



# Postavljanje statičke web stranice za usluge fotografisanja koristeći Docker i Render

## OPERATIVNI SISTEMI I RAČUNARSTVO U OBLAKU

Aida Terzi , Hana Karzi , Amina Helja, Selma Škulj

### UVOD

Cilj ovog projekta je izrada i postavljanje jednostavne web stranice koja prezentuje usluge fotografisanja. Fokus je stavljen na brzinu učitavanja, preglednost i lakoću korištenja, kako bi korisničko iskustvo bilo što prijatnije.

Posebna pažnja posvećena je načinu postavljanja i održavanja stranice, kako bi ona bila pouzdana, lako dostupna i jednostavna za dalje unapređenje.

### Korištene tehnologije

U realizaciji ovog projekta korištene su sljedeće tehnologije:

- **HTML, CSS** – Osnovne tehnologije za izradu statičke web stranice. HTML se koristi za strukturu stranice, CSS za stilizaciju.
- **Docker** – Platforma za kontejnerizaciju aplikacija. Omogućava pakovanje web stranice i svih njenih zavisnosti u kontejner, čime se olakšava pokretanje u različitim okruženjima.
- **Render** – Platforma za deploy i hosting aplikacija. Render omogućava jednostavno hostovanje Docker kontejnera i automatski upravlja sigurnošću i dostupnošću stranice.
- **GitHub** – Za verzionisanje koda i saradnju, korišten je GitHub, koji omogućava efikasno praćenje promjena i kolaboraciju.

## Koraci izrade projekta:

1. Odabir cloud platforme
2. Kreiranje jednostavne web stranice
3. Dockerizacija aplikacije
4. Povezivanje sa GitHub repozitorijumom
5. Postavljanje aplikacije na cloud platformu
6. Testiranje dostupnosti aplikacije

### 1. Odabir cloud platforme

Za deployment aplikacije odabrali smo **Render** kao cloud platformu zbog njene jednostavnosti i podrške za automatsko postavljanje aplikacija direktno iz GitHub repozitorijuma. Render omogućava lako upravljanje Docker containerima, nudi besplatan hosting za web servise, i ne zahtijeva dodatnu konfiguraciju servera, što je idealno za brzu i efikasnu realizaciju studentskog projekta. Takođe, platforma omogućava jednostavno povezivanje sa domenom i pruža korisnički interfejs pogodan za početnike.

Za potrebe projekta odabrali smo opciju **Free Web Service** na cloud platformi **Render**, koja omogućava besplatno postavljanje statičkih i dinamičkih web aplikacija direktno iz GitHub repozitorijuma.

Ova opcija (Free Web Service) omogućava:

- **Automatski deployment** prilikom svakog "push-a" na GitHub
- **Podršku za Docker kontejnere**
- **Besplatni hosting** sa ograničenim resursima (512 MB RAM, 0.1 CPU)

### 2. Kreiranje jednostavne web stranice

Za realizaciju projekta razvili smo jednostavnu statičku web stranicu koja se sastoji od dve osnovne stranice: **index.html** i **kontakt.html**. Ove stranice smo organizovali u projektnom folderu pod nazivom **osiruo-projekat**, gde su smešteni svi relevantni fajlovi i resursi.

**Index.html** Glavna početna stranica sadrži jednostavnu, ali jasnu poruku dobrodošlice. Pored toga, stranica pruža osnovne informacije o uslugama fotografisanja, ističući fokus na kvalitet i profesionalnost usluge. Izostavili smo složene dizajnerske elemente i slike, jer je cilj bio da se naglasi funkcionalnost i brzina učitavanja.

