

IT255 – Veb sistemi

**Projektna dokumentacija**

**Animal Adoption**

Ime i prezime: **Antea Primorac**

Broj indeksa: **5157**

Datum izrade:

# Sadržaj

[Sadržaj 2](#_Toc18)

[Uvod 2](#_Toc19)

[Pozadina i namjena 2](#_Toc20)

[Slučajevi korištenja 3](#_Toc21)

[Akteri 3](#_Toc22)

[Dijagram 3](#_Toc23)

[Opis slučajeva korištenja 3](#_Toc24)

[Registracija 3](#_Toc25)

[Login 4](#_Toc26)

[Pregled kataloga 4](#_Toc27)

[Pregled detaljnijih informacija o životinji 5](#_Toc28)

[Posvajanje životinje 5](#_Toc29)

[CRUD operacije nad korisnicima 6](#_Toc30)

[CRUD operacije nad životinjama 6](#_Toc31)

[CRUD operacije nad posvajanjima 7](#_Toc32)

[Zahtjevi 8](#_Toc33)

[Funkcionalni zahtjevi 8](#_Toc34)

[Funkcionalni zahtjevi za administratore aplikacije 8](#_Toc35)

[Nefunkcionalni zahtjevi 8](#_Toc36)

[Faza projektiranja 9](#_Toc37)

[Arhitektura 9](#_Toc38)

[Konceptualni dijagram 10](#_Toc39)

[JSON 10](#_Toc40)

[Implementacija 12](#_Toc41)

[Struktura projekta 12](#_Toc42)

[AppModule i AppComponent 12](#_Toc43)

[Router 12](#_Toc44)

[Guardovi 12](#_Toc45)

[AuthGuard 12](#_Toc46)

[AdminGuard 12](#_Toc47)

[Servisi 13](#_Toc48)

[Modeli 13](#_Toc49)

[Primjer koda za izabranu komponentu 14](#_Toc50)

[Screenshotovi 17](#_Toc51)

[Početna stranica 17](#_Toc52)

[About Us 18](#_Toc53)

[Contact Us 18](#_Toc54)

[Register 19](#_Toc55)

[Login 19](#_Toc56)

[Detalji o životinji 20](#_Toc57)

[Posvajanje (unutar stranice detalja) 21](#_Toc58)

[Potvrda o posvajanju 24](#_Toc59)

[Admin CRUD 24](#_Toc60)

[Zaključak 24](#_Toc61)

[Reference 25](#_Toc62)

# Uvod

Svrha ovog projekta je izrada web aplikacije – Animal Adoption - za virtualno (simbolično) posvajanje ugroženih životinjskih vrsta.

### Pozadina i namjena

Animal Adoption predstavlja site koji korisnicima omogućava simbolično posvajanje životinja. Namjena je potaknuti interes za očuvanje ugroženih vrsta, te omogućiti korisniku da pomogne takvim nastojanjima.

Sadrži katalog na kojem korisnici mogu vidjeti pojedinačne jedinke. Klikom na više informacija, korisnik može saznati više o pojedinoj životinji, te se po želji odlučiti za posvajanje, ispunjavanjem za to predviđene forme. Potonja predviđa unos ličnih podataka, te uplatu (plaćanje je za potrebe projekta simulirano), nakon čega korisnik može vidjeti potvrdu uspješnog okončanja procesa.

Projekt je rađen isključivo na frontendu, primjenom Angular 15, Bootstrap-a te HTML5. Od biblioteka su korišteni RxJS, ngrx, ngx-datatable te ngx-formly. Na mjesto backend baze podataka postavljen je json-server, sa kojim aplikacija ostvaruje interakcije preko servisa.

# Slučajevi korištenja

## Akteri

* **Neregistrirani korisnici** – mogu pregledati katalog životinja, informacije o siteu (About us), te kontakt podatke (Contact us).
* **Registrirani korisnici** – mogu uraditi sve što i neregistrirani korisnici, plus pregledati detalje o pojedinačnoj životinji, te popuniti formu za njeno usvajanje
* **Administratori** – mogu raditi sve što i registrirani korisnici, plus pristupiti Admin Portalu odakle mogu vršiti CRUD operacije nad entitetima (korisnici, životinje, posvajanja).

## Dijagram

TODO

## Opis slučajeva korištenja

#### Registracija

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

#### Login

|  |  |
| --- | --- |
| Kratak opis: | Korisnik se logira u aplikaciju. |
| Aktor: | Registrirani korisnik, ili Administrator |
| Preduslovi: | * Korisnik ima račun |
| Osnovni tok: | 1. Korisnik odabire Login link. 2. Korisnik unosi korisničko ime i/ili lozinku. 3. Potvrđuje unos. |
| Alternativni tok: | 1. Korisnik bira link ka formi za registraciju. 2. Prelazi se na slučaj korištenja Registracija. |
| Izuzeci: | * Pogrešno korisničko ime i/ili lozinka * Račun ne postoji |
| Rezultat: | Korisnik je uspješno autentificiran i autoriziran. Može izvršavati akcije koje su u skladu sa njegovim ovlastima (grupa Registrirani korisnik, ili grupa Administrator). |

#### Pregled kataloga

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

#### Pregled detaljnijih informacija o životinji

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

#### Posvajanje životinje

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

#### CRUD operacije nad korisnicima

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

#### CRUD operacije nad životinjama

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

#### CRUD operacije nad posvajanjima

|  |  |
| --- | --- |
| Kratak opis: |  |
| Aktor: |  |
| Preduslovi: |  |
| Osnovni tok: |  |
| Alternativni tok: |  |
| Izuzeci: |  |
| Rezultat: |  |

# Zahtjevi

### Funkcionalni zahtjevi

U ovoj sekciji navest će se funkcionalni zahtjevi sistema – specifične funkcije i servisi koje sistem treba pružati svojim korisnicima. Njima se specificira šta bi sistem trebao biti sposoban raditi – a to, između ostaloga, uključuje ulaze koje obrađuje, te sa njima povezane izlaze.

