

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)

[Zahtjevi 3](#_Toc25)

[Funkcionalni zahtjevi 3](#_Toc26)

[Nefunkcionalni zahtjevi 3](#_Toc27)

[Faza projektiranja 3](#_Toc28)

[Arhitektura 3](#_Toc29)

[Konceptualni dijagram 4](#_Toc30)

[JSON 5](#_Toc31)

[Implementacija 5](#_Toc32)

[Struktura projekta 5](#_Toc33)

[AppModule i AppComponent 5](#_Toc34)

[Router 5](#_Toc35)

[Servisi 5](#_Toc36)

[Modeli 5](#_Toc37)

[Primjer koda za izabranu komponentu 6](#_Toc38)

[Screenshotovi 6](#_Toc39)

[Zaključak 6](#_Toc40)

[Reference 6](#_Toc41)

# 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 frameworka (verzija 15), RxJS-a, Bootstrap-a te HTML5. 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, i vidjeti više informacija o izabrani životinji
* Registrirani korisnici – mogu pregledati katalog životinja, vidjeti više informacija o izabrani životinji, te ju po želji posvojiti (nakon popunjavanja forme)
* Administratori

## Dijagram

## Opis slučajeva korištenja

# Zahtjevi

### Funkcionalni zahtjevi

### Nefunkcionalni zahtjevi

# 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.

### JSON

Ispod se daje JSON struktura identificiranih entiteta.

# 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 - "velike" komponente koje se sastoje od više pojedinačnih komponenti
* services
* models

## AppModule i AppComponent

## Router

Implementiran je AuthGuard, kojim se ograničava pristup određenim stranicama.

## Servisi

TODO: Projekt sadrži AnimalService, UserService, . Svima njima zajednička je komunikacija sa json-serverom putem HttpClienta, te korištenje RxJS-a.

## 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

# Screenshotovi

# 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
* unapređenje stilizacije, korištenjem npr. Angular Material UI ili Semantic UI, a umjesto Bootstrapa, koji u svojim defaultnim postavkama daje "generičan" izgled
* 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

# 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