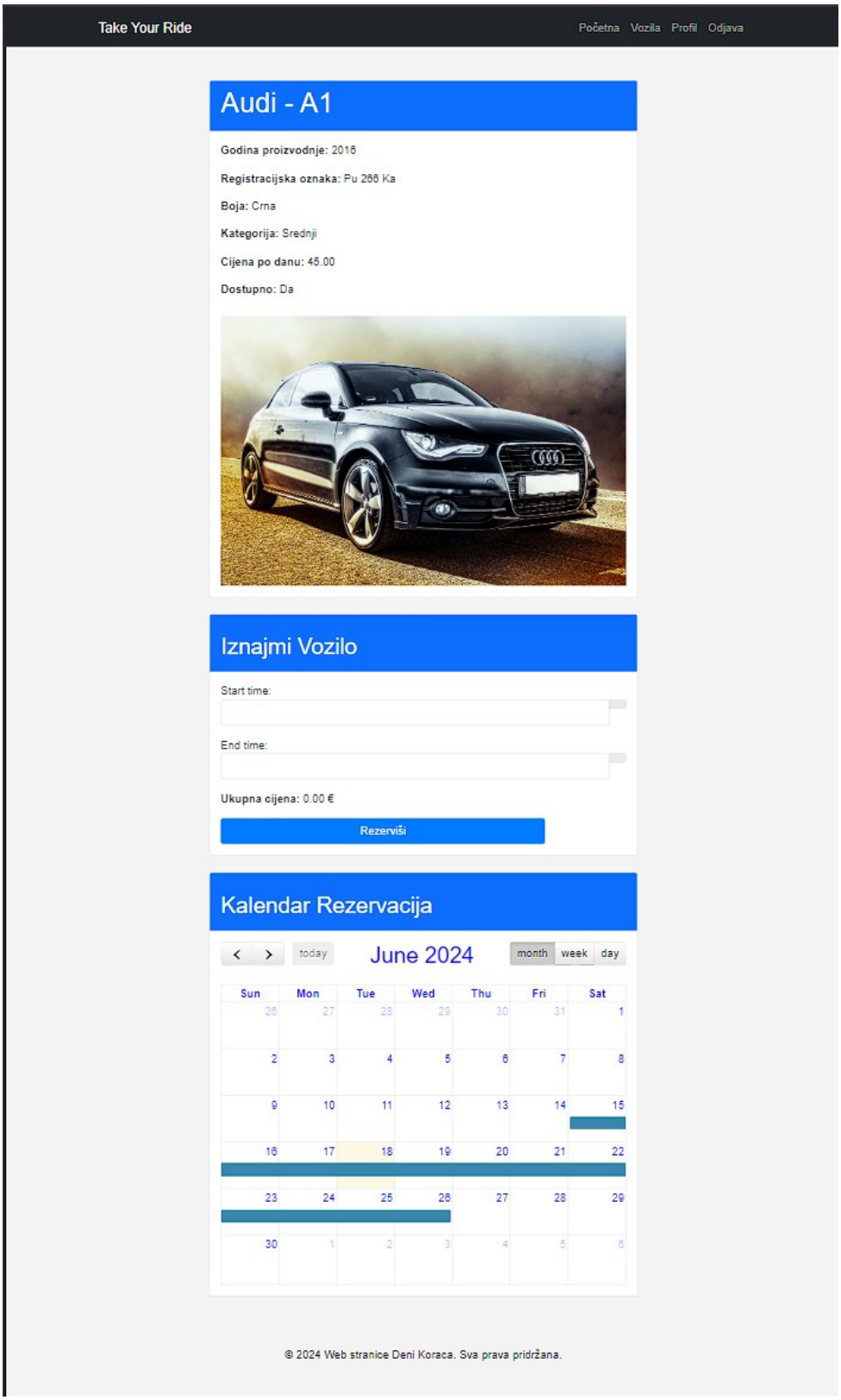
```
    $('#calendar').fullCalendar({
        header: {
            left: 'prev,next today',
            center: 'title',
            right: 'month,agendaWeek,agendaDay'
        },
        editable: false,
        events: {
            url: "{% url 'main:reservation_events' vehicle.id %}",
            type: 'GET',
            error: function() {
                alert('Došlo je do greške prilikom dohvaćanja događaja.');
            }
        },
        eventRender: function(event, element) {
            element.find('.fc-title').append('<br/><span class="fc-description">' +
event.title + '</span>');
            {% if user.is_staff %}
            element.find('.fc-title').append('<br/><span class="fc-user">Korisnik: ' +
event.user + '</span>');
            {% endif %}
        }
    });
```

Posljednji dio skripte postavlja vremensko razdoblje nakon kojeg će se HTML elementi s klasom *alert* automatski zatvorit.

```
    setTimeout(function() {
        $('.alert').alert('close');
    }, 5000);
```

Na Slici 7. prikazane su informacije o pojedinačnom vozilu. Na vrhu se nalazi navigacijska traka. Zatim slijedi naziv vozila te osnovne informacije o vozilu kao što su: godina proizvodnje, registracijska oznaka, boja, kategorija, cijena po danu, dostupnost vozila te njegova fotografija. Potom korisnik ima na raspolaganju odabrati od kad do kad želi iznajmiti vozilo te mu se automatski izračunava ukupna cijena željene rezervacije. Svoju rezervaciju korisnik potvrđuje nakon što klikne na gumb *Rezerviraj.* Na dnu se nalazi kalendar rezervacije unutar kojeg se korisnik može kretati i provjeravati koji termini su slobodni, a koji zauzeti. Također, moguće

je vidjeti tko je rezervirao automobil za koji period. Korisnik se može kretati unutar kalendara. Postoje tri mogućnosti za prikaz kalendara: mjesečni, tjedni i dnevni.



