IT Academy

Python Development 2019/2020

Globe Tours – Aplikacija za bukiranje turističkih aranžmana

Polaznik : Pavle Đurić

Mentor : Vladimir Marić

Sadržaj

Uputstvo za korisnike .......................................................3

Korisnički interfejs ...................................................................................3

Funkcionalnosti .......................................................................................5

Uputstvo za developere ................................................ 11

Pregled aplikacije ..................................................................................11

Klase (Model ) .......................................................................................12

Pogledi (Views).......................................................................................16

Forme ....................................................................................................24

Templejti ...............................................................................................26

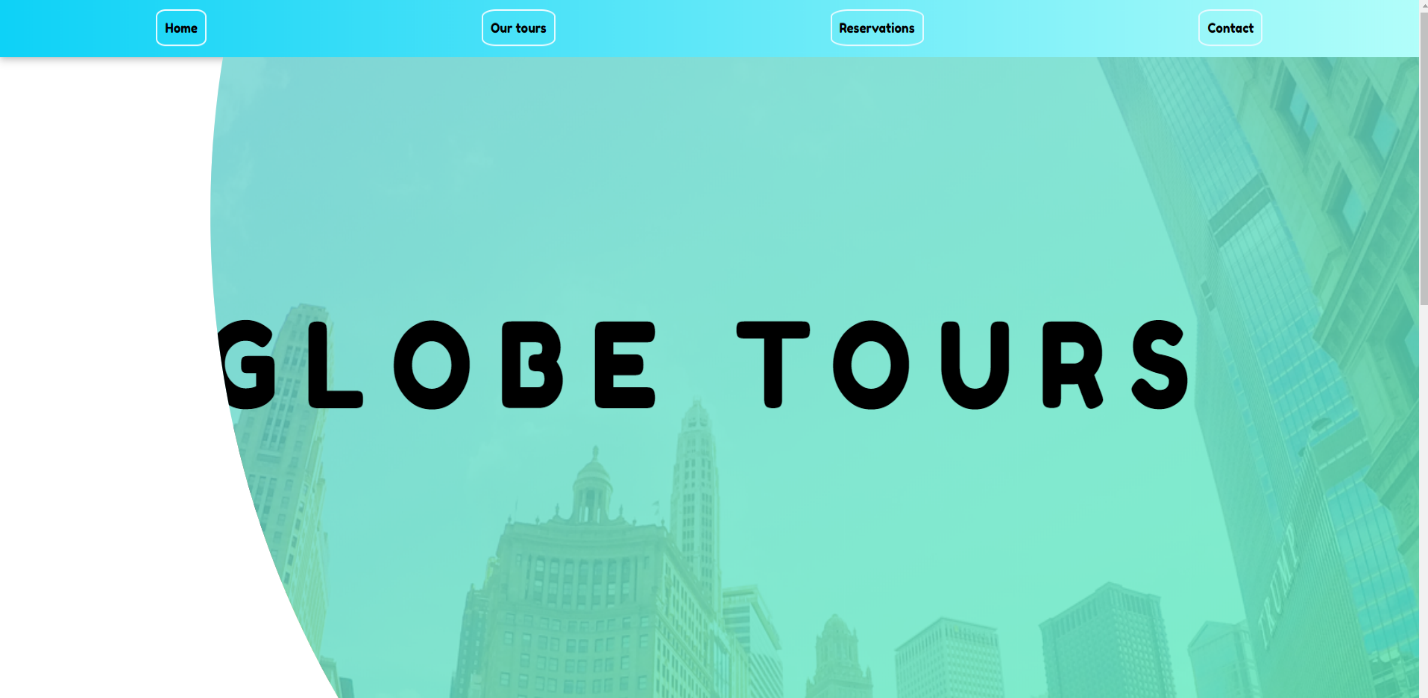
Frontend ................................................................................................30

API .........................................................................................................32

Uputstvo za korisnike

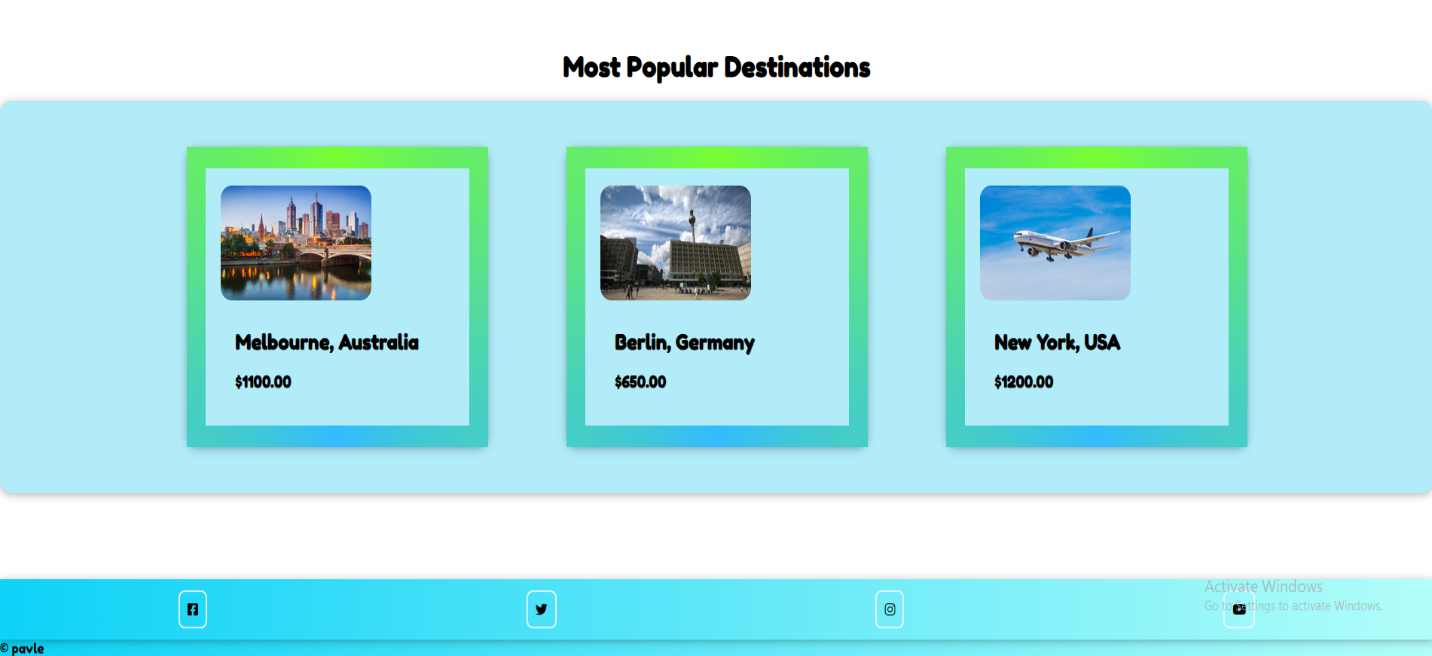
1. KORISNIČKI INTERFEJS

Globe Tours je aplikacija za bukiranje turističkih aranžmana. Korisnik aplikacije će ulaskom na početnu stranicu biti u mogućnosti da koristi nekoliko opcija.



U gornjem navigacionom meniju (Navbar-u) , korisnik ima mogućnost da pogleda dostupne turističke aranžmane klikom na taster -Our Tours-. Klikom na taster –Reservations- , korisnik ima mogućnost da napravi novu rezervaciju, da proveri stanje svoje rezervacije, ili da otkaže rezervaciju. Klikom na taster –Contact- , korisnik ima mogućnost da pošalje poruku sa željenim sadržajem korisničkom servisu Globe Tours-a. Obavezan je da navede svoju mejl adresu, kako bi agencija bila u mogućnosti da odgovori na njegovu poruku.

Daljim skrolovanjem na dole, korisnik ima priliku da vidi koji su trenutno najtraženiji aranžmani koje nudi agencija.



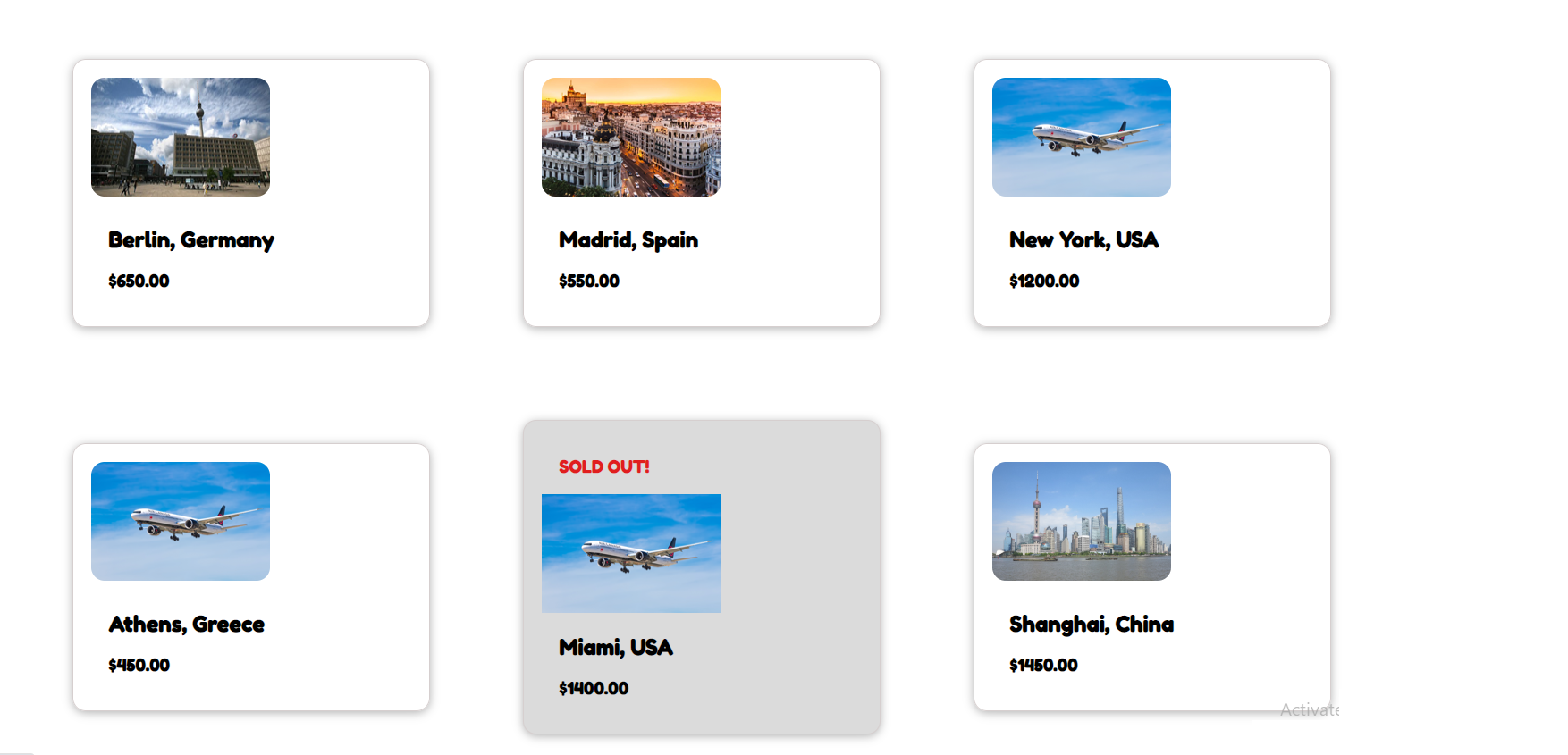
Popularnost destinacija se određuje na osnovu broja realizovanih rezervacija, i taj podatak se dobija direktno iz baze svih rezervacija.