#### Funkcionalni zahtjevi za administratore aplikacije

**Administratorski pristup:**

* Administratori se mogu prijaviti pod svojim korisničkim imenom i lozinkom
* Administratori mogu pristupiti administracijskom području (Admin Portal).
  + Osobe koje nisu administratori ne mogu pristupiti ovom području
* Admin Portal sadrži linkove ka pojedinačnim CRUD stranicama entiteta (User Management, Adoption Management, Animal Management). Svaka stranica sadrži tablicu, te formu.
  + Tablica se sastoji od redova. Svaki red odgovara postojećoj stavki, te pored nje sadržava dugme za brisanje (Delete) i dugme za ažuriranje (Edit) stavke.
  + Forma omogućava dodavanje nove stavke, kao i ažuriranje postojeće stavke (ako je administrator kliknuo Edit za neki red tablice).

**Upravljanje korisnicima:**

* Administratori stranice mogu vidjeti sve korisnike stranice
* Administratori stranice mogu uređivati korisnike
* Administratori stranice mogu izbrisati korisnike
* Administratori stranice mogu kreirati nove korisnike, uključujući i nove administratore

**Upravljanje životinjama:**

* Administratori stranice mogu vidjeti sve životinje
* Administratori stranice mogu uređivati postojeće životinje
* Administratori stranice mogu izbrisati životinje
* Administratori stranice mogu dodati nove životinje

**Upravljanje posvajanjima:**

* Administratori stranice mogu vidjeti sva posvajanja
* Administratori stranice mogu uređivati postojeća posvajanja
* Administratori stranice mogu brisati posvajanja
* Administratori stranice mogu dodavati nova posvajanja

### Nefunkcionalni zahtjevi

**Performanse**

* Svaka pojedinačna stranica se treba učitati unutar 1.5 sekunde, najmanje 95% vremena.
* Odziv na korisnikove akcije nad formama treba biti unutar 0.1 sekunde.
* Aplikacija treba zadržati stabilno vrijeme odziva uz opterećenje do 1000 istovremenih korisnika.

**Dostupnost**

* Aplikacija treba biti dostupna korisnicima najmanje 99.5% vremena. Iznimka su zakazana održavanja sitea ili baze podataka, koja će biti obavljana svake prve nedjelje u mjesecu u vrijeme kad ima najmanje korisničkog prometa.

**Sigurnost**

* Korisnici ne bi trebali moći pristupiti računu-nalogu bez odgovarajućeg korisničkog imena i lozinke.
* Neautorizirani korisnici, i korisnici koji nisu administratori, ne bi trebali moći pristupiti administratorskom području, niti koristiti njegove funkcije (CRUD operacije nad bazom podataka).
* (Nije obuhvaćeno unutar ovog projekta) Aplikacija treba koristiti HTTPS za enkripciju podataka u tranzitu.
* (Nije obuhvaćeno unutar ovog projekta) Korisnici ne bi trebali moći vidjeti informacije o posvajanjima s drugih računa.
* (Nije obuhvaćeno unutar ovog projekta) Radi bolje sigurnosti plaćanja, aplikacija treba koristiti procesor plaćanja, primjerice PayPal ili Stripe.

**Kompatibilnost**

* Website treba biti kompatibilan i dostupan za korištenje na svim modernim web preglednicima koji podržavaju ECMAScript 6 i HTML 5.
* Website treba koristiti responzivan dizajn za mobilne uređaje.

**Upotrebljivost**

* Website treba omogućiti jasnu i jednostavnu navigaciju.
* Website treba koristiti lako čitljiv font prikladne vještine.
* Website treba biti dostupan osobama sa oštećenjima vida, sluha, motorike...

# Faza projektiranja

### Arhitektura

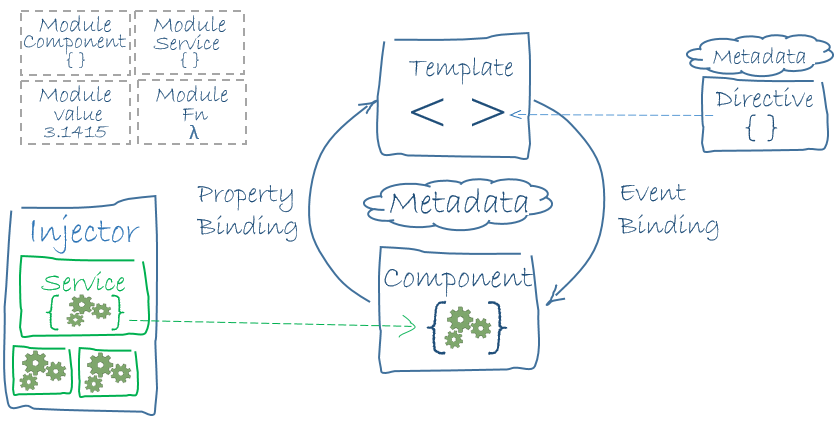
Inicijalna verzija Angulara, nazvana AngularJS, koristila je MVC arhitekturu. Međutim, od verzije 2 nadalje, Angular oslanja se na arhitekturu zasnovanu na komponentama (component-based architecture), gdje komponente obnašaju uloge koje u tradicionalnom MVC-u vrše kontroleri i viewovi.