Slika 7. Detalji o vozilu

4.8.6.2. Dodavanje vozila

Definirana je URL putanja koja se koristi za dodavanje novog vozila u datoteci *urls.py* koja poziva funkciju *add_vehicle* iz datoteke *views.py*.

```
path('vehicles/add/', views.add_vehicle, name='add_vehicle')
```

Funkcija *add_vehicle* sadrži if petlju za provjeru je li poslan zahtjev za dodavanje novog vozila putem POST metode. Kreira se obrazac *VehicleForm* u kojem se popunjavaju svi podaci o vozilu. Pomoću if petlje provjerava se ispravnost obrasca te ukoliko je ispravan sprema se novo vozilo. Nakon što korisnik spremi novo vozilo, preusmjerava ga se na stranicu s popisom svih vozila.

```
if request.method == 'POST':
        form = VehicleForm(request.POST, request.FILES)
        if form.is_valid():
            form.save()
            return redirect('main:vehicles')
```

Ukoliko se ne radi o POST zahtjevu, onda se prikazuje prazan obrazac te se ponovno učitava stranica.

Add_vechile.html

HTML datoteka je proširenje na *landing.html* stranicu koja najprije učitava potrebne predloške.

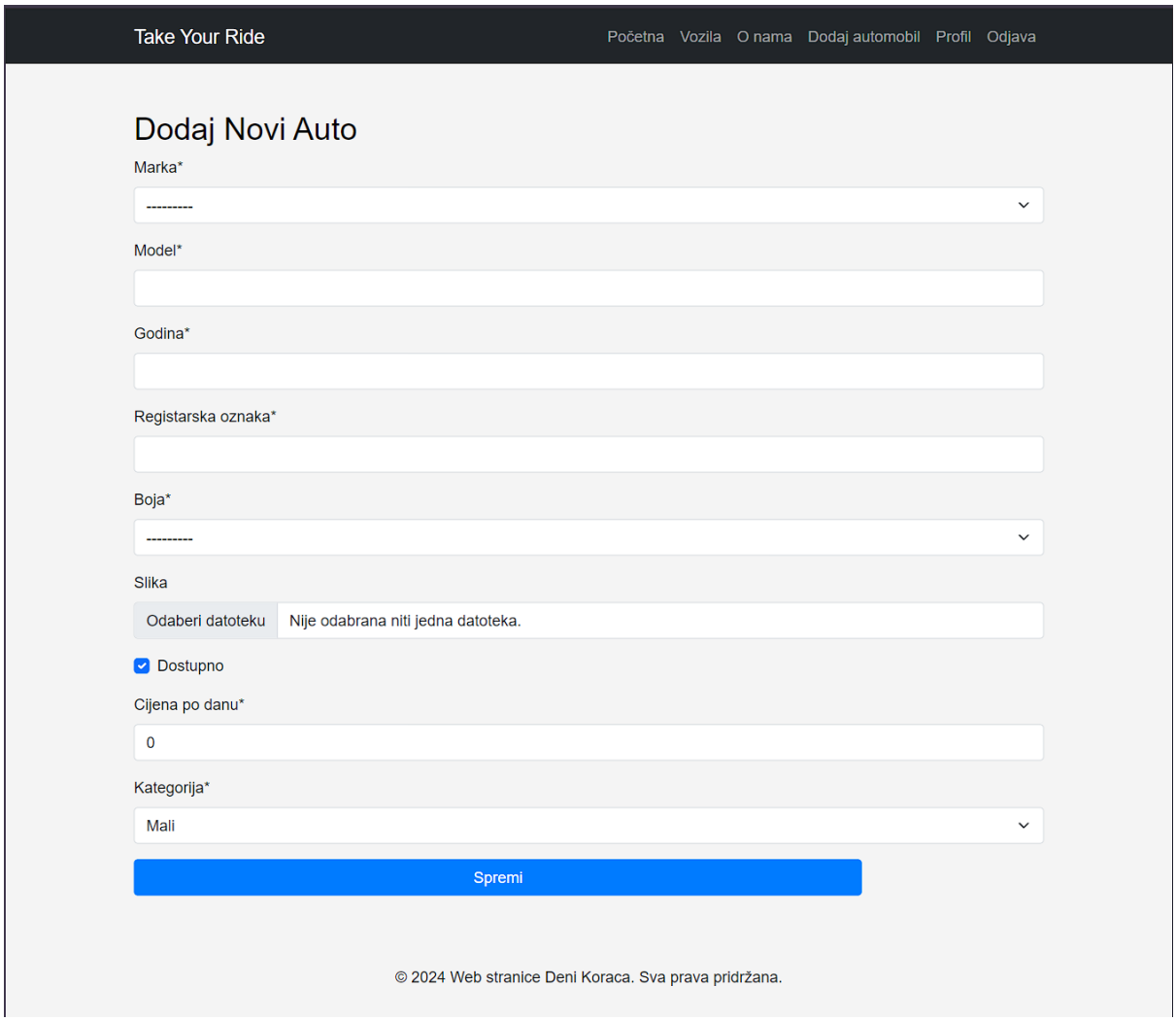
```
{% load static %}
{% load crispy_forms_tags %}
```