Klikom na jedan od tri najpopularnija aranžmana, korisnik dobija opširnije informacije o tom aranžmanu.

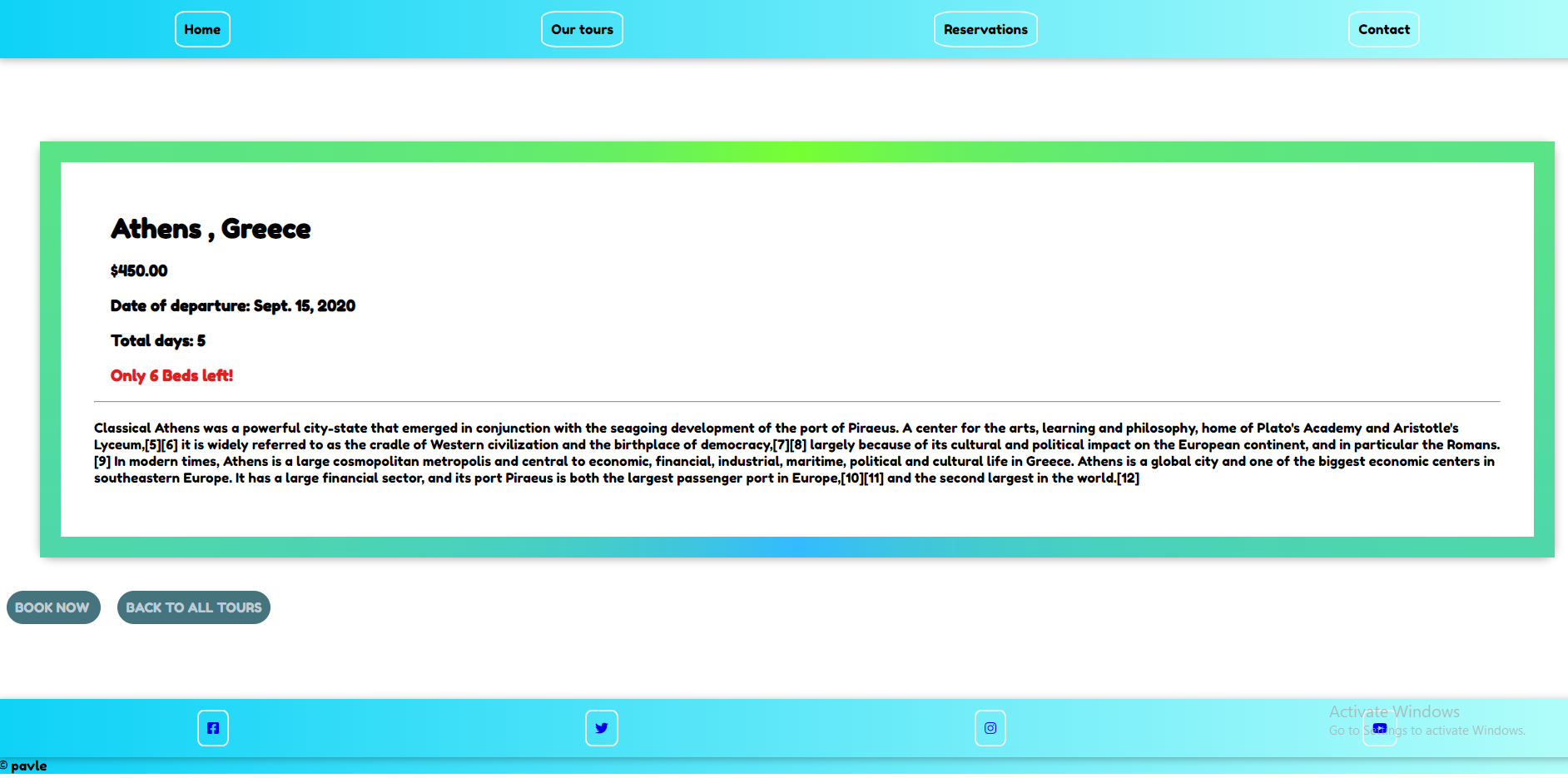
U zaglavlju, korisnik može klikom na jednu od ikonica da pristupi stanicama agencije na nekoj od popularnih društvenih mreža (FB, Instagram, Twitter, Youtube).

1. FUNKCIONALNOSTI

Klikom na taster -Our Tours- , korisniku se prikazuje meni sa svim trenutnim turističkim aranžmanima. Aranžmani su sortirani po datumu unosa u bazu, pa su najaktuelniji aranžmani na dnu stranice.

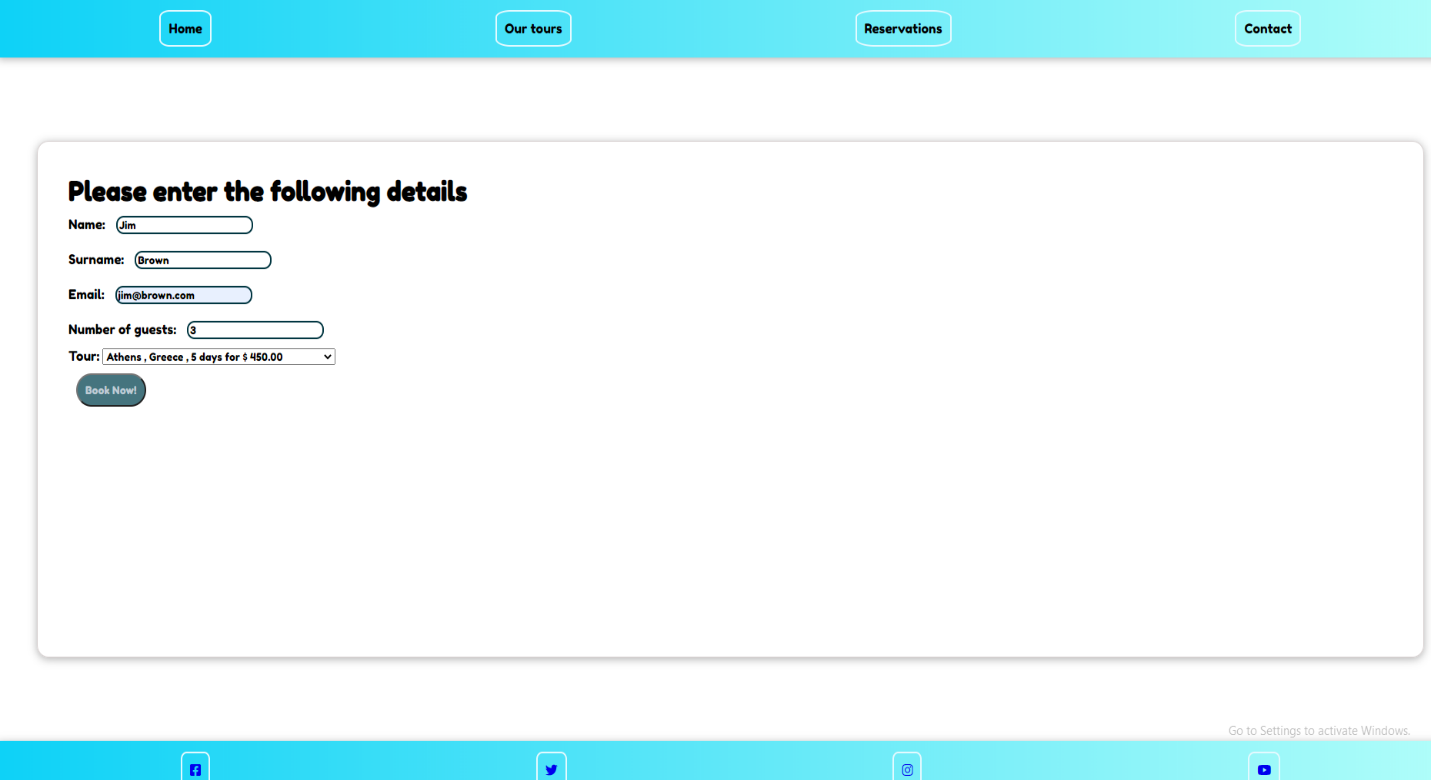


Kao što se vidi na slici, aranžman koji je rasprodan , ima tamnosivu pozadinu i crvenim slovima piše „SOLD OUT“ . Na sve ostale aranžmane, korisnik može kliknuti i o njima dobiti dodatne informacije .



Na detaljnom prikazu aranžmana , mogu se videti naziv destinacije i država, cena aranžmana po osobi, datum polaska , ukupan broj noćenja, kao i broj preostalih mesta. Ukoliko ima manje od 15 slobodnih mesta, taj podatak će biti ispisan crvenim slovima, dok će isti podatak biti ispisan normalnim crnim slovima , ukoliko je broj preostalih mesta veći od 15.