Arhitektura Angular aplikacije, kao što je opisano u službenoj-zvaničnoj dokumentaciji, izgrađena je na skupu međusobno povezanih koncepata:

* **Moduli**: Angular organizira kod u module, pri čemu korijenski modul, AppModule, služi kao ulazna točka (entry point). Moduli grupiraju srodni kod u funkcionalne skupove.
* **Komponente i pogledi**: Komponente su osnovne građevne jedinice. One upravljaju pogledima (viewovima), koji u biti predstavljaju vizualni dio aplikacije. Pogledi se kreiraju korištenjem templateova. Potonji jesu HTML datoteke, koje pored "čistog" HTML-a, koriste i elemente te atribute svojstvene Angularu.
* **Servisi i umetanje zavisnosti**: servisi pružaju specifičnu funkcionalnost koja nije direktno vezana uz prikaz elemenata. Umetanje zavisnosti (dependency injection) omogućava komponentama da koriste servise na način koji poboljšava modularnost i ponovnu upotrebljivost koda.
* **Rutiranje**: Angular Router omogućava navigaciju od jednog pogleda do drugog dok korisnici obavljaju zadatke aplikacije.

Ova arhitektura priža modularnost, omogućava jasno razdvajanje odgovornosti, i olakšava testiranje. Sve to čini Angular robusnom platformom za izgradnju skalabilnih web aplikacija.

Donja slika, koja također potiče iz službene dokumentacije, prikazuje pojednostavljeni dijagram arhitekture.



Slika 1 – Arhitekturalni dijagram Angular aplikacija (izvor: angular.io)

### Konceptualni dijagram

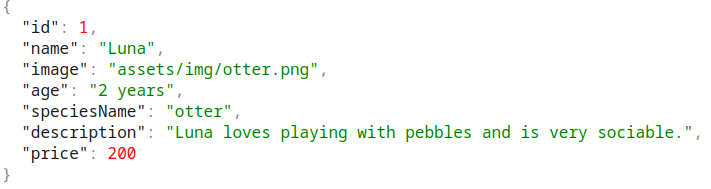
S obzirom da nije implementiran backend, nije ni projektirana prava, fizička baza podataka.

Jedan korisnik može posvojiti više životinja. Jednu životinju može posvojiti više korisnika. Za svako posvojenje čuva se ID korisnika i ID posvajanja.

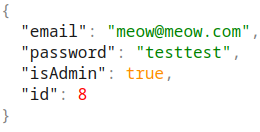
### JSON

Ispod se daje JSON struktura identificiranih entiteta.

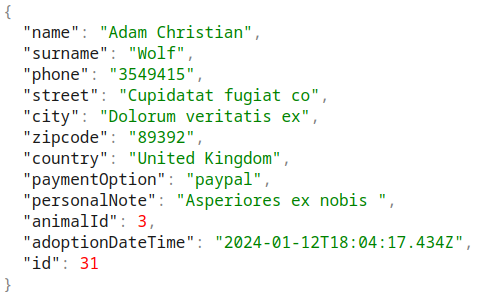
**Animal:**



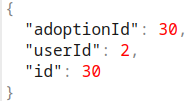
**User:**



**Adoption:**



**UserAdoption:**



Kod UserAdoptiona, vrijedi istaknuti da **id** i **adoptionId** imaju isto značenje – json-server je taj koji je automatski dodao atribut **id**.

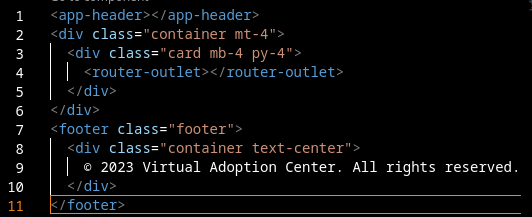
# Implementacija

## Struktura projekta

Unutar app/, aplikacija se sastoji od sljedećih osnovnih foldera:

* components - sadrži pojedinačne komponente, uključujući one koje se koriste na više različitih stranica, npr. header
* pages – stranice websitea; mogu se sastojati od jedne ili više komponenti
* services – poslovna logika, poput CRUD operacija nad entitetima, provjere postojanja korisnika, podešavanja trenutne korisničke sesije...
* models – klase koje predstavljaju definiciju struktura podataka, npr. Animal

## AppModule i AppComponent



TODO: AppModule

## Router

TODO Određene stranice aktiviraju guardove.