Predložak *static* omogućuje korištenje biblioteke koja dopušta pristup statičkim datotekama kao što je CSS. *Crispy_forms_tags* učitava biblioteku koja je dio Django Crispy Forms te omogućuje fleksibilni i lijep prikaz obrazaca. Kako bi naslov bio postavljen na „Dodaj nova vozila“ definiran je u bloku naslov (engl. *Block title*). U bloku za sadržaj definiran je glavni dio stranice, dodan je naslov „Dodaj novi auto“ te je obrazac postavljen na POST metodu kako bi bilo omogućeno slati podatke na server i atributom *enctype* koji omogućuje slanje slike automobila. Pomoću Django predloška uključen je CSRF token za sigurnost. Obrazac je prikazan koristeći Crispy Forms filter. Na kraju se nalazi gumb za slanje obrasca.

```
  <h2 style="color: black;  ">Dodaj Novi Auto</h2>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form|crispy }}
        <button type="submit" class="btn btn-primary">Spremi</button>
    </form>
```

Na Slici 8. prikazan je postupak dodavanja novog vozila. Na vrhu se nalazi navigacijska traka, zatim slijede podaci koje korisnik mora popuniti. Potrebno je unijeti marku, model vozila, godinu izdanja, registracijsku oznaku, boju, sliku, da li je vozilo dostupno, cijenu po danu te kategoriju vozila. Na kraju korisnik potvrđuje unos novog vozila klikom na gumb *Spremi.* Model, godinu izdanja, cijenu po danu i registracijsku oznaku korisnik samostalno unosi. Što se tiče fotografije vozila, nju je potrebno prenijeti. Marku, boju vozila i kategoriju korisnik odabire iz padajućeg izbornika (liste). Ukoliko je vozilo dostupno, korisnik stavlja kvačicu. U

suprotnom ostavlja prazno. Sve podatke osim slike i dostupnosti vozila je obavezno popuniti (označeni su sa zvjezdicom).



Slika 8. Dodavanje novog automobila

4.8.6.3. Uređivanje vozila

Funkcija *edit_vehicle* omogućuje administrativnim korisnicima uređivanje detalja postojećeg vozila. Funkcija dohvaća zahtjev i ID vozila kojeg se želi urediti. Ako objekt ne postoji prema danom primarnom ključu (engl. *Primary key*) koji je zapravo ID vozila onda se javlja HTTP pogreška. Zatim se provjerava metoda POST pomoću *if request.method=='POST'*. Potom slijedi provjera ispravnosti popunjenog obrasca uz *if form.is_valid()*, te se obrazac pohranjuje i događa se preusmjeravanje na stranicu s popisom vozila.