Korisnik u levom uglu sada ima dve opcije. Prva - Book Now - daje korisniku mogućnost da odmah rezerviše svoje mesto . Druga opcija – Back To All Tours – vraća korisnika na stranicu sa celom ponudom aranžmana. Ukoliko se opredeli za opciju trenutnog bukiranja , otvara mu se sledeći interfejs :



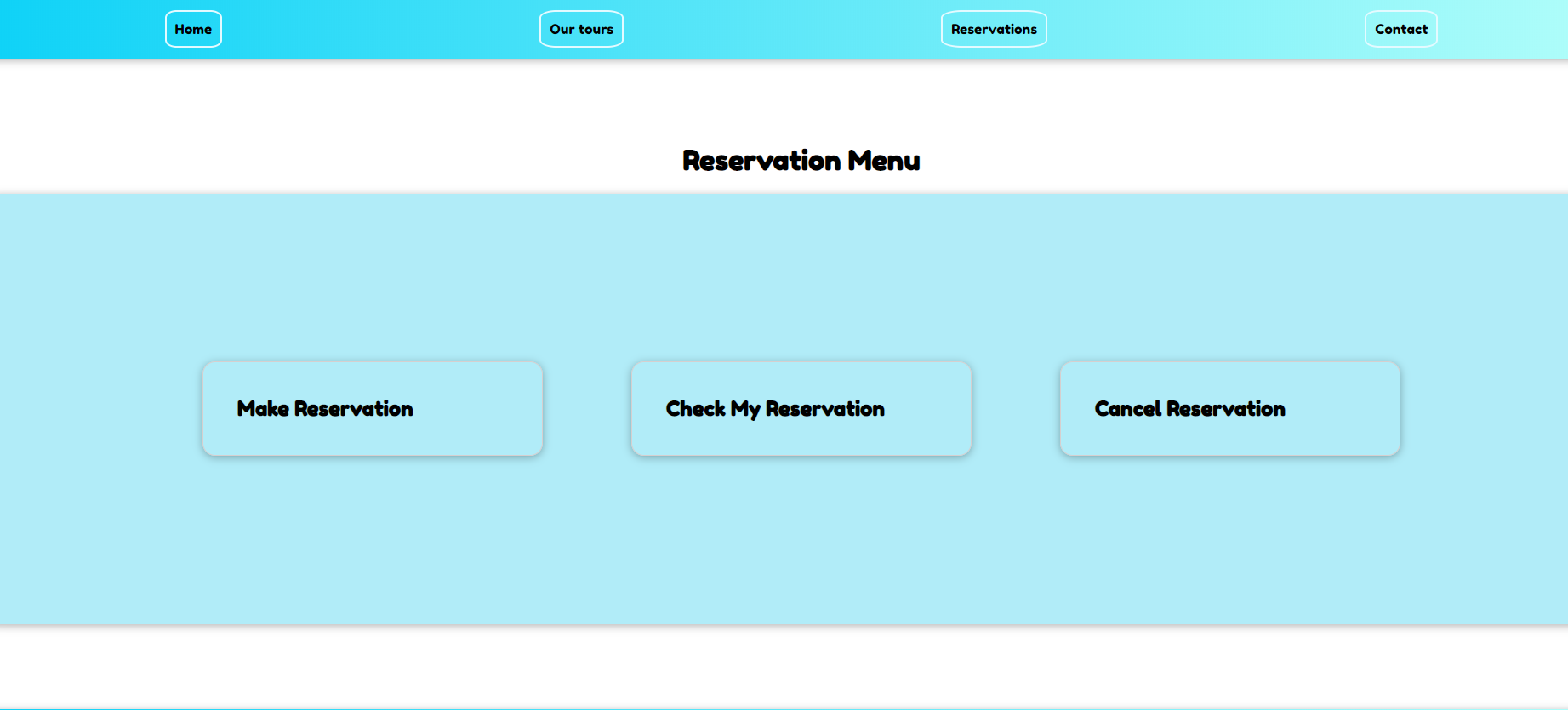
Korisnik zatim popunjava formular, gde navodi ime, prezime i mejl adresu nosioca rezervacije, kao i ukupan broj putnika koji će koristiti rezervaciju, i naravno željeni aranžman.

Ukoliko korisnik ne unese validnu mejl adresu, biće opomenut da to učini. Takođe, ukoliko korisnik unese veći broj putnika nego što ima slobodnih mesta, biće mu skrenuta pažnja da nažalost nema toliko slobodnih mesta za željeni aranžman.

Ukoliko su sva polja formulara validno popunjena, korisnik biva vraćen na početni meni , na kojem će iznad naslova Globe Tours pisati „Thank you (ime korisnika) , your tour to (naziv destinacije) has been booked successfully“ .

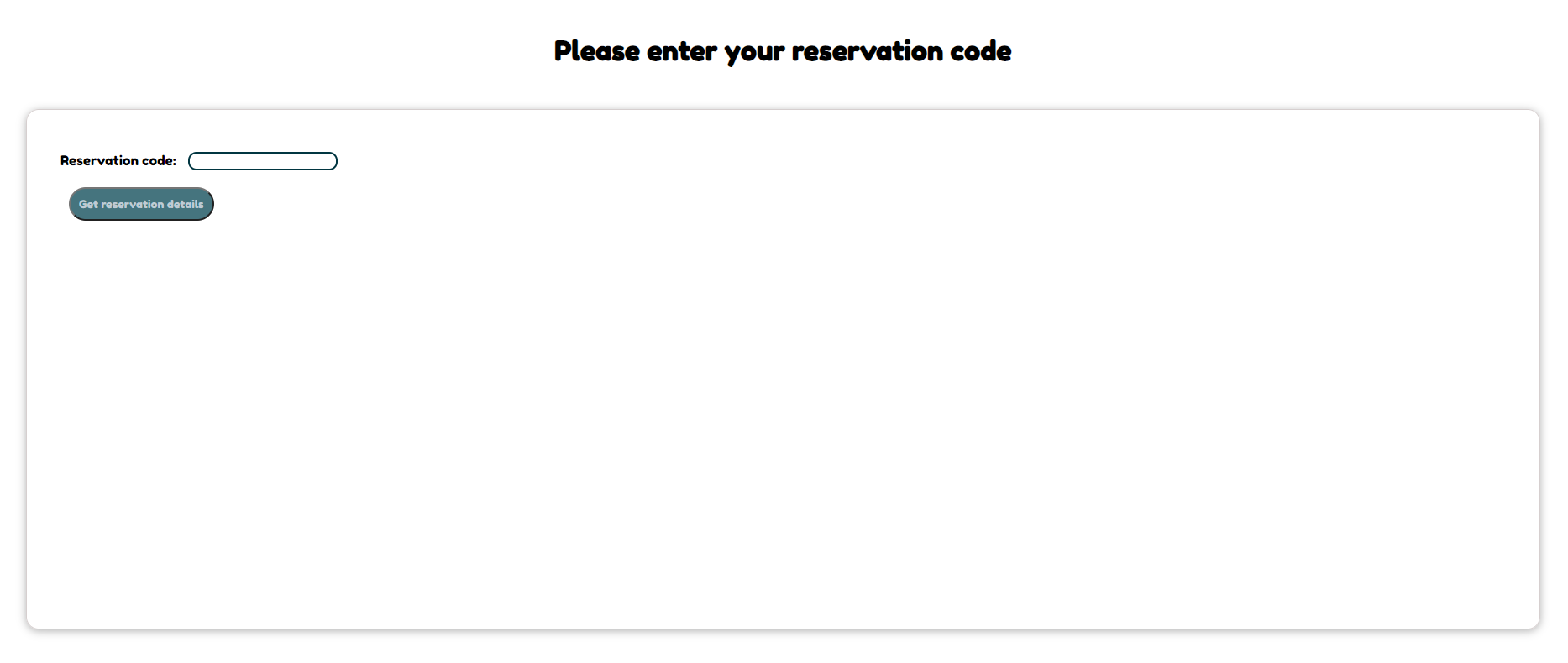
Korisniku će automatski biti poslat mejl sa svim detaljima o rezervaciji, i što je najbitnije , u tom mejlu će dobiti i rezervacioni kod uz pomoć kojeg je moguće da prati stanje svoje rezervacije, ili da eventualno otkaže rezervaciju. Zbog toga je izuzetno važno da korisnik pravilno navede svoju mejl adresu, kako ne bi slučajno njegov rezervacioni kod bio poslat na nečiju tuđu adresu.

Korisnik klikom na taster „Reservations“ ima mogućnost da proveri detalje svoje rezervacije, da napravi novu rezervaciju ili da otkaže postojeću rezervaciju.

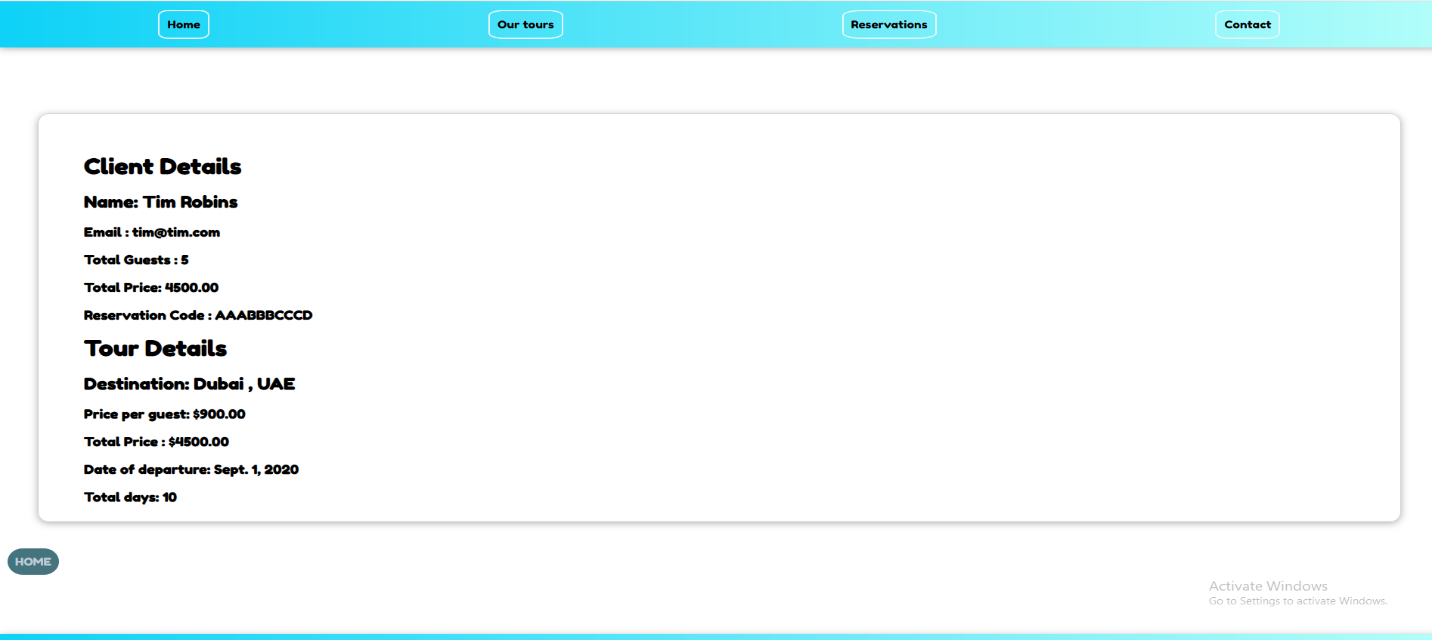


Ukoliko odabere opciju – Make Reservation – aplikacija će ga uputiti na istu stranicu koju vidi pritiskom na taster – Book Now - prilikom pregleda detalja aranžmana.