## Guardovi

### AuthGuard

TODO AuthGuard ograđuje stranicu sa detaljima o životinji (stranica sadrži formu za posvajanje), tako da je dostupna samo registriranim korisnicima

### AdminGuard

TODO AdminGuard osigurava da CRUD stranicama mogu pristupiti samo administratori sistema.

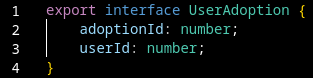
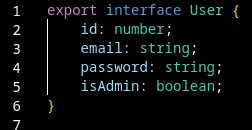
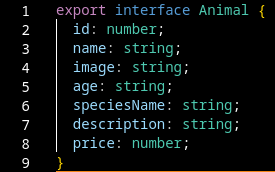
## Servisi

TODO: Projekt sadrži AnimalService, UserService, AdoptionService, CountryService. Vrijedi istaknuti da je CountryService ujedno najjednostavnija servisna klasa, čija je jedina namjena, putem HttpClient-a, dohvatiti listu zemalja sa <https://restcountries.com/v3.1/all.>

Prve tri klase također koriste HttpClient, ali za komunikaciju sa json-serverom.

## Modeli

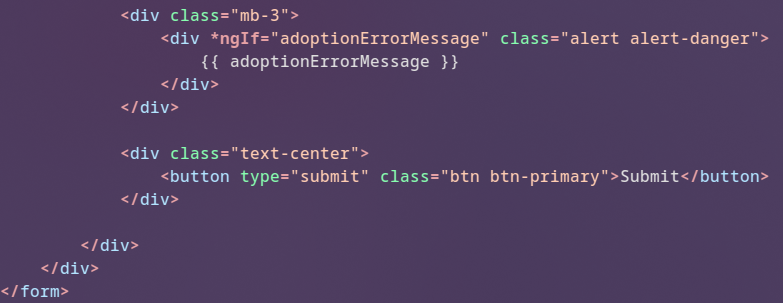
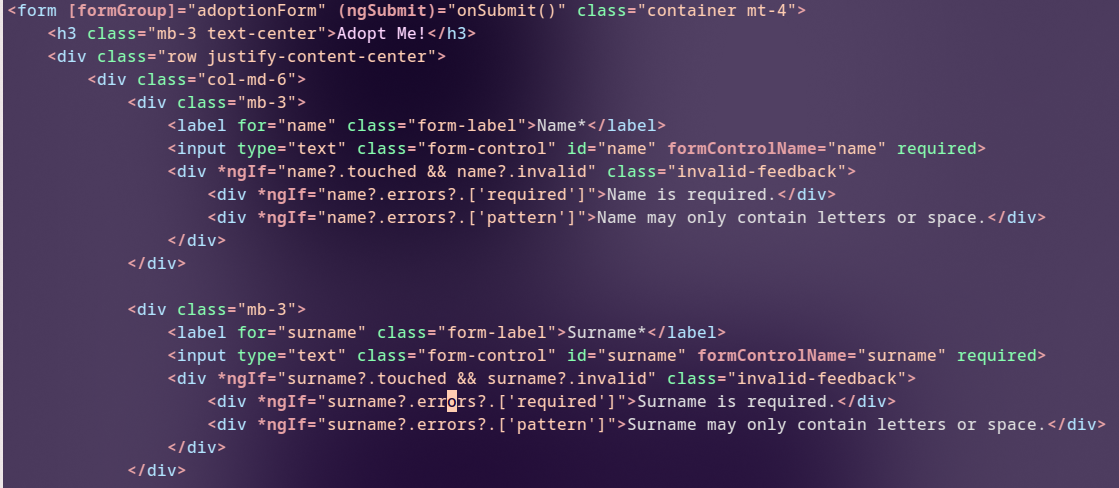
TODO: Projekt sadrži modele Animal, User. Modeli su ključni za definiranje strukture podataka koji će biti korišteni u aplikaciji. Kod njih se možda i najbolje vide velike prednosti koje TypeScript pruža u odnosu na vanilla JavaScript. Naime, kod definira koji će sve podaci ili atributi biti uključeni, te specificira njihove tipove. Na taj način se osigurava type safety, smanjuje vjerojatnost nastanka bugova, te osigurava očuvanje konzistentnosti strukture podataka (jer programski kod ne može negdje drugdje ad hoc izostaviti ili dodati suvišni atribut).