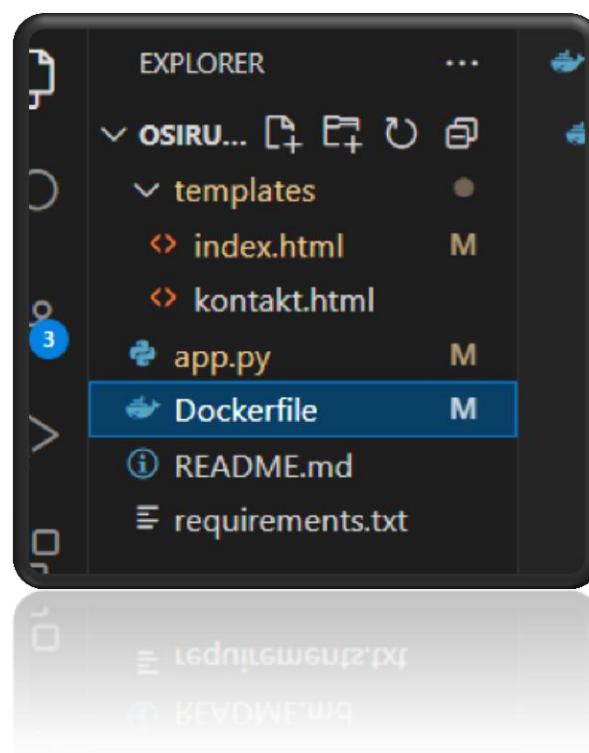
**Kontakt.html** Ova stranica služi za prikaz kontakt podataka i jednostavnog kontakt formulara kroz koji potencijalni klijenti mogu poslati upit ili poruku. Formular je statički i služi samo kao vizuelni element u ovom projektu, bez funkcionalnosti slanja.

Web stranica je izrađena koristeći Visual Studio Code, sa fokusom na jednostavnost i brzo učitavanje. Satičkog je tipa, što znači da se njen sadržaj ne menja dinamički na serveru, što dodatno doprinosi brzini.

### 3. Dockerizacija web stranice

**Dockerizacija** je proces kojim se neka aplikacija (npr. naša Flask web aplikacija) pakuje u tzv. **Docker kontejner**. Taj kontejner sadrži sve što je potrebno za rad aplikacije:

- operativni sistem,
- programski jezik (npr. Python),
- .py fajlove,
- HTML stranice,
- biblioteke i zavisnosti (iz requirements.txt).



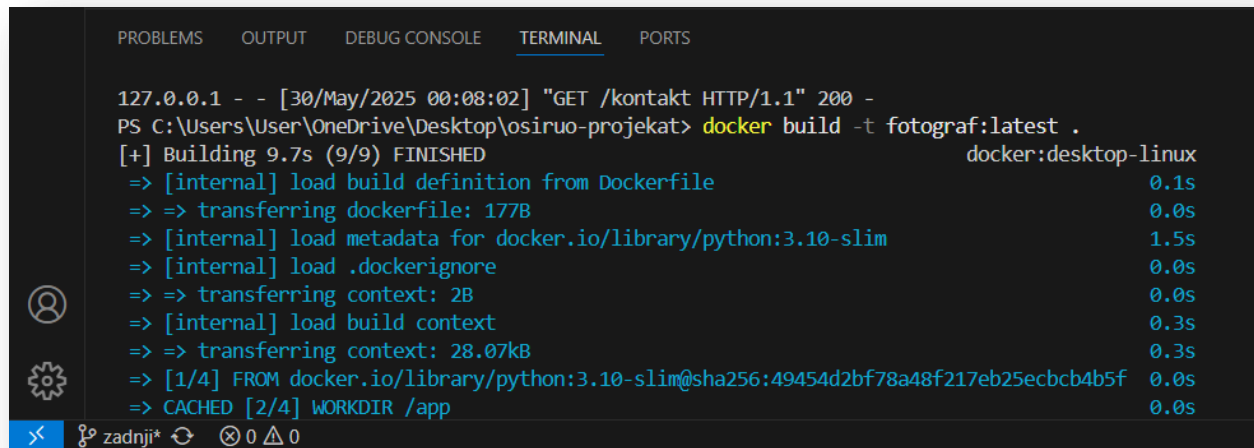
## Kreiranje Docker kontejnera (build i run)

U procesu kreiranja Docker kontejnera, prvi korak je izgradnja Docker slike pomoću komande:

```
docker build -t fotograf:latest .
```

Ova komanda govori Dockeru da napravi novu sliku na osnovu uputstava koja se nalaze u Dockerfile fajlu unutar trenutnog direktorija (što označava tačka na kraju komande).