Ukoliko klikne na opciju – Check My Reservation – otvoriće mu se stranica u kojoj će se od njega zahtevati jedinstveni rezervacioni kod koji je dobio u mejlu prilikom vršenja rezervacije.

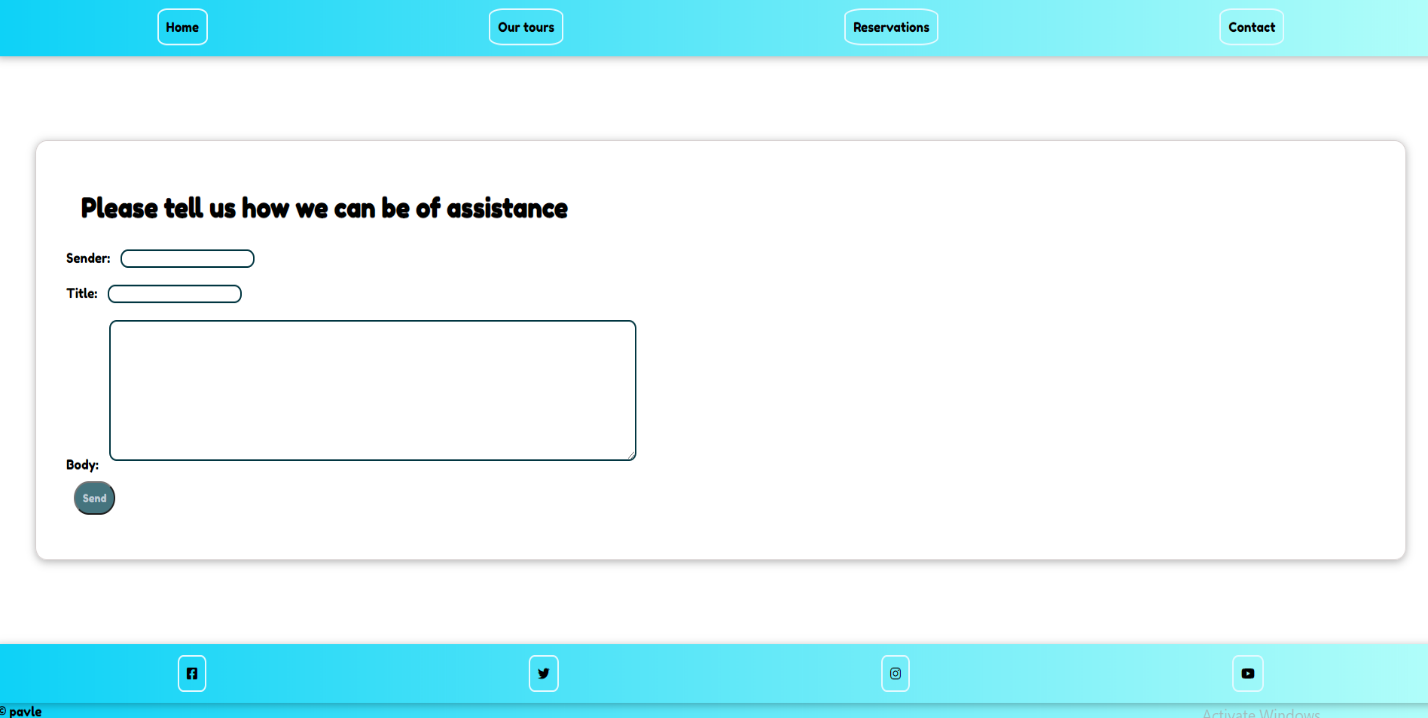


Ukoliko korisnik unese neispravni kod , dobiće upozorenje da kod koji je uneo nije validan. Ukoliko unese ispravan kod, otvoriće mu se stranica u kojoj će videti detalje svog bukiranja:

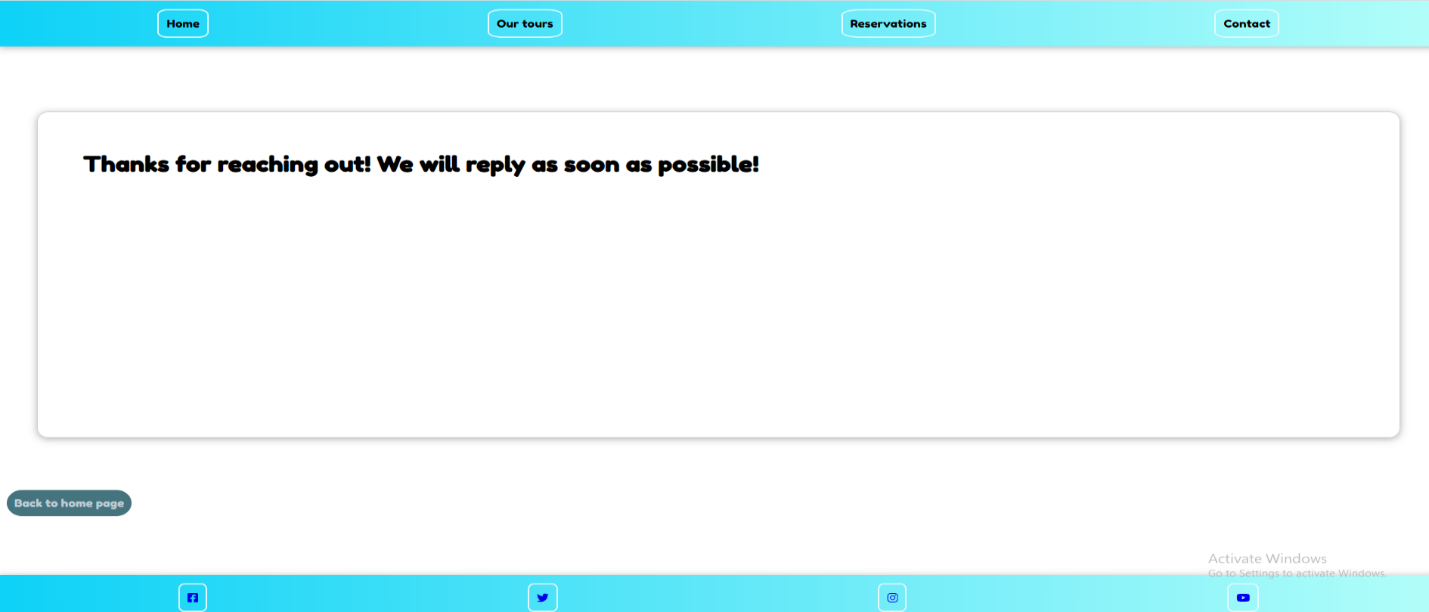


Sličan je proces i kada korisnik odabere opciju – Cancel Reservation – osim što , naravno , umesto detalja o rezervaciji , korisnikova rezervacija biva obrisana iz baze. Zbog ovoga je potrebno biti veoma oprezan prilikom korišćenja ove opcije.

Poslednja funkcionalnost aplikacije jeste – Contact – uz pomoć koje korisnik može poslati mejl korisničkom servisu agencije. Prilikom klika na taj taster otvara mu se sledeći interfejs :



Korisnik popunjava formular sa svojom mejl adresom, naslovom poruke i samom porukom. Prilikom slanja poruke , klikće na taster send i ukoliko je uneo validnu mejl adresu otvara se sledeća stranica:

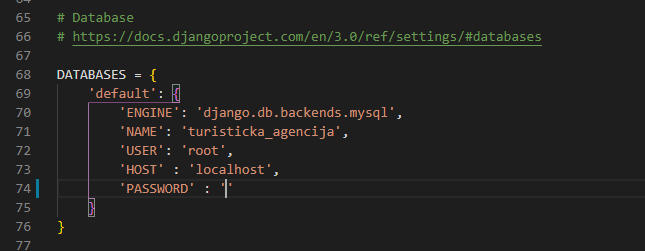


nakon koje korisnik ima opciju povratka na početnu stranicu, ili odabir neke od ostalih tastera iz navigacionog menija.

Uputstvo za developere

1. Pregled aplikacije

Globe Tours je veb aplikacija za bukiranje turističkih aranžmana napisana u Python programskom jeziku upotrebom Django radnog okvira. Za bazu podataka je korišćen MySQL , ali s obzirom da Django koristi svoj ugrađeni objektno-relacioni maper, proces migracije na neku drugu bazu je prilično jednostavan promenom svega nekoliko linija koda u fajlu settings.py .



Zamenom parametara na linijama 70 – 74 sa lakoćom prelazimo na željenu bazu ukoliko za time postoji potreba.

Da bi aplikacija funkicionisala pravilno , pored Python interpretera, potrebno ja da imamo instaliran Django radni okvir, kao i Django Rest Framework biblioteku, uz pomoć koje koristimo API koji ide uz aplikaciju. S obzirom da aplikacija sadrži slike, Django zahteva modul Pillow, a s obzirom da koristimo MySQL moramo imati instaliran i modul mysql-clent. Sve ove module možemo instalirati unosom komande pip install (naziv modula) u komandnoj liniji.

