# Uzgoj biljaka na ekološki način uz potporu web aplikacije

**Matokanović, Valentin**

**Master's thesis / Diplomski rad**

**2023**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:200:237945

*Rights / Prava:* In copyright / Zaštićeno autorskim pravom.

*Download date / Datum preuzimanja:* **2025-03-24**

*Repository / Repozitorij:*

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK**

**Sveučilišni studij**

# UZGOJ BILJAKA NA EKOLOŠKI NAČIN UZ POTPORU WEB-APLIKACIJE

**Diplomski rad**

**Valentin Matokanović**

**Osijek, 2023.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

**Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit**

**Osijek, 17.09.2023.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za diplomski ispit

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Valentin Matokanović |
| **Studij, smjer:** | Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika' |
| **Mat. br. Pristupnika, godina upisa:** | D-1373, 07.10.2021. |
| **OIB studenta:** | 87186830340 |
| **Mentor:** | prof. dr. sc. Dominika Crnjac Milić |
| **Sumentor:** | prof. dr. sc. Dominika Crnjac-Milić |
| **Sumentor iz tvrtke:** | Helena Stjepanović |
| **Predsjednik Povjerenstva:** | izv. prof. dr. sc. Alfonzo Baumgartner |
| **Član Povjerenstva 1:** | prof. dr. sc. Dominika Crnjac-Milić |
| **Član Povjerenstva 2:** | doc. dr. sc. Tomislav Galba |
| **Naslov diplomskog rada:** | Uzgoj biljaka na ekološki način uz potporu web aplikacije |
| **Znanstvena grana diplomskog rada:** | **Programsko inženjerstvo (zn. polje računarstvo)** |
| **Zadatak diplomskog rada:** | U današnje vrijeme javlja se sve veća potreba ljudi za bavljenje uzgojem biljaka koje su za konzumaciju, ali i onih koje imaju funkciju ukrašavanja njihovog okruženja te pročišćavanja zraka. Prisutan je trend uzgoja i održavanja biljaka na ekološki način no znanje o tome je često ograničeno i slabo dostupno. Kako bi se promicala svijest o važnosti, povećao broj zainteresiranih te dijelilo znanje o tome ovim radom treba izraditi web aplikaciju koja je interaktivna, jednostavna za korištenje i prilagodljiva za sve vrste uređaja (responzivni dizajn). Potrebno je dizajnirati i realizirati odgovarajuću bazu podataka koju će aplikacija koristiti za pohranu svih potrebnih podataka. Putem aplikacije bi zainteresirani korisnici mogli komunicirati. Mogli bi dijeliti informacije |
| **Prijedlog ocjene pismenog dijela ispita (diplomskog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene od strane mentora:** | 17.09.2023. |

| | |
|---|---|
| Potvrda mentora o predaji konačne verzije rada: | *Mentor elektronički potpisao predaju konačne verzije.* |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 29.09.2023.**

| | |
|---|---|
| **Ime i prezime studenta:** | Valentin Matokanović |
| **Studij**: | Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika' |
| **Mat. br. studenta, godina upisa:** | D-1373, 07.10.2021. |
| **Turnitin podudaranje [%]:** | 2 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Uzgoj biljaka na ekološki način uz potporu web aplikacije**

izrađen pod vodstvom mentora prof. dr. sc. Dominika Crnjac Milić

i sumentora prof. dr. sc. Dominika Crnjac-Milić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

**1. UVOD** ..... **1**

**2. UZGOJ BILJAKA NA EKOLOŠKI NAČIN** ..... **3**

**2.1. Načela uzgoja biljaka na ekološki način** ..... **3**

**2.2. Princip uzgoja biljaka na ekološki način u praksi** ..... **4**

**2.3. Prednosti ekološkog uzgoja biljaka** ..... **7**

**3. PREGLED WEB APLIKACIJA ZA UZGOJ BILJAKA** ..... **9**

**3.1. Prednosti korištenja web aplikacija u poljoprivredi** ..... **9**

**3.2. Pregled i analiza postojećih rješenja** ..... **10**

**4. ARHITEKTURA WEB APLIKACIJE** ..... **15**

**4.1. Odabrane tehnologije za web-aplikaciju uzgoja biljaka na ekološki način** ..... **18**

4.1.1. React ..... 19

4.1.2. Express ..... 20

4.1.3. Node ..... 21

4.1.4. MongoDB ..... 21

**5. BAZA PODATAKA I UPRAVLJANJE PODACIMA** ..... **22**

**5.1. Modeliranje baze podataka** ..... **22**

**6. FUNKCIONALNOSTI WEB APLIKACIJE** ..... **24**

**6.1. Registracija i prijava korisnika** ..... **24**

**6.2. Modeliranje i upravljanje biljkama** ..... **27**

6.2.1. Modeliranje i upravljanje razgovorima ..... 30

**6.3. Modeliranje i upravljanje oglasima** ..... **34**

**6.4. Međuslojevi (engl. *middlewares*)** ..... **38**

**7. DIZAJN KORISNIČKOG SUČELJA** ..... **40**

**7.1. Responzivni web dizajn** ..... **41**

**7.2. Navigacijska traka** ..... **43**

**7.3. Stranice za registraciju i prijavu korisnika** ..... **45**

**7.4. Stranice za pregled i dodavanje biljaka** ..... **48**

**7.5. Stranice za pregled i dodavanje oglasa** ..... **55**

**8. ZAKLJUČAK** ..... **60**

**LITERATURA** ..... **62**

**SAŽETAK** ..... **65**

**ABSTRACT** ..... **66**

**PRILOZI** ..... **67**

**ŽIVOTOPIS** ..... **68**

# 1. UVOD

U današnje vrijeme, kada su klimatski i ekološki izazovi sve teži, ekološki uzgoj biljaka nastoji doprinijeti očuvanju okoliša i zdravlja ljudi. Uzgoj biljaka na ekološki način ne samo da dovodi do uzgoja zdravijih i hranjivijih namirnica, nego i do održivosti i učinkovitosti uzgoja biljaka te smanjenja negativnog utjecaja na okoliš. Sve veći broj ljudi prepoznaje važnost očuvanja okoliša i okreće se tome kako bi očuvali vlastito zdravlje i prirodu.

Zbog navedenih razloga, važno je promovirati i informirati ljude o takvom načinu uzgoja. Ponekad je teško pronaći informacije, a upravo taj problem nastoji se riješiti razvojem ove web aplikacije. Važno je podržati i educirati korisnike o ekološkom uzgoju biljaka. To je potrebno učiniti na način da ljudi prepoznaju važnost ekološkog uzgoja, usvoje takve metode i na kraju krajeva doprinesu krajnjem cilju očuvanja okoliša.

Da bi se to ostvarilo na takav način, važno je omogućiti korisnicima web aplikacije jednostavan i lak pristup informacijama o ekološkom uzgoju biljaka. Tim informacijama može pristupiti svaki korisnik, bio on registriran ili ne, što otvara vrata svim ljudima da naprave prvi korak. Također, kako bi ju mogli ugodno koristiti apsolutno svi korisnici, aplikacija je responzivna za sve vrste uređaja. Osim samih informacija o uzgoju raznih biljaka, omogućena je i komunikacija samih korisnika. To dovodi do stvaranja jedne zajednice koja se međusobno podržava, savjetuje i dijeli znanje o takvom načinu uzgoja. Nadalje, aplikacija omogućuje proizvođačima oglašavanje raznih proizvoda vezanih uz uzgoj na ekološki način, što je vrlo korisno za obje strane, kako za prodavače, tako i za same korisnike. Korisnici nakon informiranja o načinu uzgoja mogu odmah pronaći proizvode koji su im potrebni. Na taj način korisnici će podržati domaće ljude i OPG-ove (obiteljska poljoprivredna gospodarstva), a ujedno će brzo i jednostavno pronaći sve što im je potrebno.

Cilj ovog diplomskog rada je razviti web aplikaciju koja je jednostavna za korištenje, interaktivna i prilagodljiva za sve vrste uređaja. Motivacija za izradu ove web aplikacije je promocija svijesti o važnosti ekološkog uzgoja, povećanje broja zainteresiranih te dijeljenje znanja o ekološkom uzgoju. Potrebno je dizajnirati i realizirati odgovarajuću bazu podataka koju će aplikacija koristiti za pohranu svih potrebnih podataka. Putem aplikacije zainteresirani korisnici trebaju imati mogućnost komunikacije, dijeljenja informacija o vrstama biljaka koje uzgajaju ili planiraju uzgajati, dijeljenja savjeta itd. Putem aplikacije treba se omogućiti oglašavanje proizvođačima proizvoda koji pomažu ekološkom uzgoju biljaka te pružanje informacija o korisnosti njihovih proizvoda korisnicima.

U drugom poglavlju analizirana je i opisana metoda ekološkog uzgoja biljaka. Objašnjena su načela, principi (procesi) te prednosti ekološkog uzgoja. Treće poglavlje se odnosi na analizu sličnih postojećih rješenja. Opisano je svako rješenje kako bi se moglo usporediti s rješenjem u ovom diplomskom radu. Četvrto poglavlje opisuje arhitekturu web aplikacije te izabrane tehnologije za razvoj ove web aplikacije. U petom poglavlju je opisana baza podataka, vrste baza podataka te odabir i modeliranje baze podataka za ovu web aplikaciju. U šestom poglavlju opisane su funkcionalnosti web aplikacije, odnosno metode za upravljanje podacima i interakciju s bazom podataka (poslužiteljski dio). U sedmom poglavlju prikazan je dizajn cijele web aplikacije, odnosno korisničko sučelje (klijentski dio). Zadnje poglavlje odnosi se na zaključak koji sadrži kratki pregled i analizu postignutih rješenja, razloge korištenja odabranih tehnologija te mogućnosti daljnjeg razvoja.