Parametar **-t fotograf:latest** dodjeljuje toj slici naziv **fotograf** i oznaku verzije **latest**, što olakšava kasnije pozivanje slike. Tokom build procesa, Docker kopira sve potrebne fajlove iz projekta, instalira zavisnosti i priprema okruženje u kojem će aplikacija raditi. Kao rezultat, dobija se funkcionalna Docker slika koja sadrži tvoju web aplikaciju spremnu za pokretanje u kontejneru.

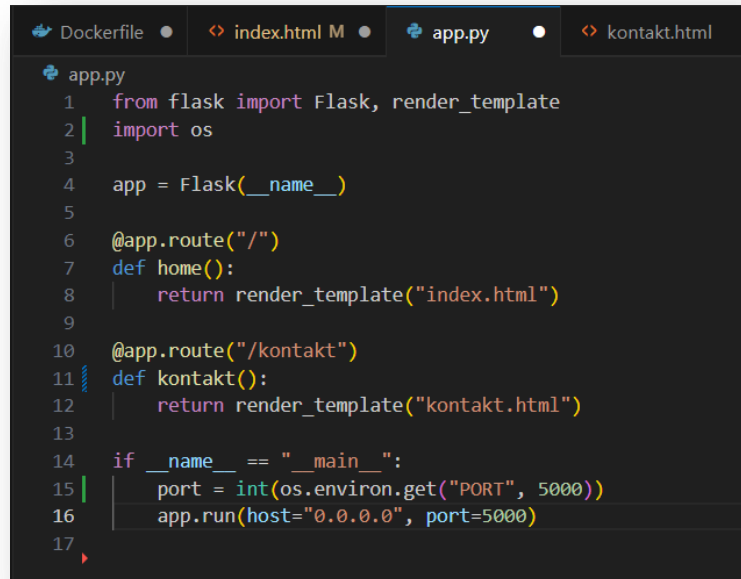


```
127.0.0.1 - - [30/May/2025 00:08:02] "GET /kontakt HTTP/1.1" 200 -
PS C:\Users\User\OneDrive\Desktop\osiruo-projekat> docker build -t fotograf:latest .
[+] Building 9.7s (9/9) FINISHED                                docker:desktop-linux
=> [internal] load build definition from Dockerfile             0.1s
=> => transferring dockerfile: 177B                             0.0s
=> [internal] load metadata for docker.io/library/python:3.10-slim 1.5s
=> [internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build context                                0.3s
=> => transferring context: 28.07kB                               0.3s
=> [1/4] FROM docker.io/library/python:3.10-slim@sha256:49454d2bf78a48f217eb25ecbcb4b5f 0.0s
=> CACHED [2/4] WORKDIR /app                                    0.0s
```

Nakon što je Docker slika uspješno kreirana, sljedeći korak je pokretanje kontejnera pomoću komande **docker run -d -p 5000:5000 --name kontejner-haas fotograf**. Ova komanda omogućava da se aplikacija pokrene unutar kontejnera na osnovu prethodno napravljene slike nazvane **fotograf**.

Opcija **-d** znači da se kontejner pokreće u pozadini (detached mode), dok **-p 5000:5000** mapira port 5000 na računaru (hostu) na isti port unutar kontejnera, što omogućava pristup aplikaciji preko localhost:5000. Dodatno, **--name kontejner-haas** dodjeljuje kontejneru prepoznatljivo ime **kontejner-haas**, što olakšava njegovo upravljanje.

## Pokretanje Flask aplikacije na portu 5000



```
app.py
1  from flask import Flask, render_template
2  import os
3
4  app = Flask(__name__)
5
6  @app.route("/")
7  def home():
8      return render_template("index.html")
9
10 @app.route("/kontakt")
11 def kontakt():
12     return render_template("kontakt.html")
13
14 if __name__ == "__main__":
15     port = int(os.environ.get("PORT", 5000))
16     app.run(host="0.0.0.0", port=5000)
17
```

U fajlu app.py vidi se da se Flask aplikacija pokreće na portu 5000 pomoću sljedećeg dijela koda:

---

```
if __name__ == "__main__":
```

```
    port = int(os.environ.get("PORT", 5000))
```

```
    app.run(host="0.0.0.0", port=5000)
```

---

**os.environ.get("PORT", 5000)** – pokušava uzeti broj porta iz okruženja (npr. iz Docker ili Render servera). Ako nije definisan, koristi **5000** kao podrazumijevani port.