Django radni okvir inače implementira arhitekturalni šablon -Model View Template- koji je takoreći identičan mnogo poznatijem -Model View Controller- šablonu, s time što je u Djangovom slučaju Controller zapravo View, a ono što je kolokvijalno poznato kao View, Django naziva Template. U skladu sa time, u ovom uputstvu ćemo koristiti Djangovu taksonomiju. Аplikaciju pokrećemo tako što u komandnoj liniji unosimo : python manage.py runserver .

Aplikacija koristi port 8000 , ali to se takođe može izmeniti u fajlu settings.py .

1. Klase (Model)

S obzirom da aplikacija koristi već pomenuti ORM , ceo model je napisan u objektno orijentisanom stilu programiranja.

Model je zapravo prilično jednostavan, i sastoji se od svega tri klase:

Tour , Client i ContactMessage .

Tour klasa je klasa čijim se pozivanjem instanciraju istoimeni objekti koji služe za prikazivanje i manipulisanje turističkih aranžmana. Svaka instanca Tour klase mora imati sledeća svojstva :

class  Tour(models.Model):

    destination = models.CharField(max\_length=256)

    country = models.CharField(max\_length=256)

    date\_of\_arrival = models.DateField()

    total\_days = models.IntegerField()

    price = models.DecimalField(decimal\_places=2,max\_digits=15)

    available\_beds = models.IntegerField()

    photo = models.ImageField(upload\_to='photos',default='photos/def.png')

    description = models.TextField(blank=True)

    confirmed\_reservations = models.IntegerField(default=0)

Destination i country služe za opis destinacije turističkog aranžmana i razdvojeni su kako bi u budućnosti mogla da se doda funkcionalnost pretrage po nazivu zemlje. Date of arrival je tip podatka datetime i određuje datum polaska.

Photo varijabla je tipa ImageField , ali to ne znači da ona u sebi sadrži fajl formata slike, već samo apsoulutnu putanju ka tom fajlu. Ta putanja se inače takođe određuje u fajlu settings.py u sledećim linijama :

128 MEDIA\_URL = '/media/'

129 MEDIA\_ROOT = os.path.join(BASE\_DIR,'aplikacija/static')

Confirmed reservations varijabla služi kako bi administrator aplikacije imao uvid u ukupan broj potvrđenih rezervacija , ali takođe i zbog funkcionalnosti – top three destinations – koja se nalazi na početnoj stranici.

Naime , jednostavnim upitom ka bazi , dolazi se do podatka o tri najpopularnije destinacije , kako bi korisniku moglo da se sugeriše destinacija koja je očito omiljena među ostalim korisnicima aplikacije.

Ovaj upit možemo naći u fajlu views.py :

top\_three\_destinations = Tour.objects.filter(confirmed\_reservations\_\_gte=2).order\_by('-confirmed\_reservations')[:3]

Klasa Tour ima i četiri metode koje implementira :

def set\_new\_number\_of\_beds(self,number\_of\_guests):

        self.available\_beds -= number\_of\_guests

        return self.available\_beds

    def cancel\_reservation(self,number\_of\_guests):

        self.available\_beds+=number\_of\_guests

        self.confirmed\_reservations-=number\_of\_guests

    def update\_confirmed\_reservations(self,number\_of\_guests):

        self.confirmed\_reservations+= number\_of\_guests

    def check\_if\_requested\_beds\_are\_avaliable(self,numb\_of\_guests):

        if self.available\_beds < numb\_of\_guests:

            return False

        else:

            return True

Metoda set\_new\_number\_of\_beds služi da se prilikom izvršenja rezervacije, broj slobodnih kreveta za taj aranžman ažurira, odnosno umanji, za broj gostiju koji se nalaze u toj rezervaciji.

Metoda cancel\_reservation se poziva kada se korisnik odluči za otkazivanje rezervacije . Tada se broj slobodnih kreveta uvećava za broj kreveta sa otkazane rezervacije, a broj potvrđenih mesta se umanjuje za isti taj broj.

Metoda update\_confirmed\_reservations se poziva prilikom registrovanja nove rezervacije, i povećava broj potvrđenih rezervacija za broj gostiju sa novoregistrovane rezervacije.

Metoda check\_if\_requested\_beds\_are\_available je važna , jer ukoliko korisnik želi da bukira više kreveta nego što je slobodno, ova metoda će vratiti boolean vrednost False, i korisnik će biti onemogućen da izvrši rezervaciju.

Klasa Client je klasa koja služi za instancijanje objekata korisnika , i ona se instancira onda kada korisnik izvrši rezervaciju nekog aranžmana.

Ona sadrži sledeća polja:

name = models.CharField(max\_length=256)

    surname = models.CharField(max\_length=256)

    email = models.CharField(max\_length=256)

    number\_of\_guests =models.IntegerField()

    tour = models.ForeignKey(Tour,on\_delete=models.CASCADE)

    total\_price = models.DecimalField(decimal\_places=2,max\_digits=15,default=0)

    reservation\_code = models.CharField(max\_length=50,unique=True)

Osim imena,prezimena i mejl adrese, od korisnika se prilikom popunjavanja HTML forme o rezervaciji traži i da unese aranžman koji želi. Nakon uspešno popunjene forme, instancira se objekat Client i odmah se pozivaju metode get\_total\_price i generate\_reservation\_code

 def get\_total\_price(self):

        return self.number\_of\_guests \* self.tour.price

    def generate\_reservation\_code(self):

        chars =['1','2','3','4','5','6','7','8','9','0','A','a','B','b','C','c','D','d','E','e','F','f','G','g','H','h','I','i','J','j','K','k','L','l','M','m']

        self.reservation\_code ="".join(random.sample(chars,10))

        return self.reservation\_code

koje služe da bi se popunila preostala dva polja (total\_price i reservation\_code) . Polje reservation\_code mora biti jedinstven , a s obzirom da je u pitanju polje od 10 karaktera koji se dobijaju od 36 mogućih opcija (pritome je redosled karaktera bitan) , upotrebom kombinatorike, odnosno varijacije dobijamo da je verovatnoća ponavljanja dva ista koda 1/9.22 \* 1014 , što u prevodu znači da je algoritam za generisanje koda zadovoljavajuć s obzirom na očekivani broj rezervacija.

Klasa ContactMessage se instancira kada korisnik upotrebi funkcionalnost slanja poruke u sedište agencije. Ona sadrži sledeća polja:

    sender=models.EmailField()

    title=models.CharField(max\_length=256)

    body =models.TextField()

Dakle , ponovo se radi o HTML formi koja sadrži tri obavezna polja : sender, title i body. U polje sender korisnik unosi svoju mejl adresu, koja mora biti validna (mora sadržati „@“ simbol) . Title i body služe za unos naslova i tela poruke.

Valja napomenuti da se sve promene u models.py fajlu prenose na bazu podataka komandom iz komandne linije : - python manage.py makemigrations – a zatim , ukoliko ne postoje greške – python manage.py migrate

1. Pogledi (Views)

Kao što je navedeno u pregledu aplikacije, taksonomija Djanga je takva da se funkcije koje su ključne za rutiranje aplikacije (često nazivane kontrolerima ) nazivaju pogledi (views) .

Te funkcije se nalaze u fajlu views.py i one su klučne za spajanje podataka iz modela sa korisničnim interfejsom , o kojem će biti reči u poglavlju „Templejti“ .

Django poznaje dva načina definisanja pogleda, pa pored standardnih funkcija, odnosno „function based views“ (FBV), postoji mogućnost implementacija takozvanih „class based views“, odnosno pogleda baziranih na klasama (CBV). Aplikacija Globe Tours implementira i jedne i druge u zavisnosti od pogodnosti koje svaka donosi. CBV pogledi izuzetno pojednostavljuju neke stvari, ali zauzvrat često developeru oduzimaju kontrolu, pa je za situacije gde je potrebna veća specifičnost ipak preporučeno da se koriste FBV pogledi.

Globe Tours ukupno implementira šest FBV pogleda i četiri CBV pogleda.

Krenimo redom :

def HomeView(request):

    top\_three\_destinations = Tour.objects.filter(confirmed\_reservations\_\_gte=2).order\_by('-confirmed\_reservations')[:3]

    if request.method=="GET":

        return render(request,'aplikacija/home.html',{'top\_three':top\_three\_destinations})

    else:

        form= ClientForm(request.POST)

        if form.is\_valid():

            name = form.cleaned\_data['name']

            surname=form.cleaned\_data['surname']

            email=form.cleaned\_data['email']

            tour\_id=form.cleaned\_data['tour'].pk

            number\_of\_guests=form.cleaned\_data['number\_of\_guests']

            total\_price=number\_of\_guests\*form.cleaned\_data['tour'].price

            tour\_name=form.cleaned\_data['tour'].destination

            with transaction.atomic():

                tour=Tour.objects.select\_for\_update().filter(destination=tour\_name)[0]

                beds\_available = tour.check\_if\_requested\_beds\_are\_avaliable(number\_of\_guests)

                if beds\_available:

                    tour.set\_new\_number\_of\_beds(number\_of\_guests)

                    tour.update\_confirmed\_reservations(number\_of\_guests)

                    tour.save()

                    client = Client.objects.create(name=name,surname=surname,email=email,number\_of\_guests=number\_of\_guests,tour\_id=tour\_id,total\_price=total\_price)

                    reservation\_code = client.generate\_reservation\_code()

                    client.save()

                    email\_message = f'Thank you {name} {surname} for booking your trip to {tour\_name}\n\n Here are the details:\n Reservation Code : {reservation\_code}\nNumber of guests: {number\_of\_guests}\nTotal Price: {total\_price}\nDate of arrival: {tour.date\_of\_arrival}\n\nBest regards,\n GLOBE TOURS'

                    try:

                        send\_mail(f'Globe tour reservation to {tour\_name}',email\_message,settings.EMAIL\_HOST\_USER,[email,],fail\_silently=False)

                        message =f'thank you {name} ! Your reservation to {tour} for {number\_of\_guests} guests , with a total price of {total\_price} has been booked successfully.'