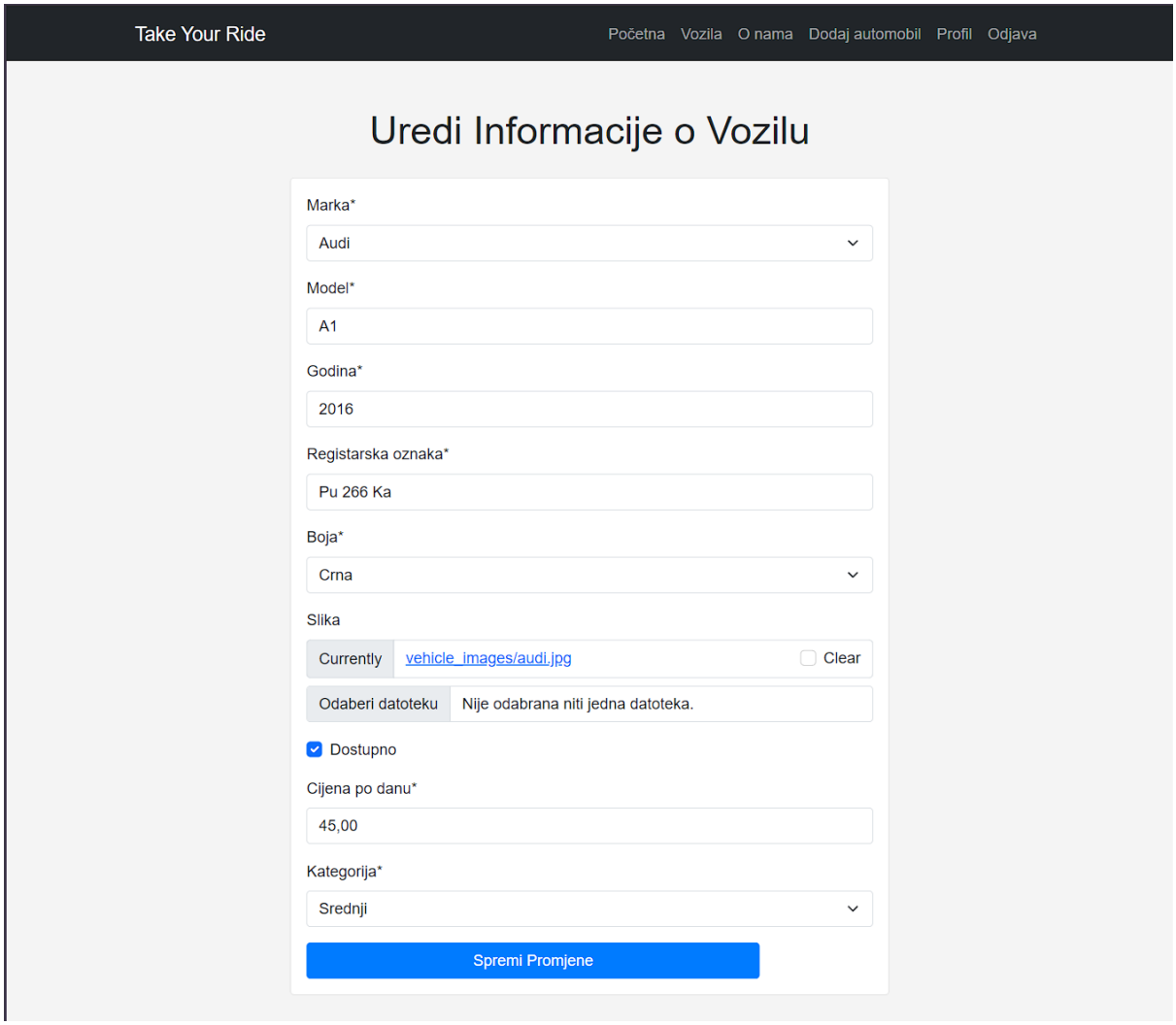
```
def edit_vehicle(request, vehicle_id):
    vehicle = get_object_or_404(Vehicle, pk=vehicle_id)
    if request.method == 'POST':
        form = VehicleForm(request.POST, request.FILES, instance=vehicle)
        if form.is_valid():
            form.save()
            return redirect('main:vehicles')
    else:
        form = VehicleForm(instance=vehicle)
    return render(request, 'edit_vehicle.html', {'form': form, 'vehicle': vehicle})
```

Edit_vehicle.html

Dokument *edit_vehicle.html* koristi se za prikaz stranice na kojoj korisnik može uređivati informacije o vozilu. Koristi se osnovni predložak pomoću naredbe *{% extends "landing.html" %}*. Zatim se učitava *crispy_forms* za stiliziranje obrasca unutar predloška - *{% load crispy_forms_tags %}*. Potom slijedi blok sa naslovom „Uredi vozilo“ te dolazi blok sa sadržajem. Unutar ovog bloka nalazi se Bootstrap kontejner koji dodaje marginu na vrhu i dnu te postavlja centrirani naslov s marginom na dnu. Zatim slijedi Bootstrap kartica za obrazac koja koristi POST metodu i omogućava prijenos podataka pomoću naredbe *enctype="multipart/form-data.* Koristi se CSRF token kako bi se spriječili potencijalni napadi. Prikaz obrasca koristi *crispy_forms* za stiliziranje obrasca. Na dnu se nalazi gumb za spremanje *<button type="submit" class="btn btn-primary">Spremi Promjene</button>*.

```
{% block content %}
<div class="container my-5">
    <h1 class="text-center mb-4">Uredi Informacije o Vozilu</h1>
    <div class="row">
        <div class="col-md-8 offset-md-2">
            <div class="card">
                <div class="card-body">
                    <form method="post" enctype="multipart/form-data">
                        {% csrf_token %}
                        {{ form|crispy }}
                        <button type="submit" class="btn btn-primary">Spremi
Promjene</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```

Na Slici 9. prikazan je postupak uređivanja informacija o vozilu. Na samom vrhu se nalazi navigacijska traka. Zatim slijedi naslov stranice „Uređivanje informacija o vozilu“ te obrazac za izmjenu podataka unutar kojeg se već nalaze postojeći podaci. Potrebno je unijeti marku, model vozila, godinu izdanja, registracijsku oznaku, boju, sliku, da li je vozilo dostupno, cijenu po danu te kategoriju vozila. Na kraju korisnik potvrđuje unos novog vozila klikom na gumb *Spremi promjene.* Model, godinu izdanja, cijenu po danu i registracijsku oznaku korisnik samostalno unosi. Što se tiče fotografije vozila, nju je potrebno prenijeti. Marku, boju vozila i kategoriju korisnik odabire iz padajućeg izbornika (liste). Ukoliko je vozilo dostupno, korisnik stavlja kvačicu. U suprotnom ostavlja prazno. Sve podatke osim slike i dostupnosti vozila je obavezno popuniti (označeni su sa zvjezdicom).



Slika 9. Uređivanje informacija o vozilu

4.8.6.4. Brisanje vozila

U datoteci *urls.py* definira se putanja koja uključuje primarni ključ (engl. *Primary key*) vozila koji se želi obrisati. Kada korisnik pristupa URL putanji, pokreće se funkcija *DeleteVehicleView* iz datoteke *views.py*.