# 2. UZGOJ BILJAKA NA EKOLOŠKI NAČIN

Uzgoj biljaka na ekološki način je agrikulturalni sustav kojemu je cilj pružiti svojim potrošačima svježe, prirodne i zdrave proizvode. Takav uzgoj naziva se još i organskim, prirodnim, biološkim, alternativnim i tradicionalnim. Da bi se ostvario, potrebno je primjenjivati određena pravila i principe pri uzgoju biljaka s ciljem minimalnog utjecaja na okoliš, promovirajući prirodne cikluse i održive prakse. U takvom uzgoju biljaka potpuno je zabranjena upotreba GMO-a (genetski modificiranih organizama), a ograničena je upotreba umjetnih (kemijskih i sintetičkih) gnojiva, pesticida i regulatora rasta. Važno je zaštititi tlo, zrak, biljne, genetske i životinjske resurse te da proizvodnja bude ekonomski održiva. Kako bi biljke bile uzgojene na ekološki način, uzgoj mora biti u skladu s Pravilnikom o ekološkoj proizvodnji u uzgoju bilja i u proizvodnji biljnih proizvoda.

## 2.1. Načela uzgoja biljaka na ekološki način

Prema Pravilniku o ekološkoj proizvodnji u uzgoju bilja i u proizvodnji biljnih proizvoda, članku 2., uzgoj biljaka na ekološki način temelji se na sljedećim načelima:

„ - *očuvanju biološke i krajobrazne raznolikosti, posebice stabilnosti prirodnih staništa i očuvanju samoniklih biljnih vrsta,*

- *usklađivanju i pravilnom gospodarenju glede izbora usjeva, biljnih vrsta i sorti, višegodišnjih plodoreda, odabira načina obrade tla, gnojidbe i zaštite te jačanja otpornosti na štetočinje,*

- *brizi za pravilnu njegu tla; čuvanju i povećanju njegove plodnosti i biološke aktivnosti, sadržaja organskih tvari i hraniva, poboljšanju strukture tla te postupcima njegove zaštite od raznih oblika degradacije,*

- *zaštiti korisnih organizama: oprašivača, predatora, ptica i drugih,*

- *ekološki opravdanoj preradi i uporabi (recikliranju) otpada iz proizvodnje,*

- *proizvodnji koja isključuje ili samo iznimno dopušta uporabu agrokemikalija (mineralnih gnojiva i raznih kemijskih sredstava za zaštitu bilja).”* [1]

Ta načela se konkretno odnose na:

- očuvanje prirode, surađivanje s njom te poštivanje njenih zakona i ritmova (npr. proizvodnja koja je prilagođena utjecajima ekoloških čimbenika nekog krajolika, kao što su toplina, zrak, voda i svjetlost)

- poticanje biološke aktivnosti unutar gospodarstva (aktivnosti u kojima sudjeluju mikroorganizmi, životinjski svijet, flora i fauna tla)
- smanjenje korištenja fosilnih goriva (korištenje obnovljivih izvora energije te ponovna upotreba otpada)
- proizvodnja proizvoda koji imaju visoke nutritivne vrijednosti i poželjno je da su ekonomski dostupni svima
- važnost povezivanja prirode i čovjeka
- briga za zdravlje svojih potrošača
- fokus je na optimalnoj proizvodnji, a ne maksimalnoj (cilj ne treba biti isključivo maksimiziranje profita, nego i poštivanje svih načela ekološkog uzgoja, iako je profit manji) [2, str.1]

## 2.2. Princip uzgoja biljaka na ekološki način u praksi

U praksi, uzgoj biljaka na ekološki način sadrži mnogo različitih procesa koji se trebaju poduzeti kako bi uzgoj zaista bio ekološki. Ti koraci uključuju utvrđivanje ekoloških čimbenika, razne agrobiotehničke zahvate i ostale procese koji doprinose oblikovanju uvjeta uzgoja. Biljke se razvijaju u dva ambijenta, u dijelu koji je u tlu (podzemni dio, korijen) te u dijelu koji je iznad tla, odnosno u atmosferi (nadzemni dio).

Pojam „stvaranje snage“ podrazumijeva stvaranje zemljanih i nadzemnih uvjeta koji potiču rast zdravih biljaka, suzbijanje štetnih organizama i promicanje korisnih organizama. Cilj je osigurati dovoljnu biološku raznolikost i uvjete za zdrav rast biljaka – ispod i iznad zemlje, tj. u prostoru (npr. granice usjeva, udaljenost između redova itd.) i u vremenu (npr. pokrovni usjevi, rotacije itd.)

Prema izvoru [3, str.111], radnje koje stvaranje snage iznad zemlje obuhvaća su:

- odabrati usjeve i sorte biljaka otporne na lokalne štetočine (ali i uz druge kvalitete kao što su urod, okus itd.)
- koristiti odgovarajuću gustoću sadnje
- napraviti granice polja i zone unutar polja koje privlače korisne kukce. To obično uključuje sadnju mješovitog cvjetnog bilja oko ili unutar polja, kako bi se napravilo sklonište i hranu za korisne kukce.
- koristiti pokrovne usjeve zbog nekoliko prednosti kao što su staništa za korisne kukce, dodavanje dušika i organske tvari u tlo, smanjivanje erozije i povećavanje infiltracije vode u tlo

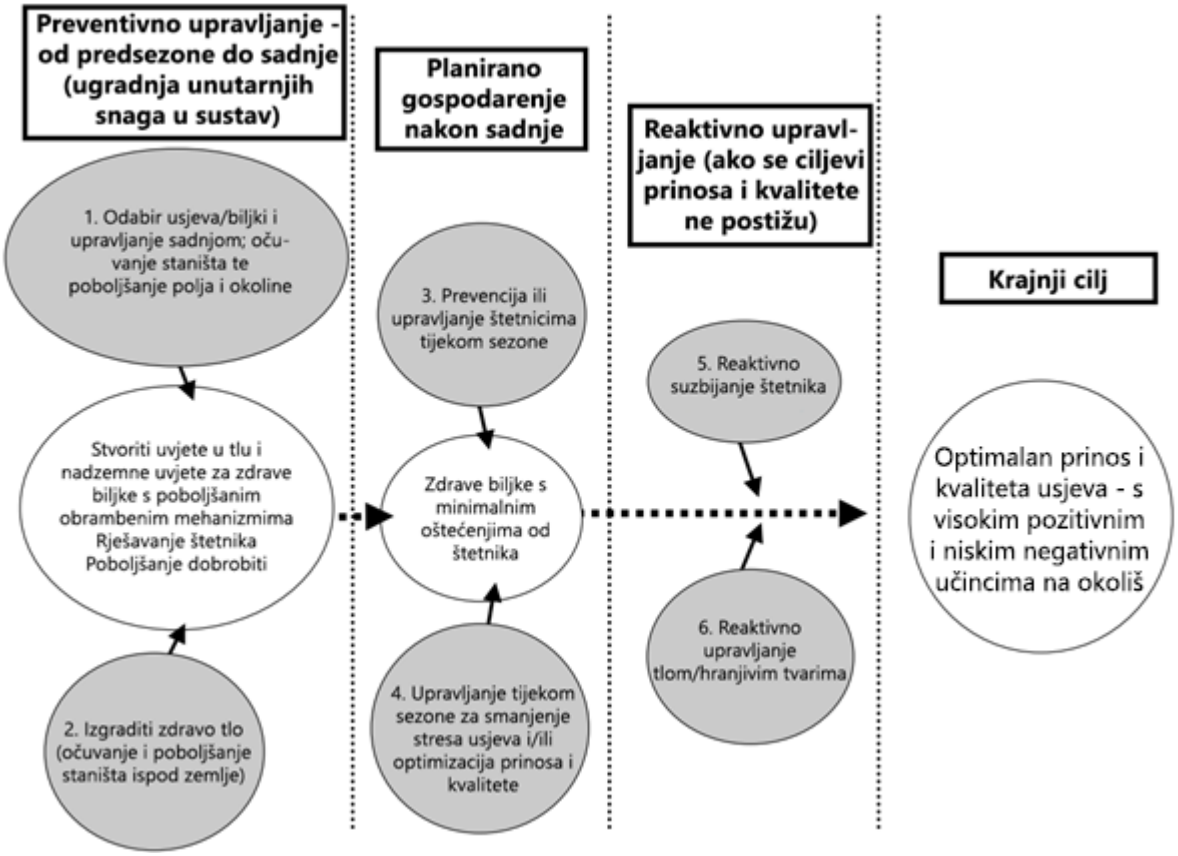
Također, prema izvoru [3, str.112], radnje koje stvaranje snage tla obuhvaća su:

- dodati velike količine organskog materijala na regularnoj bazi – životinjski gnoj, kompost, lišće drveća, pokrovni usjev itd. Organske tvari u tlu i njihovo upravljanje je srce dobrog zdravog tla koje ima iznimnu snagu da donosi zdrave biljke koje imaju dobre obrambene mehanizme.
- koristiti različite organske materijale jer svi oni imaju različite učinke na biološka, kemijska i fizikalna svojstva tla. Biološka svojstva tla podrazumijevaju mikroorganizme i makroorganizme koji žive u tlu, a koji su zaduženi za usitnjavanje i razgradnju organskog tla, dreniranje, prozračivanje tla itd. Kemijska svojstva tla podrazumijevaju reakciju tla, odnosno njegovu pH vrijednost koja određuje sposobnost tla za vezivanje i iskorištavanje tla za hranjenje biljke. Fizikalna svojstva podrazumijevaju dubinu tla, teksturu i strukturu tla, obradu tla itd.
- pobrinuti se da tlo bude pokriveno sa živom vegetacijom i/ili ostacima usjeva koristeći pokrovne usjeve, busene trave ili smanjenu obradu tla

Rutinske ekološke prakse tijekom sezone uključuju aktivnosti kao što su provjeravanje ima li problema sa štetočinama i briga o navodnjavanju tla, ako je količina kiše nedovoljna. Obrezivanje stabala kako bi se smanjila vlažnost (uz to i gljivične bolesti) je također jedna od tih praksi, kao i kultiviranje za suzbijanje korova.