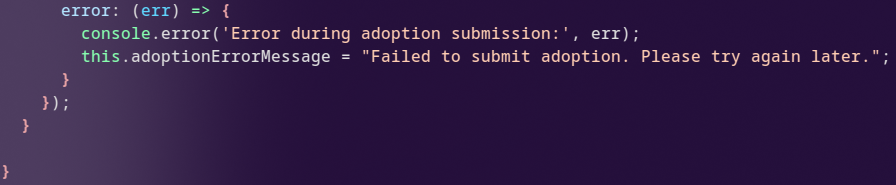
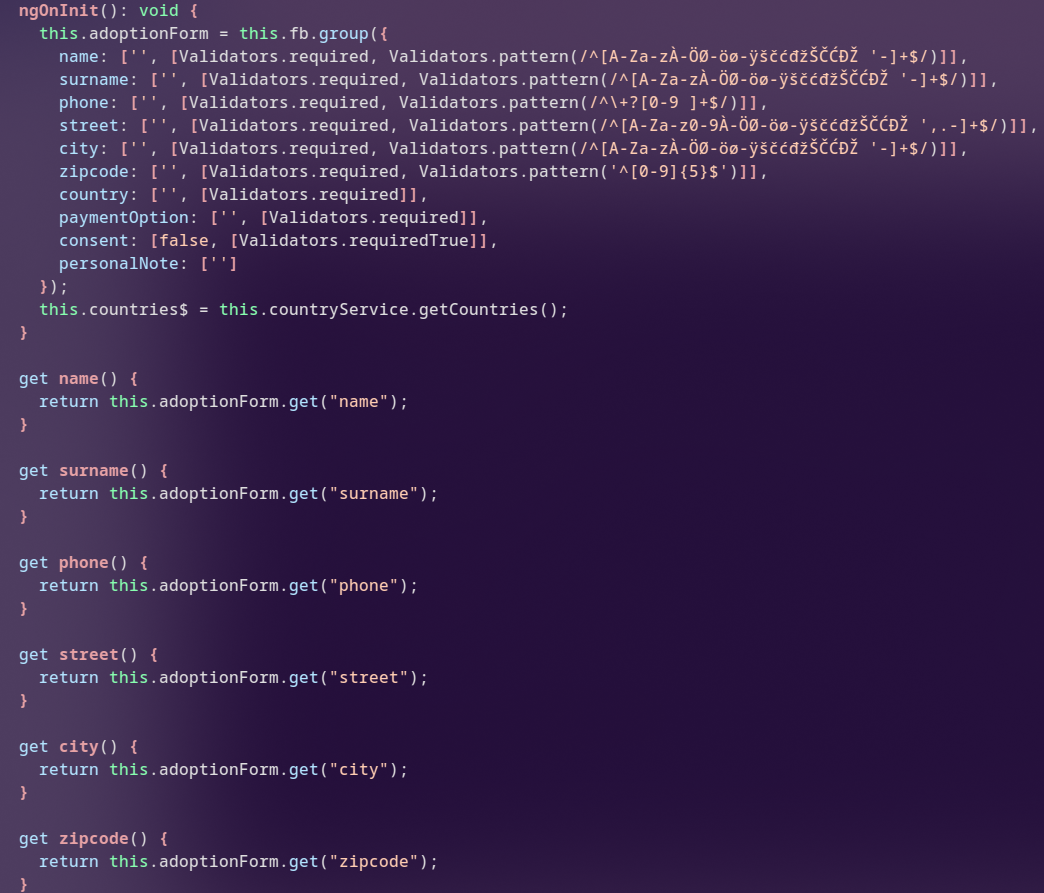
## Primjer koda za izabranu komponentu

TODO Ispod se daje prikaz AdoptionForm, koja predstavlja jednu od dvije komponente u okviru stranice Animal Details.

HTML:



Programski kod:

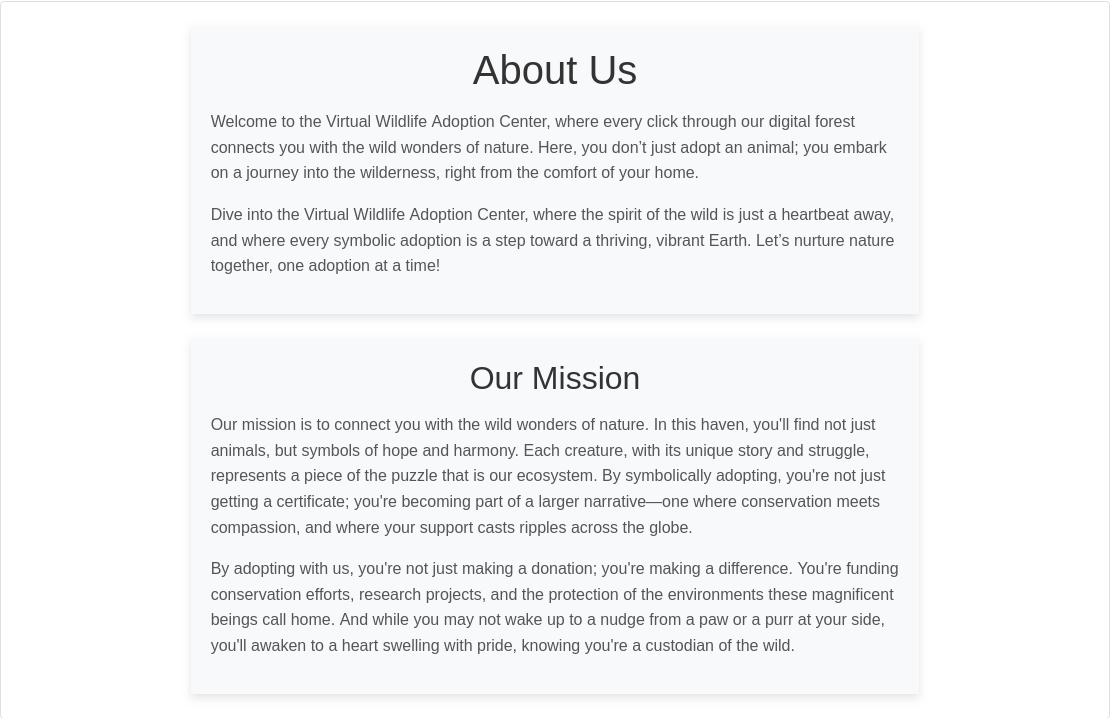


# Screenshotovi

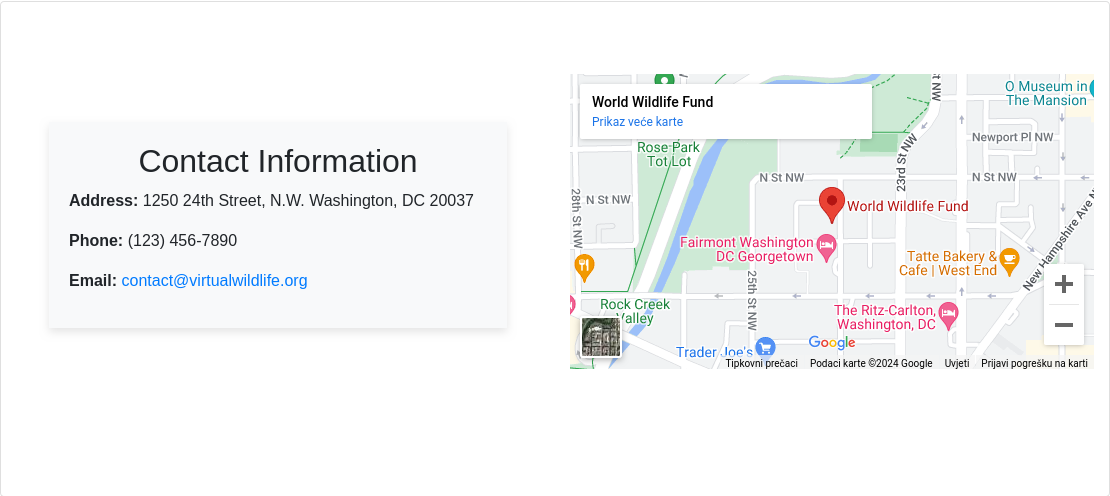
## Početna stranica

TODO

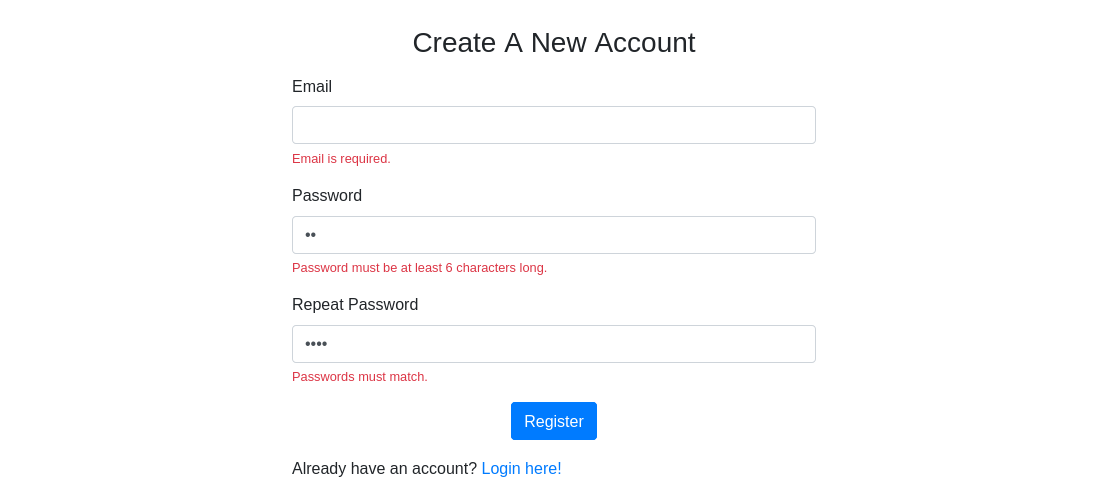