                        return render(request,'aplikacija/home.html',{'msg':message,'top\_three':top\_three\_destinations})

                    except:

                        message =f'thank you {name} ! Your reservation to {tour} for {number\_of\_guests} guests , with a total price of {total\_price} has been booked successfully.\n GOOGLE BLOCKED THE MAIL!?!'

                        return render(request,'aplikacija/home.html',{'msg':message,'top\_three':top\_three\_destinations})

                else:

                    form = ClientForm()

                    return render(request,'aplikacija/client\_form.html',{'infoMsg':f"sorry, there are less than {number\_of\_guests} beds available for {tour\_name}",'form':form})

HomeView je FBV koji se poziva kada korisnik dođe na početnu stranicu (metoda GET), i kada korisnik submit-uje HTML formu sa rezervacijom, nakon koje biva vraćen na početnu stranu (metoda POST) .

Kao što se vidi kada je metoda GET , funkcija je prilično jednostavna, dok kod metode POST stvari se malo komplikuju. Ono na šta treba obratiti naročito pažnju jeste sledeći kod:

with transaction.atomic():

     tour=Tour.objects.select\_for\_update().filter(destination=tour\_name)[0]

Ovde se radi o transakciji u bazi podataka, čime je onemogućeno da dva korisnika simultano izvrše rezervaciju istog aranžmana , čime bi se aplikacija izložila riziku tzv. „Race Condition“- . Gore navedenim kodom se implemetira tzv. „Row Level Lock“ unutar baze čime je transakcija obezbeđena. Dakle ne postoji mogućnost da se desi situacija gde neki aranžman ima recimo tri slobodna mesta, i da dva korisnika istovremeno rezervišu ta tri mesta.

Od funkcionalnosti ovog pogleda, takođe bih izdvojio kod :

send\_mail(f'Globe tour reservation to {tour\_name}',email\_message,settings.EMAIL\_HOST\_USER,[email,],fail\_silently=False)

čime se šalje mejl sa jedinstvenim rezervacionim kodom na adresu korisnika koji je bukirao aplikaciju. Da bi ova izuzetno važna funkcionalnost bila moguća, potrebno je dodati sledeće linije u fajl settings.py :

DEFAULT\_FROM\_EMAIL = 'globetours011@gmail.com'

SERVER\_EMAIL = 'globetours011@gmail.com'

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_USE\_TLS = True

EMAIL\_HOST = 'smtp.gmail.com'

EMAIL\_HOST\_USER = 'globetours011@gmail.com'

EMAIL\_HOST\_PASSWORD = 'SIFRA IDE OVDE'

EMAIL\_PORT = 587

Ukoliko je mejl adresa na gmail hostu, potrebno je takođe unutar gmail aplikacije omogućiti slanje mejla od strane „third party apps“ jer Gugl inače ovo iz bezbednosnih razloga ne dozvoljava.

ThanksView je takođe FBV i on se poziva kada korisnik submituje HTML formu koja šalje poruku agenciji, čime dobija nazad stranicu sa zahvalnicom za slanje poruke. Forma ima action parametar podešen na rutu „/thanks“ .

def thanksView(request):

    form=ContactForm(request.POST)

    if form.is\_valid():

        sender=form.cleaned\_data['sender']

        title=form.cleaned\_data['title']

        body=form.cleaned\_data['body']

        message=ContactMessage.objects.create(sender=sender,title=title,body=body)

        message.save()

        try:

            send\_mail(title,f'from: {sender} \n {body}',settings.EMAIL\_HOST\_USER,[settings.EMAIL\_HOST\_USER,],fail\_silently=False)

            return render(request,'aplikacija/thanks.html')

        except:

            return (request,'aplikacija/whoops.html')

    else:

        form = ContactForm()

        return render(request,'aplikacija/contactmessage\_form.html',{'form':form,'infoMsg':"please enter a valid email address"})

Kao što vidimo iz koda, ukoliko je forma validno popunjena , instancira se objekat klase ContactMessage, koja se popunjava parametrima iz forme. Metodom message.save() , ovaj obekat se pomoću ORM-a čuva u bazi podataka. Već poznata funkcija send\_mail() šalje mejl ovaj put od korisnika ka agenciji ,s time što agencija faktički sama sebi šalje mejl, a korisnikov mejl se navodi u telu poruke, jer aplikaciji naravno nije dozvoljen pristup mejlu korisnika.

ReservationView je FBV koji se poziva kada korisnik odabere taster „Reservations“ (GET metoda) , i kada se opredeli da otkaže rezervaciju submitovanjem HTML forme gde unosi svoj rezervacioni kod (POST metoda) .

def reservationView(request):

    if request.method == "GET":

        return render(request,'aplikacija/reservations.html')

    else:

        reservation\_code = request.POST.get('reservation\_code')

        client\_exists = Client.objects.filter(reservation\_code=reservation\_code)

        if client\_exists:

            client = client\_exists[0]

            tour = client.tour.destination

            tour\_object = Tour.objects.filter(destination=tour)[0]

            total\_guests = client.number\_of\_guests

            tour\_object.cancel\_reservation(total\_guests)

            tour\_object.save()

            client.delete()

            return render(request,'aplikacija/reservations.html',{'infoMsg':f'You have successfully cancelled your trip to {tour}'})

        else:

            return render(request,'aplikacija/cancelReservation.html',{'infoMsg':'INVALID RESERVATION CODE'})

Kao što se vidi iz koda, ukoliko klijent unese neispravan rezervacioni kod, biva vraćen na stranicu otkazivanja rezervacije uz napomenu „INVALID RESERVATION CODE“ . Ukoliko je unos rezervacionog koda validan, aktivira se metod „cancel\_reservation“ klase Tour a instanca klase Client se metodom „delete“ briše iz baze podataka.

Pogledi checkReservationView i cancelReservationView su prilično slični , s time što se jedan poziva odabirom tastera Check reservation, a drugi odabirom tastera cancel reservation i oba podržavaju samo GET metodu i vraćaju odgovarajuće templejte :

def checkReservationView(request):

    return render(request,'aplikacija/checkReservation.html')

def cancelReservationView(request):

    return render(request,'aplikacija/cancelReservation.html')

Poslednji od FBV pogleda jeste ReservationDetailView koji se poziva kada korisnik submituje HTML formu u kojoj navodeći svoj rezervacioni kod zahteva od aplikacije da vidi detalje svoje rezervacije:

def reservationDetailView(request):

    code = request.POST.get('reservation\_code')

    client\_exists = Client.objects.filter(reservation\_code=code)

    if client\_exists:

        return render(request,'aplikacija/detailedReservation.html',{'client':cli ent\_exists[0]})

    else:

        return render(request,'aplikacija/checkReservation.html',{'infoMsg':'INVALID RESERVATION CODE'})

Ovaj pogled proverava ispravnost rezervacionog koda, i zatim vraća templejt sa detaljima rezervacije (ukoliko je rezervacioni kod ispravan) ili natrag na stranicu gde se unosi rezervacioni kod uz napomenu „INVALID RESERVATION CODE“.

Prelazimo sada na poglede koji se zasnivaju na klasama , od kojih je prvi TourListView :

class TourListView(ListView):

    model = Tour

    def get\_queryset(self):

        return Tour.objects.all()

Kao što se vidi iz koda, ne može jednostavnije. Klasa nasledi klasu ListView koja se nalazi u modulu django.generic.views . ListView je idealan kada želimo jednostavno da dođemo svih instanci određene klase, odnosno svih redova određene SQL tabele. Samo navedemo koji model nas interesuje, i metodom get\_queryset navedemo da li želimo sve objekte ili postoji neki eventualni filter. Dobijamo natrag promenljivu koja je tip podatka lista kroz koju možemo lako da iteriramo u templejtu.

Ako je CBV pogled TourListView delovala jednostavno , šta onda reći za TourDetailView pogled:

class TourDetailView(DetailView):

    model = Tour

Dakle nasleđuje se klasa DetailView, koja je takođe iz modula django.generic.views. Samo se unese model koji nas interesuje, i uz jednu liniju koda Django će odraditi ostalo za Vas. Ono što je bitno da se napomene kod CBV pogleda , jeste da se prilikom navođenja njihovih ruta u fajlu urls.py , nije dovoljno samo navesti ime pogleda (kao što je to slučaj sa FBV pogledjima) već se mora pozvati i metoda „as\_view“ . Dakle, to bi u urls.py fajlu izgledalo ovako :

url(r'^tour/(?P<pk>\d+)$',views.TourDetailView.as\_view(),name='tour\_detail'),

Kod DetailView-a je naravno potrebno i navesti neki identifikator po kojem će pogled znati koji je konkretan objekat u pitanju. To je najčešće primarni ključ tabele i on se može uneti upotrebom regularnih izraza kao što je navedeno u kodu, mada Django od verzije 2.2 podržava i drugačiji način rešavanja ovog problema, pa se umesto mukotrpnog regularnog izraza može upotrebiti i elegantnije rešenje :

path('tour/<int:pk>',views.TourDetailView.as\_view(),name='tour\_detail'),

Dakle , umesto čitavog onog zbunjujućeg šablona, samo se u uglastim zagradama navede tip podatka (u ovom slučaju int) , a zatim se prosledi pk kao identifikator za primarni ključ.

Sledeći CBV pogled je CreateClientView koji nasleđuje klasu CreateView:

class CreateClientView(CreateView):

    redirect\_field\_name = "/aplikacija/home.html"

    model = Client

    form\_class = ClientForm

Ono što je zanimljivo jeste da CreateView osim modela, traži i redirect\_field\_name koji ukazuje na templejt koji treba aktivirati prilikom redirekcije, i form\_class koji će pozvati objekat HTML forme koji treba generisati prilikom pozivanja ovog pogleda. Kada je već pomenuto, Django osim tradicionalnih HTML formi ima sopstvenu objektnu implementaciju formi . Ove forme u vidu klasa se pišu u posebnom fajlu koji se zove forms.py i tu je proces pisanja forme još jednostavniji nego što bi inače bio. Valja napomenuti, da se opet zarad veće jednostavnosti, gubi određen nivo kontrole, pa je neki put dobro implementirati i tradicionalnu HTML formu. O Django Formama će svakako biti reči u sledećem poglavlju.