Iako se ekološki uzgoj temelji na preventivnom upravljanju, u nekim slučajevima je potrebno i reaktivno upravljanje da se spase usjevi, ako unatoč preventivnim mjerama, ipak dođe do problema sa štetočinama ili nekim drugim. U takvim situacijama, oslobađanje korisnih biokontrolnih organizama je najbolji pristup, iako će možda i mala količina nisko toksičnog pesticida biti potrebna. Korištenje folijarnog prskanja za brzi unos hranjivih tvari u biljku tijekom sezone može pomoći u svladavanju nekih nedostataka koji se pojavljuju tijekom rasta biljaka. Reaktivno upravljanje je zadnja linija obrane u ekološkom uzgoju biljaka. [4]

Na slici 2.1. prikazane su faze i procesi, odnosno sustav uzgoja biljaka na ekološki način. Svaka faza je bitna i svi procesi se nadovezuju jedni na druge kako bi se u konačnici ostvario cilj ekološkog uzgoja.




**Sl. 2.1.** Procesi sustava ekološkog uzgoja biljaka. Izradio autor prema izvoru [3, str.112]

Što se tiče tehnoloških čimbenika u praksi, izuzetno je važan odabir agrotehničkih zahvata. Potrebno ih je odabrati prema načelima ekološkog uzgoja, a osnovni agrotehnički zahvati su plodored, obrada tla i gnojidba.

Plodored je obavezna agrobiotehnička mjera i neizostavan je dio ekološkog uzgoja. Predstavlja prostornu i vremensku izmjenu redova u poljima, a sastoji se od procesa kao što su plodosmjena, ophodnja ili rotacija i odmor tla. Plodosmjena se odnosi na vremensku izmjenu kultura, odnosno na to da se na istom polju svake godine uzgaja druga kultura. Ophodnja se odnosi na prostornu izmjenu kultura, odnosno ista kultura se svake iduće godine premješta na neko drugo polje. Te prakse se izvode zbog toga što prečesti uzgoj jedne kulture na istom mjestu svake godine uzrokuje tzv. umor tla. Kako bi tlo odmorilo, primjenjuje se uzgoj travno-djetelinskih smjesa. [5]

Obrada tla predstavlja mehaničke zahvate oruđima u tlu s ciljem stvaranja povoljnijih vodozračnih odnosa koji su pogodniji za razvoj biljaka. U ekološkom načinu uzgoja, obrada tla teži prema tome da se tlo ne prevrće te da se ne miješa više slojeva tla. Prevrtanjem tla dolazi do toga da se

površinski sloj tla koji je bogat humusom, prozračan i sadrži mikroorganizme, prenosi dublje u zemlju, a donji sloj, koji nema povoljne uvjete za rast biljaka se prenosi na površinu. Također, nepravilna obrada tla povećava eroziju, a time i smanjuje količinu organskih tvari, pa se u uzgoju na ekološki način preporuča svesti obradu na minimum ili ju niti ne provoditi. Preporuka je koristiti pokrovne usjeve ili malču koji će štititi površinu tla od nepovoljnih atmosferskih utjecaja (pljuskovi, tuča, vjetar i sl.). Duboka obrada tla se provodi u kasnu jesen ili zimu, a u proljeće se još dodatno tlo poravnava i usitnjava te se na taj način priprema za sadnju. [2]

U ekološkom uzgoju je zabranjeno korištenje vrsta gnojiva koja sadrže ljudski izmet, ostatke GMO biljaka i mikroorganizama, gnojiva s konvencionalnih gospodarstava te mineralna gnojiva. Dozvoljena je uporaba organskih gnojiva (domaća s ekoloških gospodarstava i komercijalna gnojiva prirodnog porijekla), komposta i primjena zelene gnojidbe. Kompost je smjesa organskih otpadaka prerađenih od strane mikroorganizama i faune. Zelena gnojidba predstavlja unošenje nadzemne mase određenih biljaka u tlo, a kojima je svrha obogatiti tlo organskim tvarima i povećati njegovu biološku aktivnost. Najčešće se koristi kod vrsta tla koja imaju nedostatak humusa te na gospodarstvima koja ne uzgajaju stoku. [5]

## 2.3. Prednosti ekološkog uzgoja biljaka

Ekološki uzgoj biljaka ima pozitivan utjecaj na klimu, zdravlje, pa čak i ekonomiju. Prednosti su razne, ali ove tri stavke su one najbitnije.

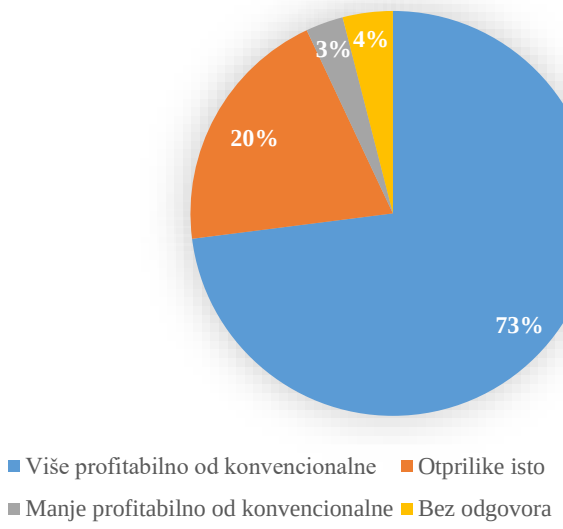
Što se tiče klime, ekološki uzgoj smanjuje stakleničke plinove (ugljični dioksid, metan, didušikov oksid itd.) eliminacijom većine fosilnih inputa temeljenih na gorivu te gradi otpornost na klimatske promjene promicanjem zdravog tla, diverzifikacijom prehrambenih usjeva te podržavanjem ugroženih staništa divljih životinja i bioraznolikosti. Podaci pokazuju da ekološki uzgoj ispušta manje dušikovog oksida zbog toga što se ne koriste kemijska gnojiva i pesticidi, za razliku od konvencionalne poljoprivrede. Također, to doprinosi i stočarskoj proizvodnji gdje dolazi do smanjenja emisije metana u usporedbi s konvencionalnim hranjenjem životinja. Izgradnjom zdravih tla koja zadržavaju vodu i pohranjuju ugljik, ekološki način uzgoja gradi otpornost i stabilizira opskrbu hranom u slučaju suše i drugih loših vremenskih uvjeta koji će se u budućnosti sve češće javljati u klimatskim promjenama. [6]

Zdravlje je važan dio ekološkog uzgoja biljaka. Istraživanja pokazuju da ljudsko zdravlje ima prednosti od ekološkog uzgoja jer se dramatično smanjuje izloženost poljoprivrednom onečišćenju u zraku, vodi i hrani. Poljoprivredni radnici i drugi ljudi koji žive u blizini konvencionalnih farmi mogu patiti od akutnih i kroničnih zdravstvenih tegoba koje su nastale zbog izloženosti

pesticidima, a studije pokazuju da ostaci pesticida u uzgojenim plodovima mogu biti štetni za zdravlje potrošača. Osim toga što ekološki uzgoj štiti zdravlje tako što izbacuje toksine iz okoliša i plodova, također proizvodi zdravije, robusnije usjeve koji imaju poboljšane nutritivne prednosti i ukusnije plodove. Zanimljiva činjenica je da voće i povrće uzgojeno na ekološki način sadrži veće razine antioksidansa koji pomažu u borbi protiv raka. Razlika je dovoljna da se može reći da ima znatan utjecaj na zdravlje i nutritivnost. Općenito, može se reći da ekološki pristup poboljšava zdravlje svega; poljoprivrednika, potrošača, ekosustava i okoliša. [7]

Iako je očuvanje okoliša i zdravlje samo po sebi dovoljna motivacija za ekološki uzgoj, također je dokazano da je i ekonomski isplativije, bar za manje poljoprivrednike. Baš zbog toga što ovaj način uzgoja ne koristi kemijska gnojiva, pesticide, fungicide ili herbicide, pa nije potrebno trošiti sredstva na njih. Sve što je potrebno za uzgoj može se reciklirati i iskoristiti resurse u prirodi. Također, može proizvesti prinose koji su u najmanju ruku usporedivi s konvencionalnim metodama, a često su i veći. Plodovi se zbog načina uzgoja mogu prodavati po malo višim cijenama kao što i zaslužuju, jer su svježi, zdravi i organski. Čak i istraživanja pokazuju da ekološka proizvodnja stvara nova radna mjesta, smanjuje nezaposlenost i potiče rast poljoprivrednog poslovanja. Međutim, industrija se još uvijek ne predaje tome jer ne žele pružiti priliku, a s ekonomske strane, za velike tvrtke je ipak isplativije kupiti jeftine kemikalije i proizvoditi što više plodova radi veće zarade.