**app.run(host="0.0.0.0", port=5000)** – pokreće Flask server tako da ga mogu pristupiti i vanjski uređaji (ne samo localhost).

## 4. Povezivanje sa Git Hub repozitorijumom

```
PS C:\Users\User\OneDrive\Desktop\osiruo-projekat> docker run -d -p 5000:5000 --name kontejner-haas fotograf
21f69ecaadf3e3928c4fc9190da636cf4c2360b4f001bbe5777ddabc957f2340
PS C:\Users\User\OneDrive\Desktop\osiruo-projekat> git init
Reinitialized existing Git repository in C:/Users/User/OneDrive/Desktop/osiruo-projekat/.git/
PS C:\Users\User\OneDrive\Desktop\osiruo-projekat> git checkout -b zadnjii
Switched to a new branch 'zadnjii'
PS C:\Users\User\OneDrive\Desktop\osiruo-projekat> git add .
PS C:\Users\User\OneDrive\Desktop\osiruo-projekat> git commit -m "valjda zadnje"
[zadnjii 3782f6a] valjda zadnje
4 files changed, 1 insertion(+), 1 deletion(-)
delete mode 100644 templates/dogadjaj.jpg
delete mode 100644 templates/portret.jpg
```

Da bi se naš projekat verzionisao i kasnije postavio na platforme kao što je Render, prvo je bilo potrebno povezati ga sa GitHub repozitorijumom. U nastavku je objašnjen postupak i značenje korištenih Git komandi:

### git init

Ova komanda inicijalizuje novi Git repozitorij u trenutnom direktoriju. To znači da tvoj lokalni projekat sada postaje "praćen" od strane Gita – sve izmjene u fajlovima mogu biti praćene i verzionisane.

### git checkout -b zadnji

Ova naredba pravi novu granu pod imenom zadnji i automatski prelazi na nju. Grane u Gitu omogućavaju paralelan razvoj i testiranje bez uticaja na glavnu (main/master) granu.

### git add .

Ovom naredbom dodajmo sve izmjene i fajlove u tzv. **staging area** – to znači da smo spremni da ih snimiš (commit-uješ).

### git commit -m "zadnje"

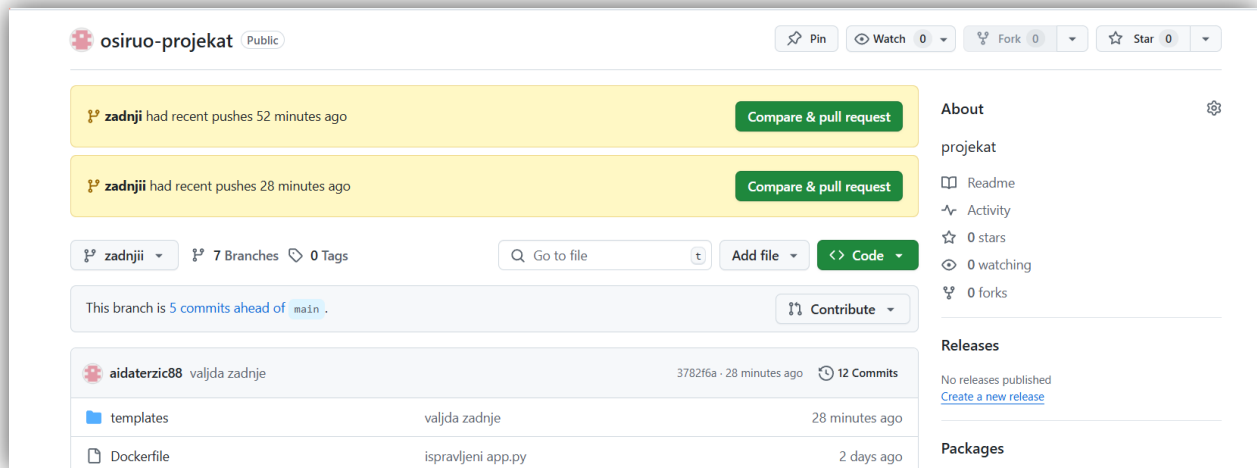
Commit znači da pravimo "snimak" trenutnog stanja projekta. Poruka "zadnje" objašnjava šta je urađeno u tom commitu (npr. završne izmjene prije objave).