```
path('vehicle/<int:pk>/delete/', views.DeleteVehicleView.as_view(), name='delete_vehicle')
```

Funkcija koristi *DeleteView* metodu koja omogućuje brisanje objekta. Poziva model *Vehicle* koji će se brisati. Pomoću *template_name* poziva HTML dokument koji će se prikazivati za potvrdu o brisanju. Kako bi se nakon uspješnog brisanja objekta vratilo na stranicu s vozilima koristi se *success_url* koji koristi *reverse_lazy* koji odgađa izvršavanje dok to nije potrebno, čime se osigurava da ne dolazi do određenih problema.

```
class DeleteVehicleView(DeleteView):
    model = Vehicle
    template_name = 'vehicle_confirm_delete.html'
    success_url = reverse_lazy('main:vehicles')
```

Metodom *get_queryset* provjera se je li korisnik administrator. Ukoliko nije, onda on neće biti u mogućnosti izbrisati vozilo.

Vehicle_confirm_delete.html

Za prikaz web stranice u HTML datoteci *vehicle_confirm_delete.html* koristi se kao proširenje na *landing.html* stranicu. Postavlja se naslov pomoću *block title* na „Potvrda brisanja“. U bloku sadržaja prikazuje se korisniku poruka želi li sigurno obrisati vozilo. Nakon poruke nalazi se obrazac koji koristi POST metodu za slanje podataka. Dodan je CSRF token zbog sigurnosti i gumb kojim se potvrđuje brisanje ili tekst kojim se poništava brisanje te se vraća popis vozila.

```
{% extends 'landing.html' %}

{% block title %}Potvrda brisanja{% endblock %}

{% block content %}
<div class="container my-5">
    <h1 class="text-center">Jeste li sigurni da želite obrisati ovo vozilo?</h1>
    <div class="text-center mt-4">
        <form method="post">
            {% csrf_token %}
            <button type="submit" class="btn btn-danger">Da, obriši</button>
            <a href="{% url 'main:vehicles' %}" class="btn btn-secondary">Ne, vrati se</a>
        </form>
    </div>
</div>
{% endblock %}
```

4.8.7. Profil

U nastavku je opisano kako funkcionira stranica Profil i njezino uređivanje ukoliko je korisnik prijavljen u web aplikaciju.

4.8.7.1. Stranica profil

Web stranica profila definirana je URL putanjom u *urls.py* datoteci, *path('profile/', views.profile_view, name='profile').* Putanja poziva *profile_view* funkciju iz *views.py* datoteke koja prikazuje profil korisnika, koristeći dekorator *@login_required* koji osigurava da ukoliko korisnik nije prijavljen, onda ne može pristupiti funkciji. Funkcija dohvaća instancu modela Korisnik koji odgovara trenutnom korisniku. U slučaju da takav korisnik ne postoji, stvara ga i postavlja primarnu vrijednost polja. Jedino polje *email_korisnika* povlači vrijednost iz *user* Django modela.

```
@login_required
def profile_view(request):
    korisnik, created = Korisnik.objects.get_or_create(user=request.user, defaults={
        'ime_korisnika': '',
        'prezime_korisnika': '',
        'adresa_korisnika': '',
        'mobitel_korisnika': '',
        'email_korisnika': request.user.email
    })
```

Funkcija generira *profile.html* i prosljeđuje objekt korisničkog profila kako bi se moglo pristupiti html datoteci te je ispisati na stranici.

Profile.html

Datoteka *profile.html* prikazuje web stranicu korisničkog profila koristeći Bootstrap za stilizaciju. Dokument se proširuje na *landing.html* stranicu te naknadno prikazuje korisnikove podatke. U nastavku se nalazi programski kod koji je opisan.

```
{% extends "landing.html" %}
{% load static %}
```

Kako bi se provjerila autentifikacija korisnika u grafičkom sučelju koristi se

{% if user.is_authenticated %}.

Slijedeći dio koda provjerava je li korisnik prijavljen. Nakon što se potvrdi da je prijavljen prikazuje se sadržaj stranice. Koristi se Bootstrap kontejner koji je vertikalno centriran u sredinu.