Na slici 2.2. prikazani su rezultati ankete koju je 2007. godine provela organizacija „Minnesota Department of Agriculture“ u kojoj su pitali farmere kakav je profit ekološkog uzgoja u odnosu na konvencionalni. Anketa se nalazi u izvoru [8, str.3].




**Sl. 2.2.** Rezultati ankete vezane uz profit ekološkog uzgoja. Izradio autor prema izvoru [8, str.3]

# 3. PREGLED WEB APLIKACIJA ZA UZGOJ BILJAKA

Pretraživanjem i analizom nekih postojećih web aplikacija vezanih uz uzgoj biljaka može se izvući inspiracija za razvoj ove web aplikacije. Stječe se uvid u prednosti korištenja web aplikacija, kao i uvid u određene funkcionalnosti koje se mogu koristiti i implementirati u ovoj web aplikaciji.

## 3.1. Prednosti korištenja web aplikacija u poljoprivredi

Korištenje web aplikacija u poljoprivredi donosi niz prednosti, posebno u kontekstu ekološkog načina uzgoja. Web aplikacija može poboljšati produktivnost, učinkovitost i održivost ekološke proizvodnje na razne načine.

Poboljšan i brz pristup informacijama je ono što bi svaka web aplikacija trebala ispuniti. Korisnicima omogućuju brz i jednostavan pristup raznim informacijama vezanim za uzgoj biljaka na ekološki način. To uključuje informacije o vrstama biljaka, plodoredima, zaštiti i jačanju bilja, organskim gnojidbama te općenitim metodama za primjenu ekološkog načina uzgoja. Te informacije mogu biti vrijedne za poljoprivrednike, OPG-ove, ali i za pojedince koji u vlastitom vrtu ili polju žele uzgajati biljke. Omogućuje im da donose bolje odluke i poboljšaju svoje metode uzgoja.

Postoje web aplikacije koje su prilagodljive i personalizirane u smislu da pružaju personalizirane savjete i informacije na temelju specifičnih potreba i uvjeta korisnika. Na temelju unesenih podataka kao što su lokacija, klima, vrsta tla i vrsta biljke, web aplikacije mogu generirati prilagođene metode uzgoja. Na primjer, ukoliko je toplija klima ili je drugačija vrsta tla, možda je potrebno više navodnjavanja i slično.

Važno je osvijestiti korisnika i o proizvodima koji se koriste kod načina uzgoja. Web aplikacija također može ponuditi informacije o proizvodima za ekološki uzgoj, kao i trgovinu tim proizvodima. To omogućuje korisnicima da imaju sve na jednom mjestu; informacije o metodama uzgoja i proizvode koji se koriste u tim metodama.

Edukacija i dijeljenje znanja su također važni aspekti koje treba uzeti u obzir. Nije dovoljno samo pružiti što više informacija, nego je bitna i kvaliteta tih informacija. Načini uzgoja su promjenjivi i napreduju kao i sve ostalo, stoga je ažuriranje informacija također bitno. Prednost web aplikacije je i dijeljenje znanja samih korisnika. Korisnici mogu međusobno dijeliti informacije i savjete te pitati druge korisnike određena pitanja. U slučaju ove web aplikacije, teorijske informacije

ponekad nisu dovoljne, pa iskustva pojedinih korisnika mogu doprinjeti i pomoći riješiti neke situacije.

## 3.2. Pregled i analiza postojećih rješenja

Postoji nekoliko web aplikacija koja su vrlo slične i iz kojih se može izvući inspiracija za razvoj ove web aplikacije. Neke od njih su: Smart Gardener, The Old Farmer's Almanac Garden Planner i Gardenate. Svaka od njih ima neke prednosti i specifičnosti koje mogu pomoći u uzgoju biljaka.

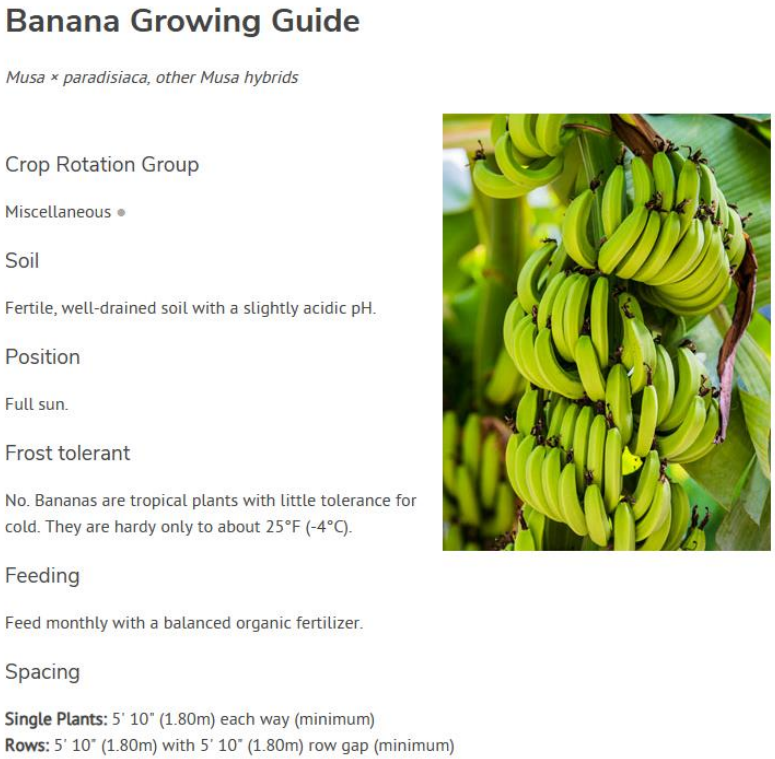
Smart Gardener je web aplikacija čiji je cilj olakšati i automatizirati organizaciju vrta. Omogućuje izradu planova uzgoja uzimajući u obzir mnogo varijabli kao što su: klima, vrsta tla, preferirane biljke itd. Cilj je odraditi sav „teški“ posao organizacije umjesto korisnika, što uključuje predlaganje biljaka, optimizirani plan vrta koji kombinira odabrane biljke, raspored sadnje biljaka itd. Još jedna prednost je što pruža listu zadataka koje bi korisnik trebao obaviti. U toj listi zadataka, korisnik se obavještava kada treba zalijevati biljke, posaditi, pripremiti, obrati itd. Nakon obavljene aktivnosti, može se označiti da je obavljena i na taj način uskladiti stvarnu situaciju i onu koja se nalazi digitalno u web aplikaciji. Nedostatak ove web aplikacije je što se korištenje plaća, ali s obzirom na značajke koje pruža i način na koji olakšava korisniku uzgoj biljaka, može se zaključiti da je razumno. Smart Gardener web aplikacija je dostupna putem izvora [9].

Na slici 3.1. prikazan je primjer zaslona s aktivnostima koje je potrebno obaviti. Za svaku aktivnost piše o kojoj se biljci radi, vrsta radnje, uputa, datum kada bi bilo poželjno obaviti te oznaku je li aktivnost obavljena.

**Sl. 3.1.** Primjer zaslona za listu zadataka. Slika preuzeta iz izvora [9]

The Old Farmer's Almanac Garden Planner je web aplikacija vrlo slična prethodnoj. Također omogućuje generiranje plana uzgoja biljaka za vrt. Njihov plan se adaptira s obzirom na lokacijsko područje korisnika jer imaju bazu podataka od preko 5000 vremenskih stanica. Prema tome, može generirati raspored oranja, sadnje i branja/sjetve biljaka. Osim toga, u web aplikaciji nalazi se baza podataka s velikim brojem vrsta biljaka. Za svaku biljku imaju informacije i upute na koji način tu biljku uzgajati. Web aplikacija sadrži i mnoštvo videozapisa vezanih za uzgoj i obradu tla kako bi prikazali korisniku kako odraditi neke aktivnosti. Osim baze podataka za metode uzgoja raznih biljaka, postoji i baza podataka za nametnike i bolesti biljaka. To je vrlo korisno za korisnika jer može identificirati nametnike i bolesti i reagirati prema uputama. Nedostatak je što se planer plaća, ali sve ostale značajke su besplatne. The Old Farmer's Almanac Garden Planner web aplikacija je dostupna putem izvora [10].

Na slici 3.2. prikazan je primjer stranice u web aplikaciji na kojoj pišu upute za uzgoj banane. Definirani su parametri poput pozicije, otpornosti na smrzavanje, gnojidbe, razmak itd.