## Dodavanje GitHub repozitorija:

Prije slanja na GitHub, moramo povezati lokalni repozitorij s udaljenim:

Komanda:

```
git remote add origin https://github.com/korisnickoime/naziv-repozitorija.git
```



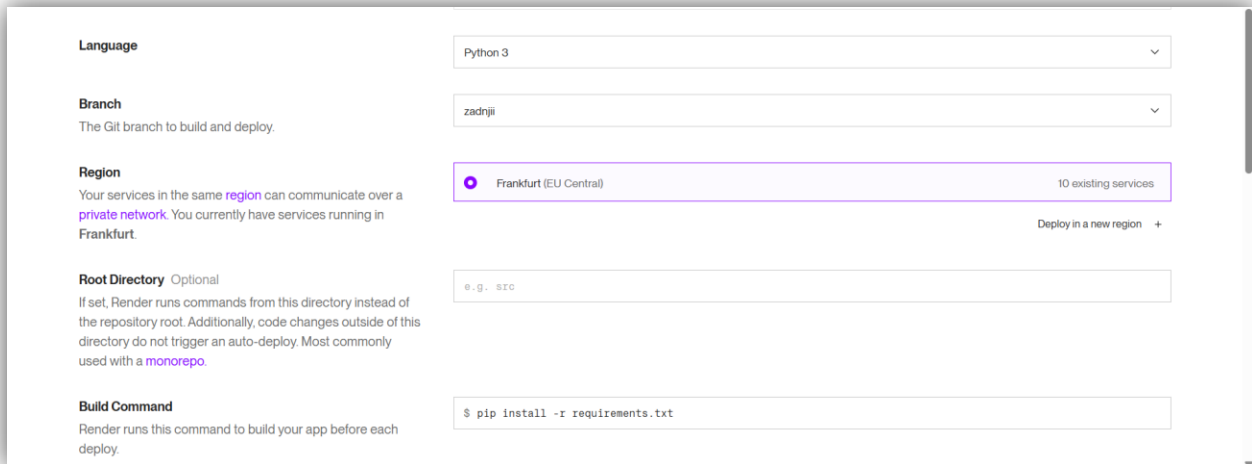
## 5. Postavljanje stranice na Web platformu

Nakon što smo projekat povezale sa GitHub repozitorijumom, sljedeći korak bio je da aplikaciju postavimo online koristeći **Render** – cloud platformu koja omogućava jednostavno hostovanje web aplikacija direktno iz GitHub-a.

**Koraci koje smo slijedile:**

1. **Prijavile smo se** [Render](#) koristeći GitHub nalog kako bi se omogućio pristup našim repozitorijima.
2. Kliknule smo na **"New"** → **"Web Service"** da bismo kreirale novu web uslugu.
3. Render je zatim prikazao listu repozitorija, a mi smo odabrale onaj gdje se nalazi naš projekat – **osiruoprojekat**.

## Popunjavanje konfiguracije servisa



The screenshot shows the 'New Web Service' configuration form in the Render dashboard. It includes fields for Language (Python 3), Branch (zadnji), Region (Frankfurt (EU Central) with 10 existing services), Root Directory (e.g. src), and Build Command (\$ pip install -r requirements.txt). A 'Deploy in a new region' link is also visible.