```
<div class="row justify-content-center">
    <div class="col-lg-8">
        <div class="card mb-4">
            <div class="card-body">
```

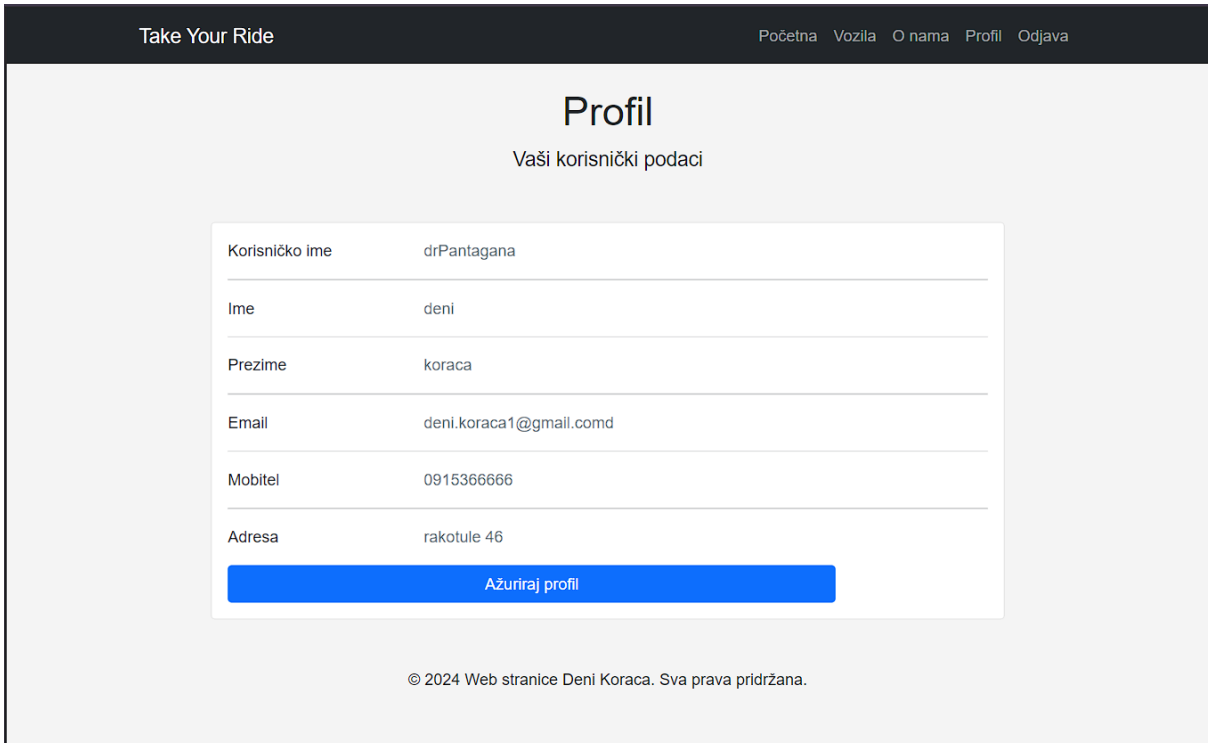
Svi korisnički podaci prikazuju se unutra kartica koja koristi *card-body* kako bi sadržaj bio pravilno oblikovan. Korisnički podaci organizirani su u redove (engl. *Row*) i stupce te raspoređeni u omjeru 1:3.

```
<div class="row">
    <div class="col-sm-3">
        <p class="mb-0">Korisničko ime</p>
    </div>
    <div class="col-sm-9">
        <p class="text-muted mb-0">{{ user.username }}</p>
    </div>
</div>
```

Na kraju kartice nalazi se gumb koji vodi na URL putanju *main:update_profile* gdje je moguće promijeniti svoje podatke.

```
<div class="row mt-3">
    <div class="col-sm-12">
        <a href="{% url 'main:update_profile' %}" class="btn btn-primary btn-
block">Ažuriraj profil</a>
    </div>
</div>
```

Na Slici 10. nalaze se podaci o profilu prijavljenog korisnika. Na samom vrhu se nalazi navigacijska traka, nakon čega slijedi naslov „Profil“ i odgovarajući podnaslov. Što se tiče podataka korisnika, prikazuju se: korisničko ime, ime korisnika, prezime korisnika, adresa elektroničke pošte, broj mobitela te adresa. Također, korisnik ima mogućnost ažurirati svoje podatke ukoliko klikne na gumb *Ažuriraj profil.*



Slika 10. Profil prijavljenog korisnika

4.8.8. Ažuriranje profila

Web stranica ažuriranja profila omogućuje korisniku da izmjeni svoje podatke. U datoteci *urls.py* povezujemo putanju s views.py funkcijom *update_profile* -

path('profile/update/', views.update_profile, name='update_profile').

U *views.py* datoteci najprije imamo dekorator *@login_required* koji osigurava da je korisnik prijavljen. Funkcija najprije pokušava dohvatit trenutnog korisnika te ukoliko to ne uspije, vraća 404 pogrešku. Nadalje, ako korisnik pokušava ažurirati podatke *request.method* dohvaća POST zahtjev. Poziva se obrazac koji je ispunjen korisničkim podacima te ukoliko je obrazac validan nakon izmjena, onda je izmjenu podataka moguće pohraniti i javlja se poruka o uspješnosti pohrane. Ukoliko obrazac nije točan, javlja se greška „Molimo ispravite greške ispod.“. Funkcija vraća stranicu ažuriranja podacima.