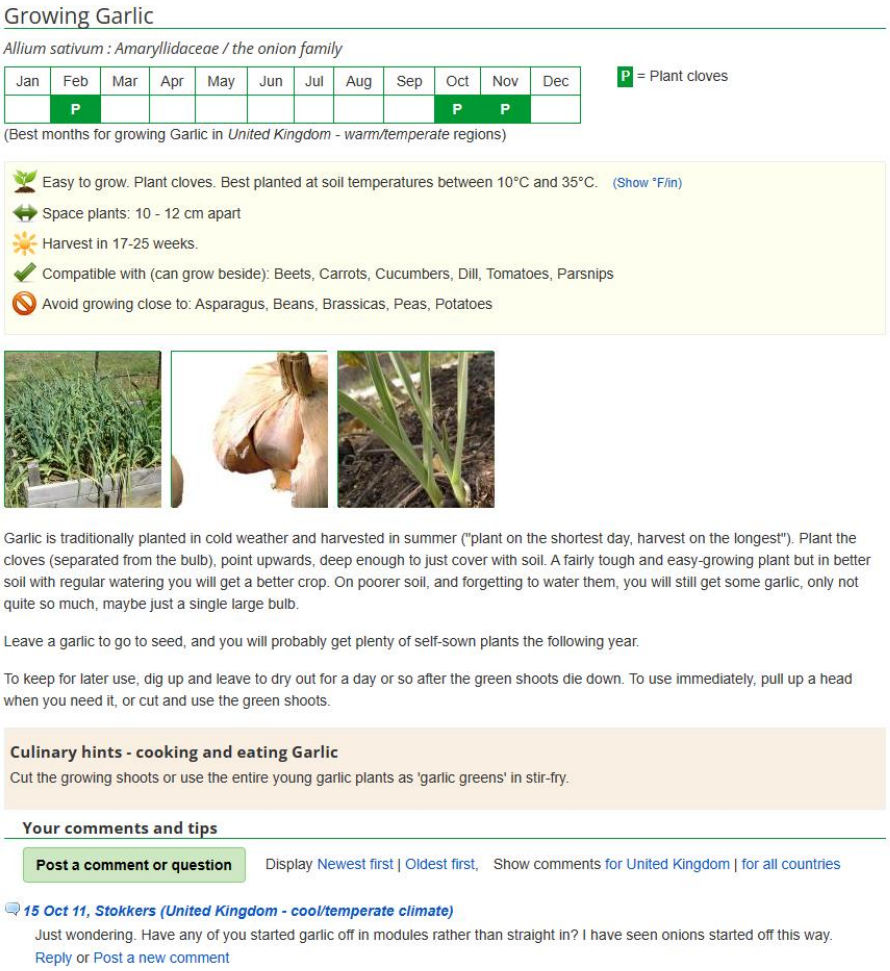



**Sl. 3.2.** Primjer stranice za metodu uzgoja biljke. Slika preuzeta iz izvora [10]

Gardenate web aplikacija ima nešto drugačiji pristup nego prethodne dvije. Ona više služi kao baza podataka za uzgoj biljaka, nema nikakav alat (npr. planer) koji automatizira i znatno olakšava korisniku organizaciju vrta. Međutim, s obzirom na mjesec u godini i odabranu klimu, prikazuje

korisnicima one biljke koje se trenutno sade. Sadrži veliku bazu podataka s uputama za uzgoj svake biljke. Na tim stranicama je omogućena i međusobna komunikacija korisnika u vezi te biljke u obliku komentara i odgovora, što je dobra funkcionalnost, jer ljudi dijele svoja iskustva i savjete te mogu pomoći jedni drugima. Prednost je što su sve funkcionalnosti dostupne besplatno, ali nedostatak je što ih nema puno kao u prethodnim web aplikacijama. Može se reći da ona samo pruža informacije, bez interaktivnosti s korisnikom. Još jedan nedostatak je što sadrži bazu podataka samo za biljke koje su povrće i ljekovito bilje, nema informacija o ostalim vrstama biljaka. Gardenate web aplikacija je dostupna putem izvora [11].

Na slici 3.3. je prikazan primjer stranice na kojoj je opisana metoda uzgoja češnjaka. Stranica prikazuje vrijeme sadnje, kratke natuknice, fotografije i detaljniji opis metode uzgoja biljke. Ispod tog dijela nalaze se komentari korisnika koji komuniciraju u vezi te biljke.




**Sl. 3.3.** Primjer stranice za metodu uzgoja biljke. Slika preuzeta iz izvora [11]

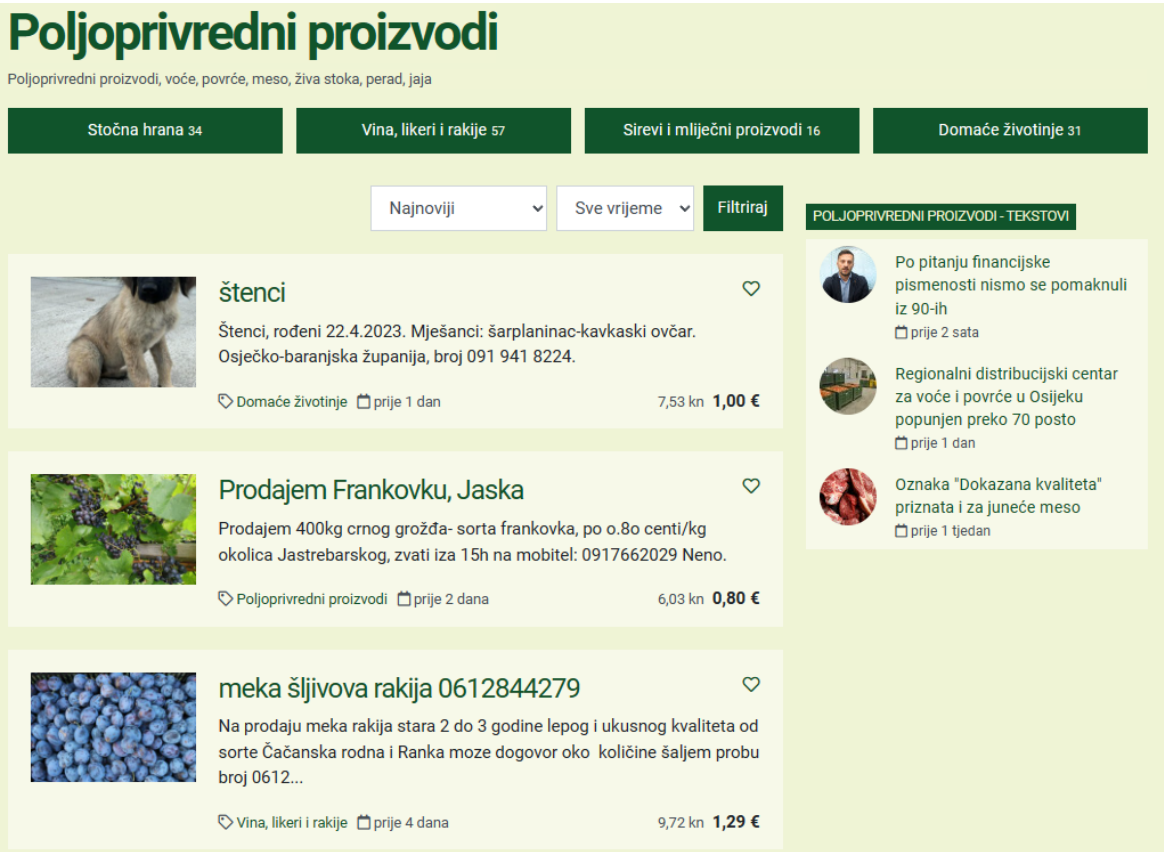
Agrivi web aplikacija predstavlja digitalnu platformu za poljoprivredu. U svojoj web aplikaciji prezentiraju softver koji omogućuje nadzor proizvodnje biljaka u stvarnom vremenu, evidenciju proizvodnje, poljoprivrednu analitiku, sljedivost proizvoda te povezivanje sa senzorima i sustavima. U ponudi imaju 3 glavna alata. Farm Insights alat je jednostavan za korištenje dizajniran za podršku poljoprivrednicima u donošenju agronomskih odluka na temelju uvida u teren u stvarnom vremenu. Osim toga, olakšava administraciju i organizaciju farme. Farm Enterprise alat služi za kontroliranje složenih operacija, donošenje odluka temeljenih na podacima u stvarnom vremenu, praćenje tržišta, optimiziranje troškova itd. Agriculture Supply Chain je alat za upravljanje poljoprivrednim lancem nabave, a olakšava taj proces digitaliziranim procesom nabave, olakšanim upravljanjem kooperacijom i ugovorima itd. Može se zaključiti da su njihovi alati usmjereni prema tvrtkama, odnosno većim poljoprivrednim proizvođačima. Također, svi alati se naplaćuju. No, kako bi se ti alati optimalno koristili, potrebno je implementirati sustav senzora jer na temelju tih podataka će alati dati najbolja rješenja. Agrivi web aplikacija je dostupna putem izvora [12].

Na slici 3.4. prikazan je njihov alat Farm Insights u kojem se može upravljati s poljima, financijama, resursima itd.

**Sl. 3.4.** Zasloni aplikacije Farm Insights. Slika preuzeta iz izvora [12]

Agroklub web aplikacija je zapravo najsličnija web aplikaciji u ovom diplomskom radu. Glavni dijelovi su poljoprivredni portal, oglasnik i baza padataka. Poljoprivredni portal sadrži članke i diskusije na razne poljoprivredne teme. Registrirani korisnici mogu ostavljati komentare i raspravljati na portalu. Oglasnik funkcionira slično kao i svaki drugi oglasnik. Korisnici mogu filtrirati, sortirati i pretraživati proizvode. Svaki oglas sadrži razne informacije kao što su slike, podaci o oglašivaču, opis, specifikacije itd. Također, web aplikacija sadrži i bazu podataka koja nudi informacije ne samo o raznim vrstama biljaka, nego i o zaštitnim sredstvima, gnojivima, stočarstvu itd. Tu se mogu naći korisne informacije kako se koriste pojedini proizvodi za uzgoj, kao i metode uzgoja pojedinih biljaka s detaljnim opisima. Iako ova web aplikacija ne nudi nikakav alat koji automatizira uzgoj, može se zaključiti da je vrlo korisna zbog obujma informacija koje pruža. Korištenje se ne naplaćuje, no kako bi se koristile sve mogućnosti, potrebno je registrirati korisnički račun (besplatno). Agroklub web aplikacija je dostupna putem izvora [13].

Na slici 3.5. prikazan je primjer stranice za oglase poljoprivrednih proizvoda. Moguće je filtrirati prema kategorijama i sortirati, a odabirom pojedinog oglasa otvara se stranica sa svim detaljima o tom oglasu.