## About Us



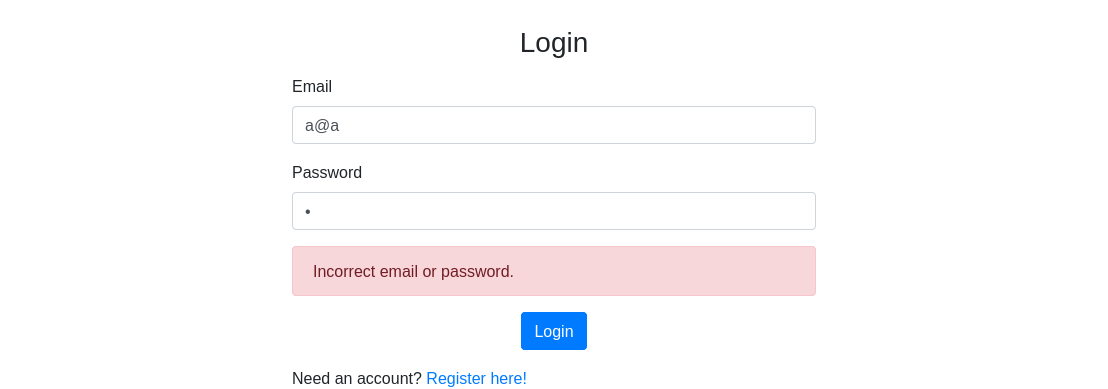
## Contact Us



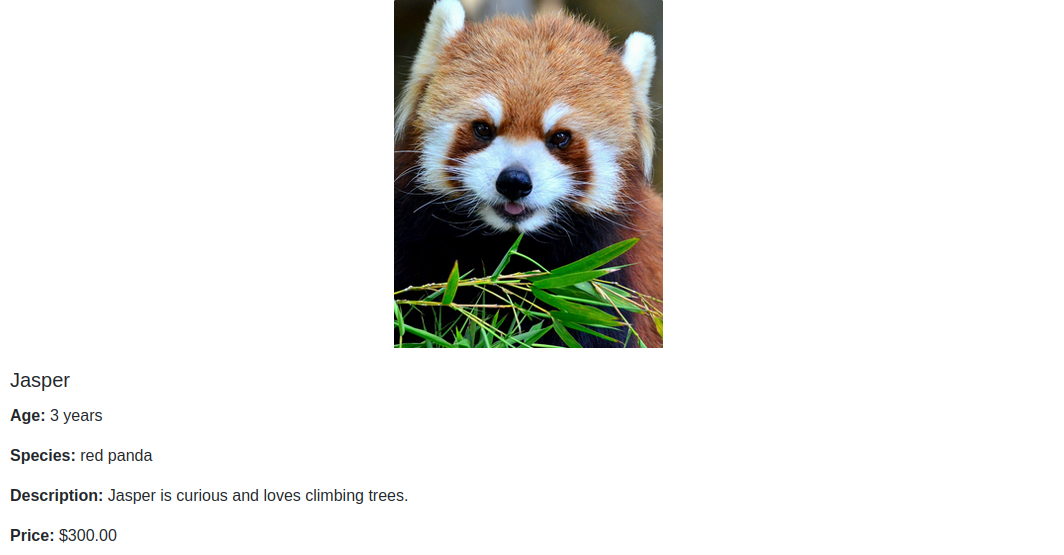
## Register



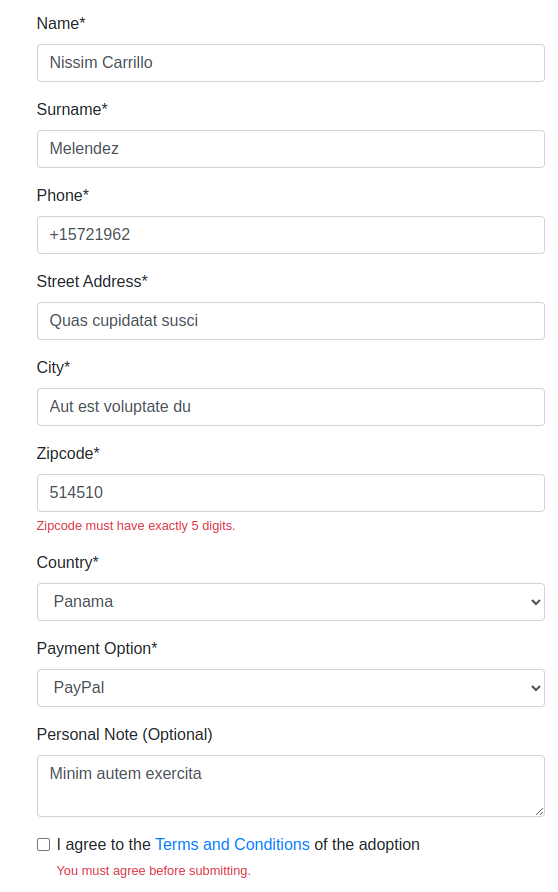
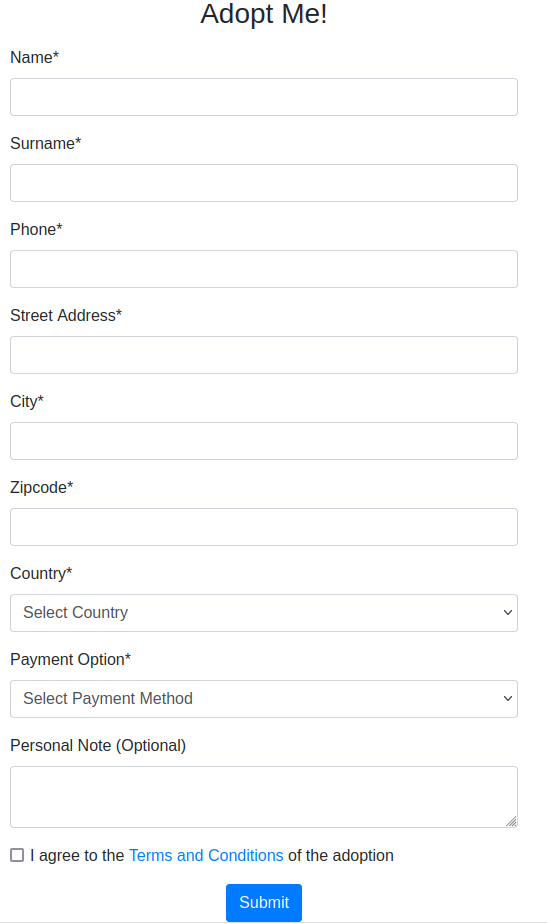
## Login