Pored CreateClientView pogleda , još jedan CBV pogled nasleđujue CreateView klasu , a to je contactView:

class contactView(CreateView):

    redirect\_field\_name="/aplikacija/thanks.html"

    model = ContactMessage

    form\_class=ContactForm

Ona se poziva kada korisnik želi da pošalje poruku agenciji, pa se kao što vidimo poziva model ContactMessage, kao i Django Forma ContactForm.

1. Forme

Kao što smo već napomenuli , Django radni okvir ima svoj način konstruisanja HTML formi tako da developer ne mora da ih piše sam unutar templejta već ih definiše na sledeći način:

from aplikacija.models import Client,ContactMessage

from django import forms

class ClientForm(forms.ModelForm):

    class Meta:

        model = Client

        fields = ('name','surname','email','number\_of\_guests','tour')

        widgets = { 'name' : forms.TextInput(attrs={"class":"clientFormField"}),

                    'surname' : forms.TextInput(attrs={"class":"clientFormField"}),

                    'email' : forms.TextInput(attrs={"class":"clientFormField"}),

                    'number\_of\_guests' : forms.NumberInput(attrs={"class":"clientFormField"}),

        }

class ContactForm(forms.ModelForm):

    class Meta:

        model = ContactMessage

        fields= ('sender','title','body')

        widgets = {

            'sender': forms.TextInput(attrs={'class': 'clientFormField'}),

            'title': forms.TextInput(attrs={'class': 'clientFormField'}),

             'body': forms.Textarea(attrs={'class': 'clientTextField'}),

        }

Klasa koju pravimo nasledi klasu ModelForm iz modula django.forms . Zatim unesemo model na koji se forma odnosi, kao i polja koja želimo da budu vidljiva prilikom popunjavanja forme. Widgets promenjiva služi da navedemo evenutalno ako želimo da polja forme ukrasimo nekim CSS klasama kao što je urađeno ovde.

U ponekim situacijama je ipak bolje koristiti tradicionalne HTML forme . Django ima razumevanja i za takve situacije , pa podatke iz forme možemo na taj način dobiti uz pomoć request objekta koji je ugrađen u Django . To smo videli iz poglavlja „Pogledi „ u kodu :

        reservation\_code = request.POST.get('reservation\_code')

1. Templejti

Django radni okvir koristi sopstveni template engine koji neodoljivo podseća na neke druge template engine popularne u Pythonu poput Jinja2 . Templejti služe da bismo na prijatan način korisniku prikazali podatke iz modela, a bez velike muke za developera . S obzirom da je danas „korisnično iskustvo“ od presudnog značaja za popularnost aplikacije, ovaj segment, kao ni CSS stilizaciju, nipošto ne treba potceniti.

Globe Tours ima ukupno dvanaest templejta, ali pogledaćemo samo nekoliko.

Glavni templejt, bez kojeg bi se teško moglo raditi jeste onaj koji je zajednički za sve templejte i najčešće se naziva base.html :

<!DOCTYPE html>

{% load static %}

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Tour Agency</title>

    <link href="https://fonts.googleapis.com/css2?family=Fredoka+One&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="{% static 'css/style.css' %}">

    <script src="https://kit.fontawesome.com/6f44727d31.js" crossorigin="anonymous"></script>

</head>

<body>

    <nav id="nav" class="skyBlue">

        <ul class='navList'>

            <li><a href="{% url 'home' %}">Home</a></li>

            <li><a href="{% url 'tour\_list' %}">Our tours</a></li>

            <li><a href="{% url 'reservations' %}">Reservations</a></li>

            <li><a href="{% url 'contact' %}">Contact</a></li>

        </ul>

    </nav>

    <main>

    {% block content %}

    {% endblock %}

</main>

    <footer id="footer" class="skyBlue">

        <ul class="navList">

            <li><a href="#"><i class="fab fa-facebook-square"></i></a></li>

            <li><a href="#"><i class="fab fa-twitter"></i></a></li>

            <li><a href="#"><i class="fab fa-instagram"></i></a></li>

            <li><a href="#"><i class="fab fa-youtube"></i></a></li>

        </ul>

        <p>&copy; pavle</p>

    </footer>

</body>

</html>

Dakle, sve što on sadrži , sadrže i svi ostali templejti, s time što se imeđu {% block content %} i {% endblock %} nalazi ono po čemu se ostali templejti razlikuju.

Tour\_detail templejt je onaj koji koji se iscrtava kada se pozove onaj famozni CBV pogled sa jednom linijom – TourDetailView .

Na njegovom primeru možemo uočiti neka pravila Djangovog Template Engine-a :

{% extends 'aplikacija/base.html' %}

{% block content %}

<div class="tourDetailTab mostPopularBorder">

    <h1>{{tour.destination}} , {{tour.country}}</h1>

    <h3>${{tour.price}}</h3>

    <h3>Date of departure: {{tour.date\_of\_arrival}}</h3>

    <h3>Total days: {{tour.total\_days}}</h3>

    {% if tour.available\_beds <= 15 %}

    <h3 class="redText">Only {{tour.available\_beds}} Beds left! </h3>

    {% else %}

    <h3> Beds left: {{tour.available\_beds}} </h3>

    {% endif %}

    <hr>

    <div id="description">

    <p>{{tour.description}}</p>

</div>

</div>

<a class="btn" href="{% url 'book\_tour' %}"> BOOK NOW </a>

<a class="btn" href="{% url 'tour\_list' %}">BACK TO ALL TOURS</a>

{% endblock %}

Prvo, vidimo da se na vrhu nalazi komanda kojom se naglašava da ovaj templejt nasleđuje koreni templejt base.html . Ključna reč „extends“ se inače ne pojavljuje nigde u Python jeziku , i po svemu sudeći, pozajmljena iz Java programskog jezika. Zatim se unutar block contenta vide neke linije koje već malo više liče na Python kod. Imamo dot operator , koji služi za pristup variablama unutar nekog objekta, kao i neizostavni if/else , s time što ovde možemo videti da se usled izostanka indentacije unutar html fajla (na osnovu kojeg Python interpeter inače ume da prepozna kraj bloka) , kraj if kondicionala obeležava se sa komandom {% endif %} .

Vidimo i da unutar <a> taga umesto normalne vrednosti za href atribut, imamo ponovo sintaksu sa {% url ‘naziv urla’ %} . Ovo je Djangov način da ukaže na url time što se u polje pod jednostrukim navodima unosi naziv url-a koji definišemo u fajlu urls.py argumentom “name “ kao na primer:

path("booktour/",views.CreateClientView.as\_view(),name="book\_tour")

Navešćemo samo još jedan templejt, a to je tour\_list templejt :

{% extends 'aplikacija/base.html' %}

{% block content %}

<table>

{% for tour in tour\_list %}

{% if forloop.counter0|divisibleby:3 %} <tr> {% endif %}

    {% if tour.available\_beds > 0 %}

    <td>

<div class="tourTab">

<a href="{% url 'tour\_detail' pk=tour.pk %}">

{% load static %}

<img src='{{ tour.photo.url }}' alt="photo" width="200" height="133">

<h2>{{tour.destination}}, {{tour.country}}</h2>

<h3>${{tour.price}}</h3>

</a>

</div>

</td>

    {% else %}

    <td>

    <div class="tourTabSold">

        <h3 class="redText">SOLD OUT!</h3>

        {% load static %}

        <img src='{{ tour.photo.url }}' alt="photo" width="200" height="133">

        <h2>{{tour.destination}}, {{tour.country}}</h2>

        <h3>${{tour.price}}</h3>

        </a>

        </div>

    </td>

    {% endif %}

{% if forloop.counter|divisibleby:3 or forloop.last %} </tr> {% endif %}

{% endfor %}

</table>

{% endblock %}

Ono što je kod ovog templejta upadljivo jeste upotreba for petlje kojom vršimo iteraciju kroz listu. Ponovo moramo, kao i kod if kondicionala, naglasiti kraj petlje sa {% endfor %} . Zanimljiv je i kod :

{% if forloop.counter|divisibleby:3 or forloop.last %} </tr> {% endif %}

Ovde vidimo upotrebu takozvanog pajpa , kojim se pozivaju određeni metodi na objektima. Ovde je pozvan metod divisiblebly , ali mogu se koristiti razni ugrađeni metodi u zavisnosti od tipa podatka .

1. Frontend

Kao što smo već naveli, dopalo se to backend developerima ili ne, korisničko iskustvo je jedno od presudnih faktora uspeha svake aplikacije. Tu se naravno ne sme potceniti brzina aplikacije (gde dobro urađen backend svakako dolazi do izražaja) , ali izgled elemenata koji se prikazuju korisniku kroz CSS stilizaciju igra veoma bitnu ulogu .

Napomenućemo samo par efekata koji su koriščeni za stilizaciju .

Slika koju korisnik vidi prilikom dolaska na početnu stranu je kombinacija efekta linear-gradient i background-image , a zaobljenost je dobijena efektom clip-path .

.hero {

    background-image: linear-gradient(to right bottom,rgba(111, 210, 213, 0.8),rgba(91, 238, 186, 0.8)), url(/static/photos/prva.png);

    background-size: cover;

    -webkit-clip-path: circle(70% at 70% 20%);

    clip-path: circle(70% at 70% 20%);

    height: 100vh;

    position: relative;

    bottom: 100px;

    text-align: center;

    letter-spacing: 34px;

    padding-top: 30vh;

    font-size: 15vh;

}

S obzirom da je clip-path relativno nov efekat, različiti browseri zahtevaju različitu implementaciju u CSS-u .