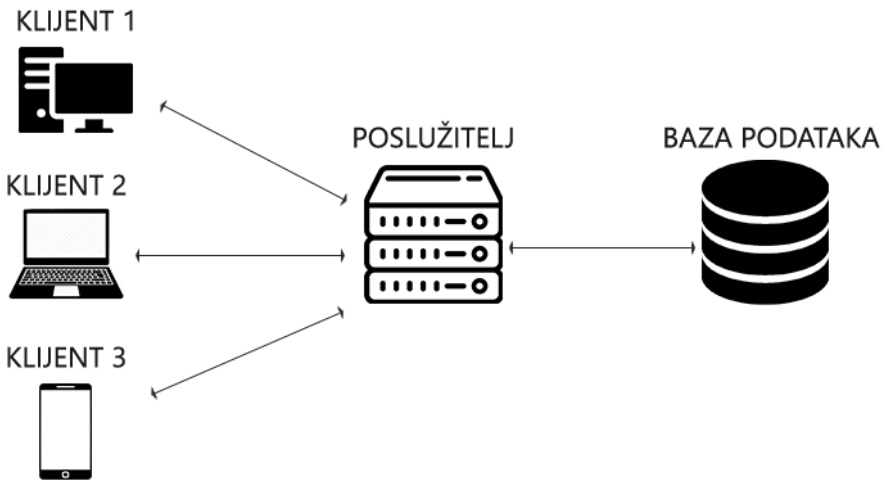



**Sl. 3.5.** Primjer stranice za oglase proizvoda. Slika preuzeta iz izvora [13]

# 4. ARHITEKTURA WEB APLIKACIJE

Web-aplikacija u ovom diplomskom radu je temeljena na klijentsko-poslužiteljskoj arhitekturi. Klijentsko-poslužiteljska arhitektura je najraširenija arhitektura koju koristi skoro svaka web aplikacija. Jako malo stranica koristi drugačiju vrstu arhitekture koja se naziva engl. *Peer to Peer*.

Klijent predstavlja računalo/uređaj, koji zapravo koristi servis i prihvaća informacije. Uređaji koji mogu biti klijent su računala, radne stanice, IoT (engl. *Internet of Things*) uređaji i sl. Server ili poslužitelj predstavlja udaljeno računalo koje pruža pristup podacima i servisima. To su obično fizički uređaji kao što su engl. *rack* poslužitelji, iako je porast računalstva u oblaku donio i virtualne poslužitelje. Poslužitelj odrađuje procese kao što su e-pošta, hosting aplikacija, internet konekcije, ispis itd. [14]




**Sl. 4.1.** Klijentsko-poslužiteljska arhitektura. Izradio autor prema izvoru [14, str.67]

Slika 4.1. prikazuje pojednostavljenu klijentsko-poslužiteljsku arhitekturu. Klijentsko-poslužiteljska arhitektura se odnosi na sustav koji poslužuje, dostavlja i upravlja većinom resursa i servisa koje klijent zahtjeva. U tom modelu, svi zahtjevi i servisi su dostavljeni preko mreže. U principu, klijent šalje zahtjev poslužitelju za određenu informaciju, a poslužitelj odgovara na to slanjem te informacije. [14]

Također, može se reći da je aplikacija temeljena na *frontend* tehnologiji koja je vezana za klijentsku stranu te *backend* tehnologiji koja je vezana za poslužiteljsku stranu web aplikacije. Današnje moderne web aplikacije ne mogu se koristiti bez ovih tehnologija.

*Frontend* je dio web aplikacije koji korisnik vidi i s kojim vrši interakciju, kao što su izbornici, kontakt obrasci, navigacija itd. U principu, to i jest klijentski dio aplikacije. Kako bi se dizajnirao

i razvio *frontend* dio aplikacije, potrebno je koristiti određene tehnologije, a to je u najjednostavnijem smislu kombinacija HTML (engl. *Hypertext Markup Language*), CSS (engl. *Cascading Style Sheets*) i JavaScripta (kontrolirani od strane web-preglednika). Kombinacijom tih tehnologija može se stvoriti korisničko sučelje s kojim korisnici vrše interakciju i koje pruža što bolji pregled i jasnoću korištenja web aplikacije.

Međutim, u današnje vrijeme rijetko se koristi čista kombinacija HTML, CSS i JavaScript-a. Razvijene su tehnologije koje se nazivaju okviri (engl. *frameworks*), a neki od njih su React, Vue.js i Angular. Te tehnologije pojednostavljuju mnoge aspekte poput strukture koda i održavanja, ponašajući se kao kostur web aplikacije. *Framework* sadrži standardizirani unaprijed napisani kod, koji olakšava i ubrzava razvoj specifične funkcionalnosti. [15]

Angular je potpuni *frontend framework*, React je UI (engl. *User Interface*) biblioteka, a Vue.js je progresivni *framework*. Oni nisu isti, pa ih ima smisla uspoređivati i razumjeti njihove razlike. Svaki od njih se temelji na komponentama i omogućuje brzu izradu značajki korisničkog sučelja. Međutim, svi oni imaju različite strukture i arhitekturu. [15]

*Backend* je s druge strane, odgovoran za obradu podataka, komunikaciju s bazom podataka i pružanje funkcionalnosti koje podržavaju rad aplikacije. Obično se sastoji od tri dijela: poslužitelj (server), aplikacija i baza podataka. *Backend* tehnologije se obično sastoje od programskih jezika kao što su PHP (engl. *Hypertext Preprocessor*), Java, JavaScript, Ruby, Python i sl. Ti jezici se nazivaju još i poslužiteljski (engl. *server-side*) jezici i koriste se za implementaciju logike i obradu zahtjeva korisnika na strani poslužitelja. Omogućavaju izradu dinamičkih web aplikacija, pružaju mogućnost upravljanja korisničkim sesijama, autorizaciju i autentifikaciju korisnika, pristup bazi podataka itd.

Naravno, postoje i *backend framework* tehnologije koje povećavaju razinu kvalitete i standardizaciju web aplikacija. Svaki *framework* je različit i ima svoje prednosti i mane. Neki su bolji za manje projekte, dok su neki bolji za veće i zahtjevnije projekte. Neki su sigurniji, dok drugi imaju kvalitetniju i opsežniju dokumentaciju. Neki od *backend frameworka* su: Django, Node.js, Spring, Laravel, Symphony, Rails itd. [16]

Baze podataka su također važan dio *backend* dijela web aplikacije. One su ključne za pohranu i upravljanje podacima u web aplikacijama. Dijele se na relacijske i nerelacijske baze podataka, odnosno na SQL (engl. *Structured Query Language*) i NoSQL (engl. *Not Only Structured Query Language*).

SQL je programski jezik koji se koristi za upravljanje podacima u relacijskim bazama podataka još od 1970-ih godina. U ranoj fazi, dok je pohrana bila skupa, SQL baze podataka su bile usmjerene na smanjenje dupliciranja podataka. U današnje vrijeme SQL se još uvijek široko koristi za postavljanje upita relacijskim bazama, gdje su podaci pohranjeni u retke i tablice koje su povezani na razne načine. Jedan zapis tablice može biti povezan jedan s drugim ili više drugih, ili više zapisa tablice može biti povezano s više zapisa u drugoj tablici. Ove relacijske baze podataka, koje nude brzu pohranu i oporavak podataka, mogu podnijeti velike količine podataka i kompleksne SQL upite. Prema tome, SQL baze podataka važne su za rukovanje strukturiranim podacima ili podacima koji imaju odnose između varijabli i entiteta. Neke od relacijskih baza podataka su: MySQL, PostgreSQL i Oracle. [17]

NoSQL je nerelacijska baza podataka, što znači da dozvoljava različite strukture od SQL baze podataka (ne koristi retke i stupce) i pruža veću fleksibilnost za korištenje formata koji najbolje odgovara podacima. Koristi se za pohranu nestrukturiranih podataka i brz pristup podacima u stvarnom vremenu. Međutim, ne znači da NoSQL uopće ne koristi SQL, ponekad NoSQL baze podataka podržavaju SQL. Preciznije, definira se kao „ne samo SQL“ odnosno engl. „*not only SQL*“. Neke od nerelacijskih baza podataka su: MongoDB, Cassandra i Redis. [18]

Također za web aplikaciju je važan i API (engl. *Application Programming Interface*). On omogućava komunikaciju između različitih komponenti softvera, odnosno može se reći da upravo on povezuje *frontend* i *backend* web aplikacije. *Backend* razvoj uključuje dizajn i implementaciju API-a kako bi se omogućila komunikacija između *frontend* i *backend* dijela aplikacije. API se koristi u mnogim slučajevima, primjerice za dobivanje podataka za web aplikaciju ili povezivanje s udaljenim poslužiteljem koje ima podatke koji se neprestano mijenjaju ili za omogućavanje da dvije aplikacije međusobno razmjenjuju podatke. [19]

Postoji više vrsta API-a, kao što su SOAP (engl. *Simple Objects Access Protocol*), JavaScript, RESTful API (engl. *REpresentational State Transfer*) itd. Kod web aplikacija najčešće se koristi tzv. RESTful API koji koristi HTTP protokol za slanje zahtjeva i primanje odgovora. Također, API može biti javni i privatni.

API se općenito može razmatrati na temelju ovih 6 stavki:

1. Općenite informacije o API-u - ime API-a, opis, kategorija, datum ažuriranja, URL i broj operacija

2. Vrsta API-a – detalji o tome je li opis API-a RESTful, RPC ili hibridni

3. Ulazni parametri – koristi li API zadane parametre, koristi li opcionalne parametre, kodirane parametre (npr. umjesto „Croatian“ koristi „hr“), koristi li parametre s alternativnim vrijednostima (npr. ulazna vrijednost je 1, 2 ili 3) itd.