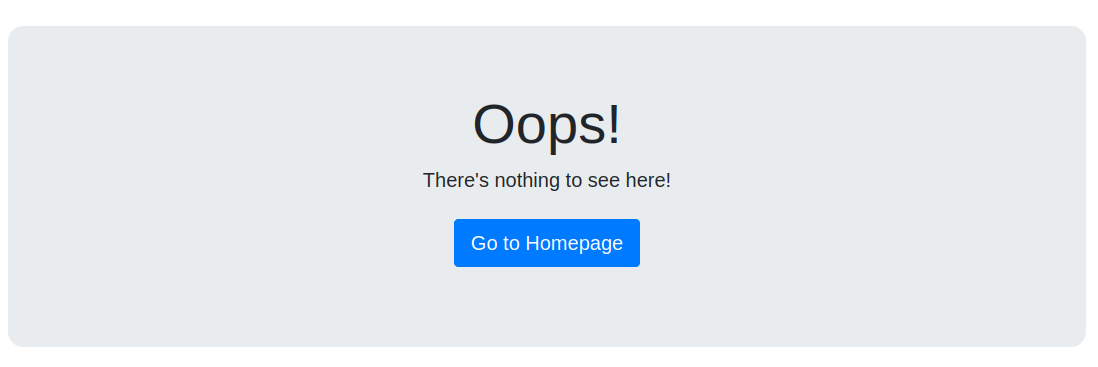
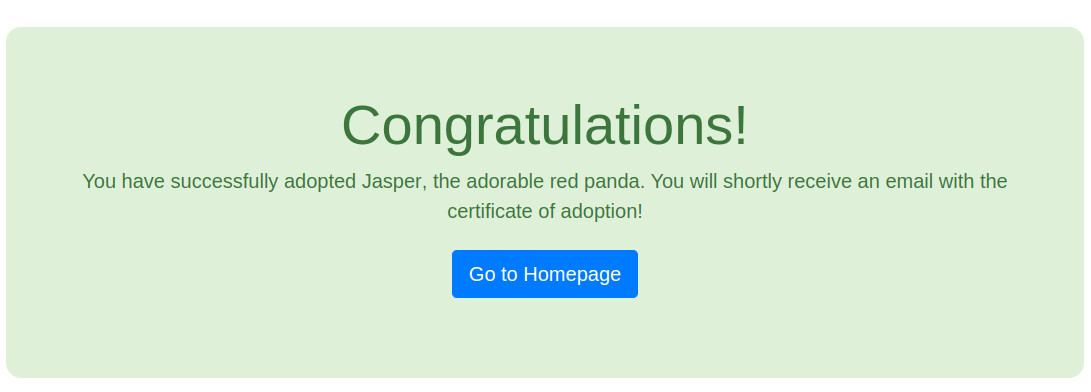
## Detalji o životinji



## Posvajanje (unutar stranice detalja)



## Potvrda o posvajanju



## Admin CRUD

TODO

# Zaključak

U sklopu ovog projekta je kreiran plan, obavljen razvoj te testiranje web sistema, korištenjem tehnologija kao što su Angular 15, json-server, Bootstrap i sl. Projekt demonstrira na koje načine Angular olakšava kreiranje jednostavnog master-detail sitea, posebno u odnosu na korištenje običnog (vanilla) JavaScripta.

Svakako, potrebno je istaknuti da postoji još dosta prostora za poboljšanje sistema opisanog u ovom dokumentu. Neki od njih su razmatrani tokom razvoja, ali nisu uključeni u finalni projekat, zbog nedostatka vremena, i/ili veće razine kompleksnosti.

Sljedeći koraci bi, između ostalog, uključivali:

* refaktoriranje koda kako bi u još većoj mjeri bio u skladu sa dobrim praksama razvoja Angular aplikacija (uvođenje modula, lazy loading-a, ...)
* detaljnija i preciznija specifikacija zahtjeva, posebno nefunkcionalnih, kako bi zadovoljavali što više karakteristika odličnih zahtjeva, prema kriterijima koje daje Karl Wiegers
* implementacija backenda, primjerice u Java Springu
* sigurna implementacija registracije i logiranja, tj. da se ni u kom momentu ne prenosi plaintext lozinka
* projektiranje i korištenje prave baze podataka, umjesto dosadašnje JSON mock baze
* integraciju sa servisima za sigurno plaćanje
* slanje potvrdnog e-maila sa certifikatom o posvajanju
* implementaciju različitih validnih formata poštanskih i telefonskih brojeva, ovisno o korisnikovoj lokaciji
* poboljšanje pretraživanja, npr. dodavanje opcija sortiranja (abecedno, po cijeni...) i filtriranja
* unapređenje stilizacije, korištenjem npr. Angular Material UI ili Semantic UI, a umjesto Bootstrapa, koji u svojim defaultnim postavkama daje "generičan" izgled

# Reference

1. Google. (2023, September 25). Introduction to Angular concepts. Angular Documentation. https://angular.io/guide/architecture
2. Google. (2017). Architecture Overview - ts - GUIDE. Angular v2 Archive. https://v2.angular.io/docs/ts/latest/guide/architecture.html
3. Introduction - ngx-datatable. <https://siemens-com.gitbook.io/ngx-datatable/>
4. Formly. https://formly.dev/