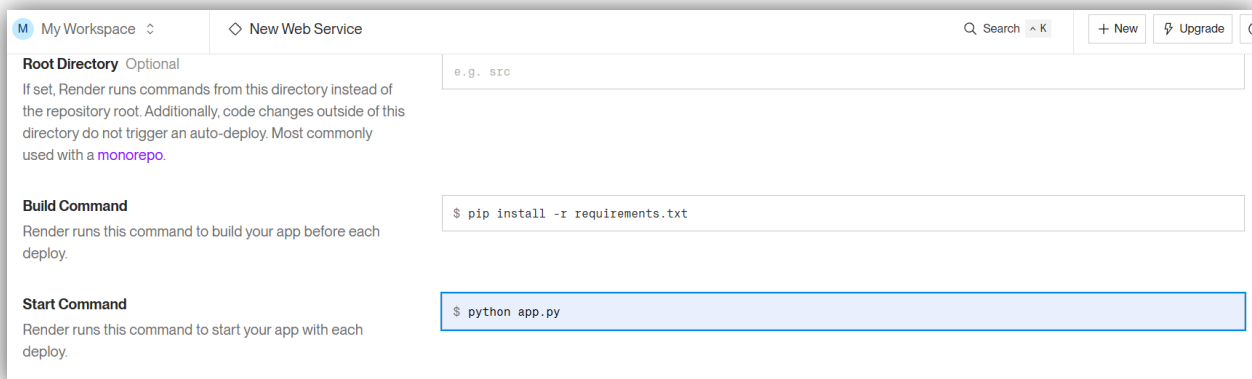
**Language** Python 3

**Branch** zadnji

**Region** Frankfurt (EU Central) 10 existing services  
Deploy in a new region +

**Root Directory** Optional  
e.g. src

**Build Command**  
\$ pip install -r requirements.txt



This screenshot shows the 'New Web Service' configuration form with the 'Start Command' field highlighted. The 'Root Directory' and 'Build Command' fields are also visible.

**Root Directory** Optional  
e.g. src

**Build Command**  
\$ pip install -r requirements.txt

**Start Command**  
\$ python app.py

### 1. Name (Ime aplikacije)

- Unosimo naziv svoje web aplikacije koji će biti vidljiv na Render kontrolnoj tabli i u URL-u.
- **Šta smo unijele:** osiruo-projekat-7
- Ime se koristi da bi lako prepoznali aplikaciju među ostalim servisima. Ovaj naziv će biti dio naše web adrese, npr. <https://osiruo-projekat-7.onrender.com>.

### 2. Region

- Lokacija servera gdje će naša aplikacija biti hostovana.
- **Šta smo izabrale:** Frankfurt (EU Central)
- To je najbliža regija našoj lokaciji (Bosna i Hercegovina), pa aplikacija brže učitava zbog manje mrežne udaljenosti.



### 3. Branch

- Traži Git granu iz koje će Render preuzeti kod.
- **Šta smo unijele:** `zadnji`
- To je bila finalna grana u kojoj se nalazi naš gotov projekat – spreman za deploy.

### 4. Root Directory

- Opisuje gdje se u projektu nalazi tvoja aplikacija (glavni folder).
- **Šta si unijela:** `.` ili `src`
- Ako su `app.py` i `requirements.txt` u glavnom folderu, unosi se tačka `..`. Ako su u podfolderu (npr. `src`), moraš to tačno navesti kako bi Render znao gdje da traži aplikaciju.

### 5. Runtime Environment/Language

- Okruženje/programski jezik koji tvoja aplikacija koristi.
- **Šta smo izabrale:** `Python 3`
- Aplikacija je rađena u Flask-u, koji je Python web framework – pa treba da se pokrene u Python okruženju.

### 6. Build Command

- Komanda koju Render treba da izvrši kako bi pripremio aplikaciju.
- **Šta smo unijele:**  
`pip install -r requirements.txt`
- Ova komanda instalira sve Python biblioteke potrebne za rad aplikacije. Sve što je navedeno u `requirements.txt` biće automatski instalirano.