```
def update_profile(request):
    korisnik = get_object_or_404(Korisnik, user=request.user)
    if request.method == 'POST':
        form = korisnikForm(request.POST, instance=korisnik)
        if form.is_valid():
            form.save()
            messages.success(request, 'Profil je uspješno ažuriran.')
            return redirect('main:profile')
        else:
            messages.error(request, 'Molimo ispravite greške ispod.')
    else:
        form = korisnikForm(instance=korisnik)

    return render(request, 'update_profile.html', {'form': form})
```

Update_profil.html

Dokument *update_profile.html* je proširenje *landing.html* stranice te koristi crispy_forms za stilizaciju obrasca.

```
{% extends "landing.html" %}
{% load crispy_forms_tags %}
```

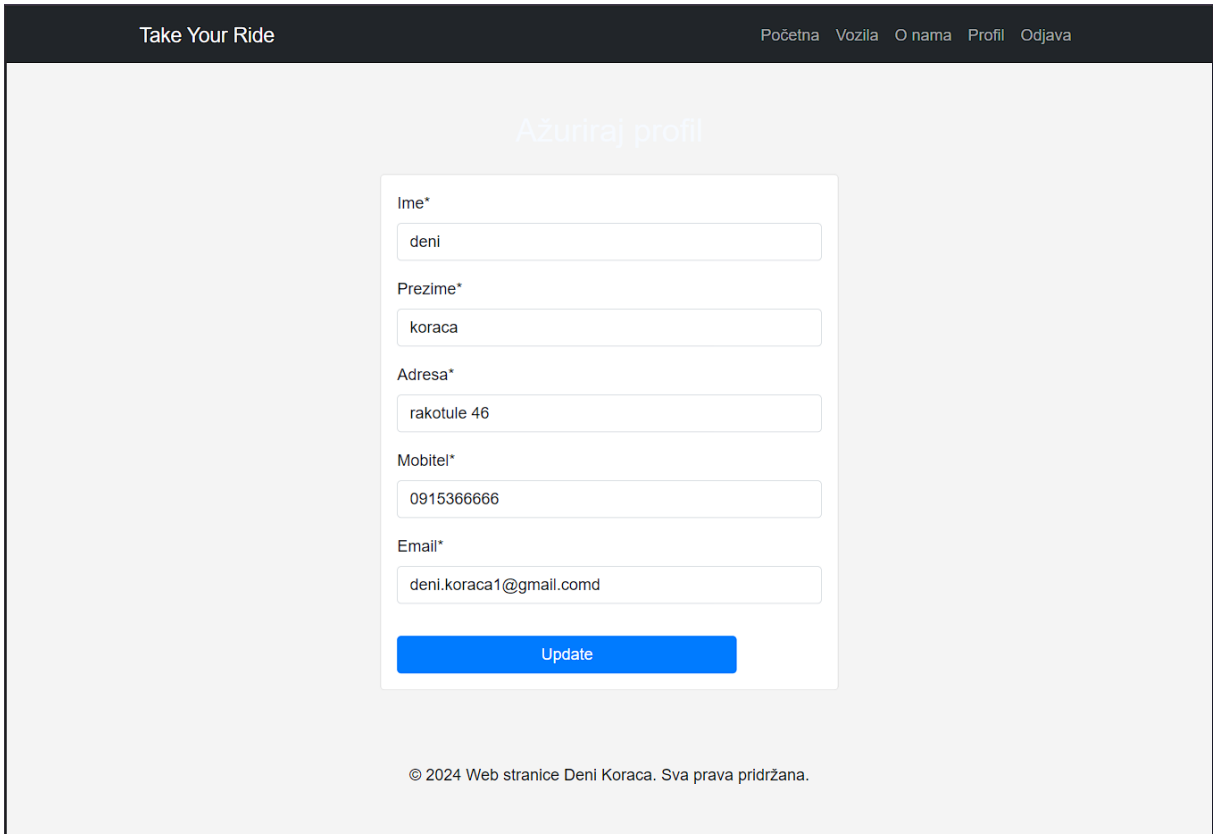
Sekcija s sadržajem stranice nalazi se unutar bloka sadržaja (engl. *Block content*), a na samom vrhu nalazi se naslov „Ažuriraj profil“ koji je uređen pomoću klasa Bootstrap5.

```
<div class="container my-5">
    <h2 class="text-center mb-4">Ažuriraj profil</h2>
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card">
                <div class="card-body">
```

Nakon naslova nalazi se obrazac koji je ispunjen podacima korisnika. Obrazac koristi POST metodu za slanje podataka te uključuje CSRF token za zaštitu od CSRF napada. Za izgled obrasca koristi se *crispy filter* iz *Crispy_forms* biblioteke koji automatski stilizira obrazac. Na kraju stranice nalazi se gumb koji šalje obrazac na obradu kako bi se novi podaci pohranili u biblioteku.