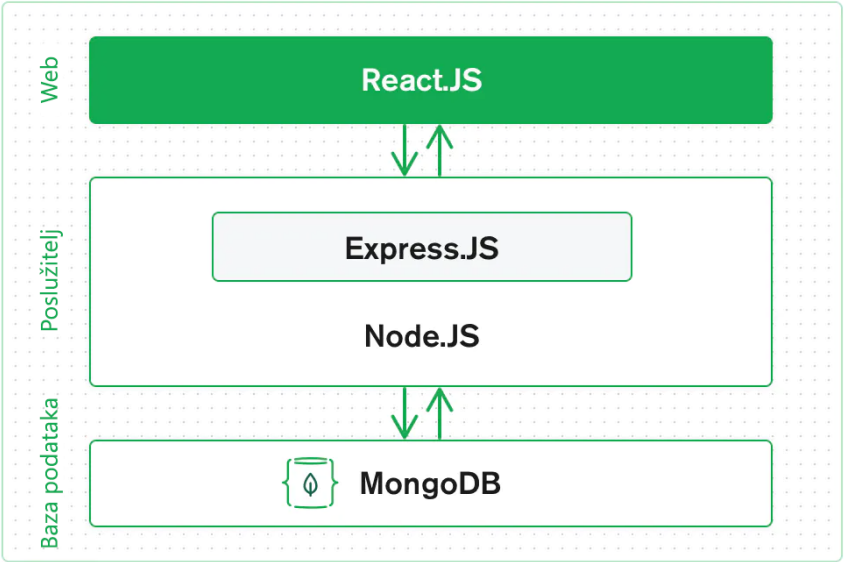
4. Izlazni formati – oblik izlaza (npr. JSON ili XML) i šalje li se kao parametar

5. Detalji o pozivanju – je li navedena HTTP metoda, je li naveden URI pozivanja, zahtjeva li API autentifikaciju te ako zahtjeva, koji je tip autentifikacije, kako se prenose ulazni parametri i kako se prenose informacije o autentifikaciji

6. Dopunska dokumentacija – sadrži li opis primjer zahtjeva, primjer odgovora i popis poruka/šifri pogrešaka [19, str.108]

## 4.1. Odabrane tehnologije za web-aplikaciju uzgoja biljaka na ekološki način

Za web-aplikaciju u ovom diplomskom radu odabran je skup tehnologija koji se naziva MERN, što je skraćenica od MongoDB, Express, React i Node.js. MongoDB je baza podataka temeljena na dokumentima, Express je Node.js web *framework*, React je JavaScript biblioteka za klijentsku stranu, Node.js je JavaScript poslužiteljsko okruženje otvorenog koda. MERN je *full-stack* (način razvoja aplikacije koji uključuje i *backend* i *frontend*) arhitektura koja pruža jednostavnu i laku konstrukciju troslojne arhitekture (*frontend*, *backend* i baza podataka) u potpunosti koristeći JavaScript i JSON.




**Sl. 4.2.** Princip rada MERN skupa tehnologija. Izradio autor prema izvoru [22]

Slika 4.2. prikazuje međusobnu povezanost i komunikaciju između tehnologija MERN skupa. React.JS predstavlja klijentski dio (*Web*), Express.JS i Node.JS predstavljaju poslužitelj, a MongoDB predstavlja bazu podataka. Klijentski dio komunicira s bazom podataka preko poslužitelja, koji se može smatrati posrednikom između te dvije strane. Komunikacija je dvosmjerna.

### 4.1.1. React

React predstavlja gornji sloj MERN skupa tehnologija. React je deklarativna JavaScript biblioteka koju je razvio Facebook (današnja Meta) i koristi se za stvaranje dinamičkih aplikacija na strani klijenta u HTML-u. Omogućuje izgradnju složenih korisničkih sučelja putem jednostavnih komponenti, njihovo povezivanje s podacima na poslužitelju i renderiranje njih kao HTML elemenata. Potiče formiranje dijelova korisničkog sučelja za višekratnu upotrebu, koji pokazuju informacije koje napreduju, odnosno mijenjaju se nakon nekog vremena. Brzo se ažurira renderiranjem važnih dijelova za čitanje svakog stanja i mijenja te informacije unutar web aplikacije. [20]

Neke od glavnih značajki React-a su:

- deklarativnost – React čini inteligentno i dinamičko korisničko sučelje (UI – engl. *User Interface*) za web-stranice i prijenosne aplikacije. Napravi se osnovna perspektiva za svako stanje unutar aplikacije, a React će učinkovito osvježavati dijelove aplikacije kada se određena informacija promjeni. Konačne perspektive čine kod uočljivijim i jednostavnijim za otklanjanje pogrešaka u kodu.
- kratka i laka krivulja učenja – jednostavna i nekompleksna priroda React-a omogućuje brzo snalaženje sa strukturom. Razumljivost je vjerojatno najbolja kvaliteta React-a. Međutim, preporuča se određeno poznavanje JavaScripta kako bi se lakše shvatio React.
- jednosmjerno povezivanje podataka – React odlikuje jednosmjerni protok podataka između stanja i slojeva u aplikaciji. To znači da podaci mogu teći u jednom smjeru između stanja aplikacije i slojeva. S druge strane, u dvosmjernom povezivanju podataka kao što je Angular, ako se promijeni model, promijenit će se pogled i obrnuto. Prednost ograničenja informacija u jednom smjeru jest bolja kontrola nad svim komponentama kroz cijelu web aplikaciju.
- virtualni DOM – poseban način sastavljanja koda u Reactu je rezultat virtualnog DOM-a (engl. *Virtual Document Object Model*). To funkcionira na način da u trenutku kada se komponenta ponovno isporuči, React ju zamjenjuje njezinim prošlim prikazom i ažurira

izvorna središta DOM-a koja su se promijenila. Taj ciklus zamjene naziva se kompromis. Prema tome, u tom ciklusu React odabire svoju produktivnost da napravi DOM čvorište, osvježi postojeći DOM ili usitni prethodni i stvori novi nadograđeni. To daje veliku prednost što se tiče prikazivanja, pogotovo kada se moraju izvršiti bezbrojne promjene podataka.

- performanse – React je poznat po izvrsnim performansama. To omogućuje upravo virtualni DOM. DOM je programski API koji upravlja HTML-om, XML-om ili XHTML-om. DOM postoji skupa u memoriji sa svim ostalim. Upravo zbog toga, kada se napravi komponenta, ona se ne komponira izravno u DOM. Ako su sve stvari jednake, sastavljaju se virtualni dijelovi koji se pretvaraju u DOM što će potaknuti glatku i bržu izvedbu. [20, str.699]

U principu, velika prednost React-a je rukovanje sučeljima koja se temelje na stanju, s minimalnim kodom i minimalnim naporom, a ima sve prednosti koje se mogu očekivati od modernog web *frameworka*, a to su: izvrsna podrška za obrasce, rukovanje pogreškama, događaje, popise itd.

Upravo zbog svih ovih prednosti i prethodnog iskustva s React-om, on je odabran kao jedan od alata za izradu web aplikacije u ovom diplomskom radu.

### 4.1.2. Express

Idući sloj, odnosno srednji sloj web aplikacije čine Express i Node. Express je *framework* na strani poslužitelja, koji radi unutar Node poslužitelja. Express je predstavljen kao „brz, samostalan, minimalistički web *framework* za Node.js“ i to je najtočnija definicija. Nadopunjuje Node sa slojem rudimentarnih značajki web aplikacije koje pružaju HTTP uslužne metode i funkcionalnost međuprograma (engl. *middleware*). Općenito, funkcionalnost međuprograma u bilo kojoj aplikaciji omogućuje dodavanje različitih komponenti kako bi radile zajedno. U specifičnom kontekstu okvira web aplikacija na strani poslužitelja, međuprogramske funkcije imaju pristup HTTP cjevovodu zahtjev-odgovor (engl. *request-response*), što znači pristup objektima zahtjev-odgovor i također sljedećim međuprogramskim funkcijama u ciklusu web aplikacije. U bilo kojoj web aplikaciji razvijenoj s Nodeom, Express se može koristiti kao API usmjeravanje i međuprogramski *framework*. [21]

Express se općenito u web aplikacijama, a također i web aplikaciji za ekološki uzgoj biljaka koristi za radnje kao što su: rukovanje API usmjeravanjem na strani poslužitelja, posluživanje statičkih datoteka klijentu, ograničavanje pristupa resursima uz integraciju provjere autentifikacije, implementiranje rukovanja pogreškama te dodavanje bilo kojeg međuprogramskog paketa koji će proširiti funkcionalnost web aplikacije prema potrebi.