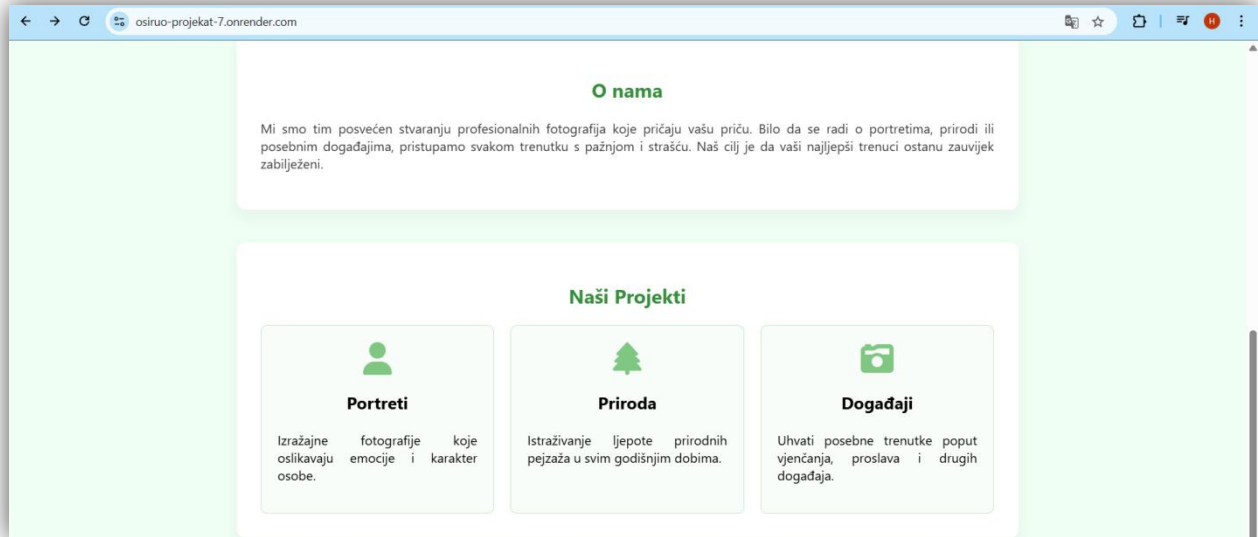
### 7. Start Command

- Komanda kojom se aplikacija pokreće.
- **Šta smo unijele:**  
`python app.py`
- To je standardan način za pokretanje Flask aplikacije. Render koristi ovu komandu da bi znao kako da aktivira tvoj server.

Zatim smo kliknule na "**Create Web Service**" / "**Deploy**", a Render je:

- povukao kod iz GitHub repozitorijuma,
- pokrenuo `pip install` da instalira zavisnosti,
- zatim `python app.py` da starta aplikaciju,
- i konačno objavio našu web stranicu na URL adresi poput:  
`https://osiruo.projkat-7.onrender.com`

## 6. Testiranje distuonosti aplikacije



Web stranica je uspješno postavljena na cloud.

Link ka aplikaciji: <https://osiruo-projekat-7.onrender.com/>

Link ka GitHub repozitoriju: <https://github.com/aidaterzic88/osiruo-projekat.git>

### Šta smo naučile kroz projekat

Kroz ovaj projekat naučile smo kako se statička web stranica može dokerizirati i postaviti na cloud koristeći platformu Render. Prvo smo pomoću **Docker-a** kreirale sliku aplikacije i pokrenule je u kontejneru na portu 5000. Zatim smo kroz **Flask** kod omogućile da aplikacija bude dostupna vanjskim uređajima.

Dalje, upoznale smo se s osnovama verzionisanja pomoću **Git-a**, gdje smo koristile naredbe poput `git init`, `git add`, `git commit` i `git push` kako bismo povezale projekat s **GitHub** repozitorijumom.

Na kraju, aplikaciju smo uspješno deployale na **Render**, gdje smo konfigurirale sve potrebne opcije kao što su jezik (Python 3), grana koda (`zadnji`), `build` i `start` komande. Time smo omogućile da naša web stranica bude dostupna online, bez potrebe za vlastitim serverom.

Ovaj proces nam je omogućio da praktično primijenimo znanja iz oblasti web razvoja, Docker kontejnerizacije, Git verzionisanja i cloud deploy-a.