```
<form method="POST" action="">
    {% csrf_token %}
    {{ form|crispy }}
    <button type="submit" class="btn btn-primary btn-block mt-3">Update</button>
</form>
```

Na Slici 11. prikan je obrazac za ažuriranje podataka korisnika. Na vrhu se nalazi navigacijska traka, nakon čega slijedi prethodno spomenuti obrazac. Unutar obrasca korisnik može izmjenjivati podatke kao što su: ime korisnika, prezime korisnika, adresa, broj telefona te adresa elektroničke pošte. Nakon što korisnik unese nove podatke, kako bi se promjene pohranile potrebno je kliknuti na gumb *Update.*



Slika 11. Ažuriranje profila prijavljenog korisnika

5. Zaključak

Unutar ovog završnog rada prikazan je proces izrade aplikacije za najam vozila „Take Your Ride“ u Django razvojnom okviru. Web aplikacija izrađena je u alatu Visual Studio Code-u te Python programskom jeziku.

Unutar same aplikacije razlikuju se 3 vrste korisnika: neregistrirani, registrirani korisnici i administrator. Svaka od vrste korisnika ima svoje određene funkcionalnosti. Primjerice, neregistrirani korisnici mogu gledati informacije o tvrtki i dostupnim vozilima. Registrirani, odnosno prijavljeni korisnici dobivaju mogućnost odabira datuma od kada do kada žele iznajmiti vozilo. Registriranim korisnicima se još nudi mogućnost filtriranja vozila po kategorijama vozila i pretraživanja. Jedino administratori imaju mogućost dodavanja novog vozila, izmjena podataka nekog vozila te brisanja vozila iz baze podataka.

Zbog lakše implementacije obrazaca korištene su dodatne biblioteke iz Django razvojnog okvira. Prva od njih je biblioteka *tempus_dominus* koja se koristi za rad s datumima i vremenom. Omogućava jednostavno dodavanje polja za unos datuma i vremena koristeći pritom razne dodatke. Iduća biblioteka, *crispy_forms,* omogućava jednostavnije i fleksibilnije stiliziranje obrazaca. Na taj način se može lakše koristiti Bootstrap. Glavna prednost je mogućnost dodavanja prilagodljivih poravnanja, margina i raznih drugih stilova bez potrebe za ručnim pisanje HTML-a za svaki obrazac. Posljednja biblioteka, *crispy_bootstrap,* je zapravo dodatak za *crispy_forms* te koristi Bootstrap 5 za izradu predložaka i stilova koje *crispy_forms* može koristiti.

Kreirani su modeli za vozilo te korisnika. Što se tiče modela vozila, on ima svoje odgovarajuće poglede kao što su: popis vozila, brisanje vozila, prikaz detalja o vozilu, iznajmljivanje vozila, dodavanje vozila i uređivanje vozila. Model korisnik ima slijedeće poglede: pogled profila, uređivanje (ažuriranje) profila, prijava korisnika, odjava korisnika te registracija. Svaki od pogleda ima i svoju odgovarajuću HTML stranicu pomoću koje se renderira prikaz kojeg krajnji korisnik vidi.

Razvijenu web aplikaciju moguće je nadograditi na više načina. Prva od mogućnosti je proširenje baze podataka, odnosno dodavanje novih vozila. S obzirom da sve veći broj koristi u većoj mjeri mobilne uređaje, bilo bi dobro napraviti mobilnu verziju aplikacije kako bi im ona uvijek bila nadohvat ruke.

Tijekom izrade ovog seminarskog rada proširio sam svoje znanje o razvojnom okviru Django. Najviše mi se definitivno svidio dio sa programskom implementacijom. U današnje vrijeme na internetu je prisutan veliki broj besplatnih i jako kvalitetnih tutorijala koji objašnjavaju kako implementirati pojedini dio koda ili koristiti programske biblioteke. Naposljetku, jako sam zadovoljan samim tijekom izrade ove web aplikacije kao i krajnjim rezultatom koji je postignut unutar ova tri mjeseca rada.

6. Literatura

[1] »django,« [Mrežno]. Available: https://docs.djangoproject.com/en/5.0/topics/security/. [Pokušaj pristupa 1 lipanj 2024.].

[2] »W3 Schools,« [Mrežno]. Available: https://www.w3schools.com/django/django_intro.php. [Pokušaj pristupa 1 Lipanj 2024.].

[3] A. Kolbina, »Grand Soft,« 8 Svibanj 2018.. [Mrežno]. Available: https://grandsoft.tech/blog/some-facts-about-django/. [Pokušaj pristupa 1 Lipanj 2024.].

[4] »Synopsys,« [Mrežno]. Available: https://www.synopsys.com/glossary/what-is-csrf.html. [Pokušaj pristupa 2 Lipanj 2024.].

[5] »Getbootstrap,« [Mrežno]. Available: https://getbootstrap.com/docs/4.0/components/alerts/. [Pokušaj pristupa 2 Lipanj 2024.].

7. Popis slika

8. Popis priloga

Poveznica na cjelokupni kod aplikacije: https://github.com/Armando259/DeniKoracaZavrsni