Svi tabovi sa aranžmanima su zasenčeni upotrebom box-shadow atributom , i zaobljeni upotebom border-radiusa. Prelaskom miša preko svakog taba taj tab se uvećava dvadeset posto upotrebom atributa transform:scale , koji da bi imao glatkoću tranzicije koristi atribut transition : all .2s

.tourTab{

    border: 1px solid #d6cfcf;

    border-radius: 15px;

    box-shadow: 0 2px 10px rgba(0,0,0,0.3);

    padding: 20px;

    margin: 50px;

    width: 400px;

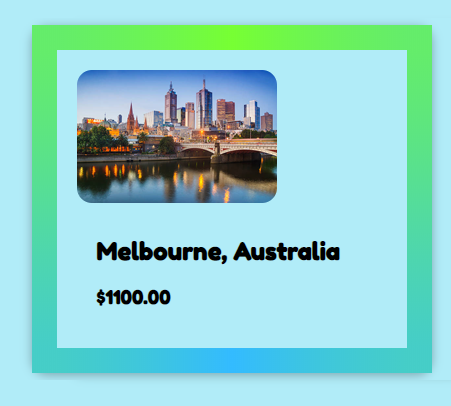
    transition: all .2s;

}

.tourTab:hover{

    transform: scale(1.2);

}

Aplikacija koristi i neke od najnovijih CSS atributa poput conic-gradient koji daje sledeći efekat 

Kao što vidimo okvir ovog taba izgleda veoma atraktivno. Nažalost , Mozilla Firefox još uvek ne podržava ovaj efekat, ali je taj problem rešen kodom

.mostPopularBorder{

    border: 25px solid #0ED2F7;

    -moz-border-image: auto;

    border-image-slice: 1;

    border-image-source: conic-gradient(

        hsl(100 100% 60%),

        hsl(200 100% 60%),

        hsl(100 100% 60%)

    );

}

tako da i u Firefoxu, i bez ovog efekta izgleda pristojno.

1. API

Web aplikacija koja ima ambiciju da bude zaista ozbiljna, mora imati i sopstveni web servis, odnosno API.

Globe Tours, s obzirom da je napravljena upotrebom Django radnog okvira, logično koristi i Django REST radni okvir za svoj API.

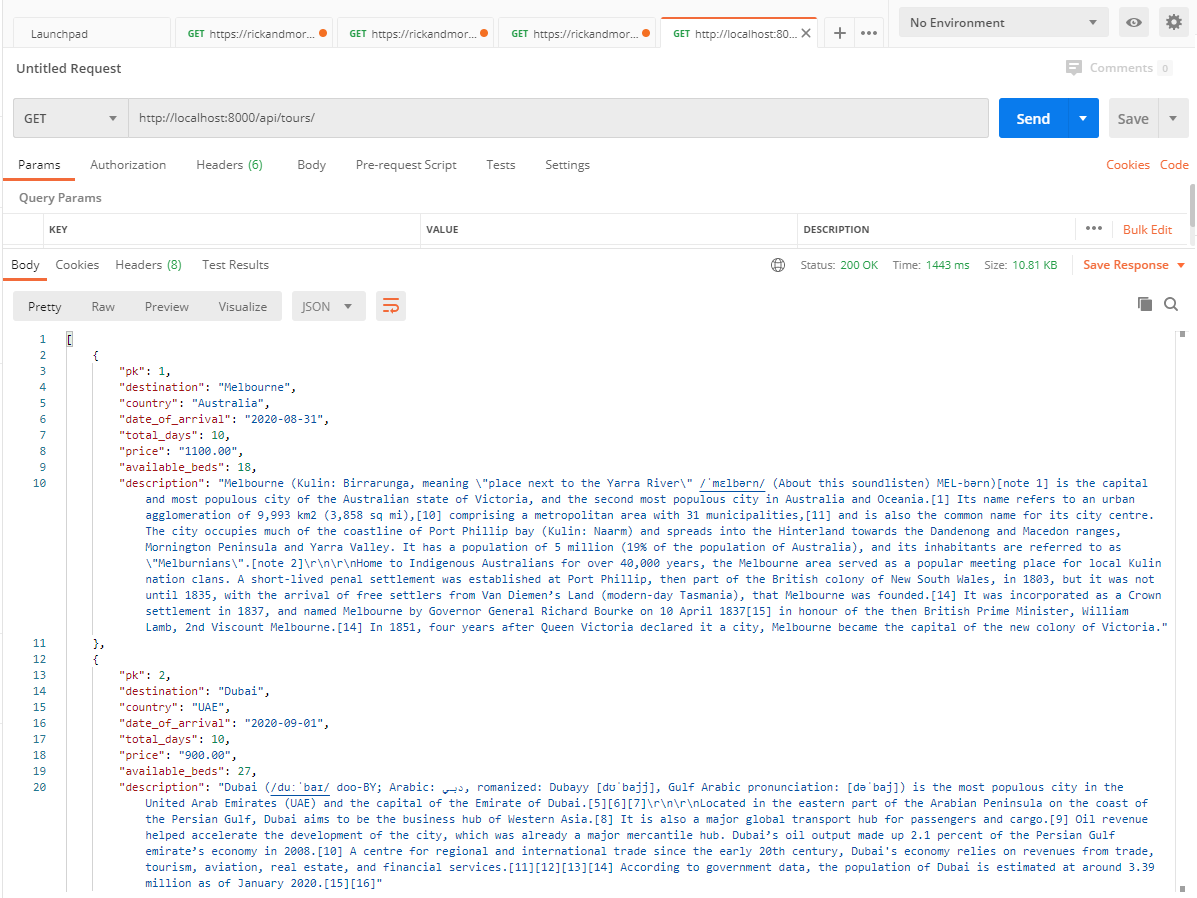
Poenta ovog API-ja jeste da druge aplikacije mogu na posredan način da pristupe bazi podataka aplikacije Globe Tours, i na taj način mogu da vide ponuđene aranžmane i eventualno da kroz sopstvenu aplikaciju izvrše bukiranje nekog od aranžmana.

Postoje dva endpoint-a koja nam ovo omogućuju. S obzirom da je za API korišćen REST protokol , svi pozivi ka njemu idu preko HTTP protokola i koriste njegove dobro poznate metode.

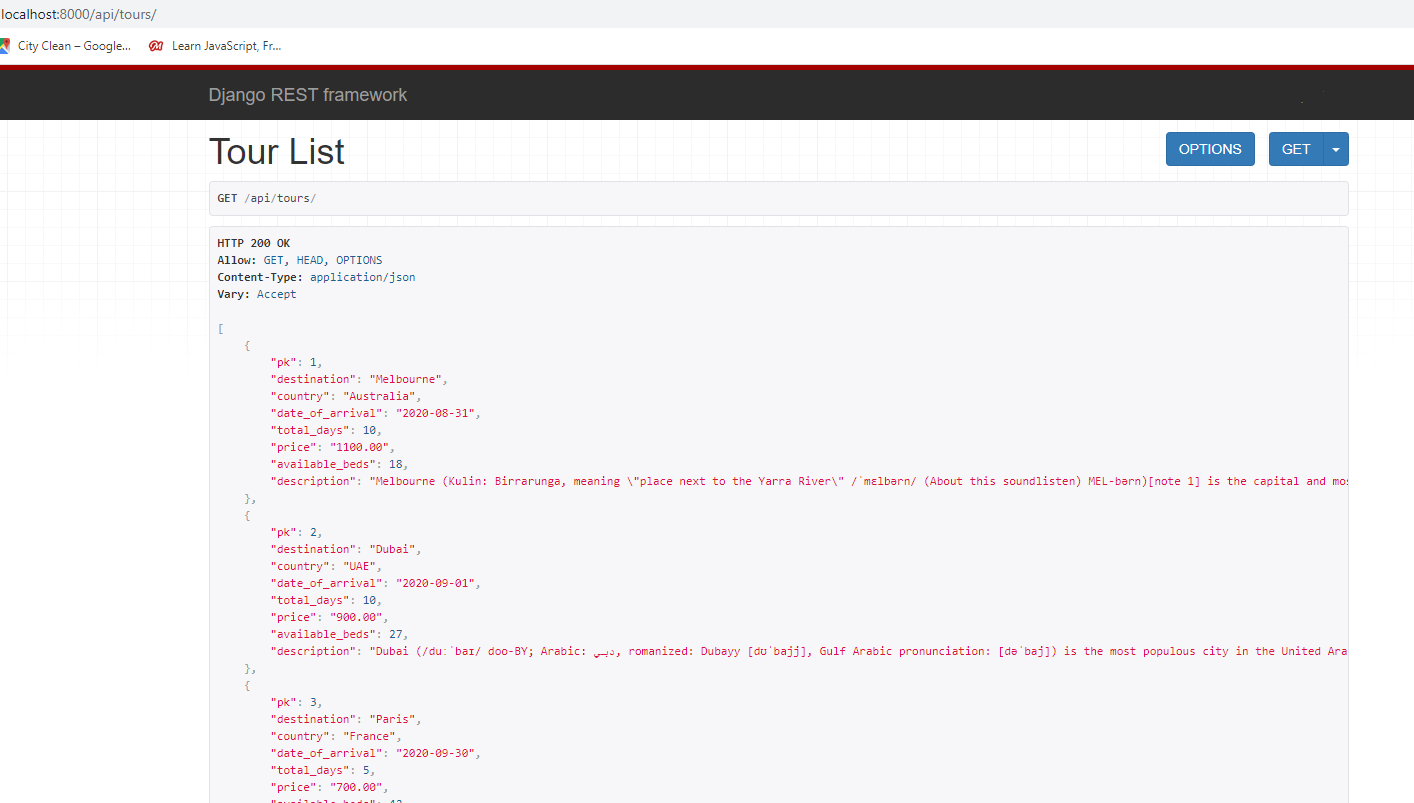
Prvi endpoint tiče se dobijanja informacija o svim raspoloživim aranžmanima. Logično, za eksploataciju ovog endpointa koristimo GET metodu, a path parametri su /api/tours . Ukoliko je host npr https://www.globetours.com onda bi kompletan url bio

<https://www.globetours.com/api/tours> .

Upotrebom aplikacije za testiranje API-ja koja se zove Postman , vidimo da poziv funkcioniše.



Inače, s obzirom da je GET metoda u pitanju, ovaj poziv smo mogli da testiramo i direktno u browseru , a Django ima i sopstveni interfejs koji dobijene podatke iz ovog poziva čini još preglednijim .



Drugi endpoint daje mogućnost bukiranja aranžmana preko neke druge aplikacije. Ona koristi POST metod a path parametri su /api/clients . U body zahteva idu podaci : name,surname,email,number\_of\_guests i id аranžmana. Nakon provere slobodnih mesta , ukoliko su svi parametri validni API vraća poruku sa uspešno unetim parametrima kojim potvrđuje rezervaciju , i klijent dobija rezervacioni kod putem mejla isto kao kada bi rezervisao direktno na sajtu Globe Tours . Ukoliko nisu pravilo uneti parametri API vraća poruku :

{'error':'invalid data'}