### 4.1.3. Node

Node je razvijen kao JavaScript *runtime* okruženje. Omogućio je početak korištenja JavaScript-a na poslužiteljskoj strani za izgradnju raznih alata i aplikacija, što nije bilo moguće prethodno zbog ograničenja tih slučajeva isključivo na preglednik. Node ima arhitekturu vođenu događajima koja je sposobna za asinkroni, neblokirajući ulaz/izlaz. Njegov jedinstevni ulaz/izlaz model eliminira čekanje za pristup posluživanju zahtjeva. To omogućuje izradu skalabilne i „lagane“ web aplikacije u stvarnom vremenu koja može učinkovito obraditi mnoge zahtjeve. [22]

Node-ov zadani sustav za upravljanje paketima, zove se engl. *Node Package Manager* ili *npm* i dolazi u paketu s Node instalacijom. On daje pristup velikom broju Node paketa za višekratnu upotrebu koje su izradili programeri diljem svijeta i može se pohvaliti da je trenutno najveći ekosustav paketa otvorenog koda na svijetu. Stoga, velika je podrška i postoje paketi za gotovo sve situacije te je zato odabran Node za razvoj ove web aplikacije. [21]

### 4.1.4. MongoDB

MongoDB je odličan izbor za bazu podataka ako se odluči na NoSQL bazu podataka. Ako aplikacija pohranjuje bilo kakve podatke (korisničke profile, komentare, uploade, događaje itd.), tada je najbolje koristiti bazu podataka s kojom je jednako lako raditi kao i s React-om, Express-om i Node-om. MongoDB je baza podataka orijentirana na dokumente koja sprema podatke u fleksibilne dokumente slične JSON-u. To znači da se polja mogu razlikovati od dokumenta do dokumenta, a modeli podataka se mogu razvijati tijekom razvoja kao odgovor na promjenu zahtjeva web aplikacije. [22]

MongoDB koristi ekspresivni jezik koji omogućuje ad-hoc upite, indeksiranje za brzo pretraživanje i agregaciju u stvarnom vremenu koja pruža moćne načine za pristup i analizu podataka uz održavanje performansi, čak iako veličina podataka raste eksponencijalno. Odabirom MongoDB kao baze podataka uz Node i Express, može se napraviti web aplikacija u potpunosti temeljena na JavaScript-u i samostalna aplikacija na strani poslužitelja. Sve navedeno odgovara zahtjevima web aplikacije za ekološki uzgoj biljaka. [21]

U principu korištenje MongoDB baze podataka funkcionira ovako: JSON dokumenti kreirani u React *frontend* dijelu mogu se poslati na Express poslužitelj, gdje se oni procesiraju te ako su ispravni pohranjuju direktno u MongoDB bazu podataka.

# 5. BAZA PODATAKA I UPRAVLJANJE PODACIMA

Baza podataka je ključni dio svake web aplikacije. Može se definirati kao organizirana kolekcija podataka na način da ti podaci budu što jednostavniji i lakši za pronaći. Koristi se za trajnu i sigurnu pohranu podataka kojima se može pristupiti u svakom trenutku. Pohranjuju se podaci poput korisničkih informacija, podataka o sesiji, informacije o proizvodima i drugih podataka web aplikacije. Glavni ciljevi baze podataka su pohranjivanje, preuzimanje i ažuriranje podataka. Web aplikacija vrši interakciju s bazom podataka tako da pohranjuje podatke i prima podatke, nakon čega se ti podaci mogu koristiti u web aplikaciji.

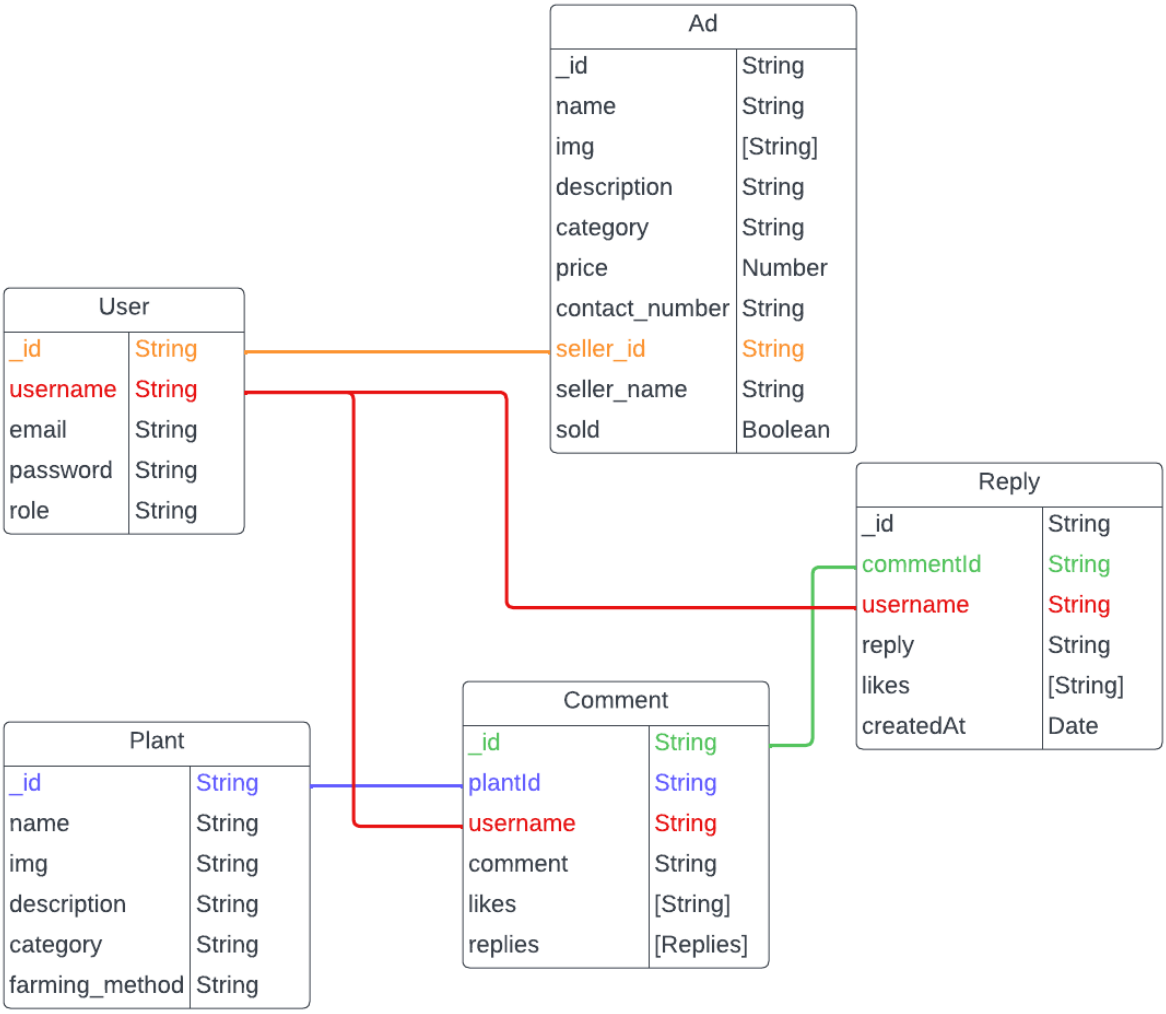
Općenito, postoje dvije vrste baza podataka, a to su relacijske i nerelacijske. Relacijska baza podataka je pohranjivanje podataka na način da se ostvaruju relacije između dva ili više entiteta. Nazivaju se još i SQL (engl. *Structured Query Language*) baze podataka prema jeziku koji se koristi za interakciju. Iako su relacijske baze tradicionalnije i jednostavne za korištenje, zadnjih godina se sve više okreću nerelacijskim bazama podataka. Glavni razlog tome je veliko povećanje obujma podataka, što kod relacijskih dovodi do toga da se gubi učinkovitost i smanjuju performanse. Nerelacijske baze podataka (engl. *Non SQL*) su učinkovitije i u većini slučajeva imaju bolje performanse za velike količine podataka. To su baze podataka koje ne moraju imati relaciju (ali mogu) i orijentirani su na dokumentima koji ne moraju biti povezani na bilo kakav način. Fleksibilnije su od relacijskih i horizontalno su skalabilne. Mogu biti znatno brže od relacijskih, a jednostavnije su i lakše za korištenje. Upravo zbog tih prednosti je odlučeno da se za razvoj ove web aplikacije koristi nerelacijska baza podataka (MongoDB). [23]

## 5.1. Modeliranje baze podataka

Kao što je već objašnjeno, MongoDB je nerelacijska baza podataka. Međutim, za razvoj ove web aplikacije ipak su potrebne određene relacije, odnosno povezanost između određenih entiteta. Korisnik (User) povezan je s oglasima (Ad), komentarima (Comment) i odgovorima (Reply). Za oglase je potreban ID korisnika kako bi se oglas povezao s korisnikom koji je kreirao taj oglas. Time se postiže da se točno zna čiji je oglas i mogu se iskoristiti pojedine informacije o korisniku, kao što je korisničko ime korisnika. Korisnik je povezan s komentarima i odgovorima na način da se korisničko ime (username) koristi u objektima komentara i odgovora. Na taj način se zna koji je korisnik autor komentara ili odgovora. Nadalje, biljke (Plant) su povezane s komentarima pomoću atributa ID biljke. Pošto su svi komentari vezani za određenu biljku, ID povezuje tu biljku i komentare kako bi se znalo na koju biljku se komentar odnosi. Svaki komentar ima odgovore, pa

su komentari i odgovori također povezani pomoću atributa ID komentara. Time se ostvaruje povezanost odgovora s određenim komentarom.

Shema opisane baze podataka i njenih relacija unutar web aplikacije prikazana je slikom 5.1.




**Sl. 5.1.** Shema baze podataka

# 6. FUNKCIONALNOSTI WEB APLIKACIJE

Funkcionalnosti web aplikacije predstavljaju sve implementirane metode koje se koriste za upravljanje podacima na poslužiteljskoj strani, za razliku od klijentske strane gdje se ti podaci koriste i prikazuju. Općenito su to metode koje vrše izravnu interakciju s bazom podataka (upisuju i dohvaćaju razne podatke), vrše razne provjere koje se trebaju provjeravati samo na strani poslužitelja itd. U ovoj web aplikaciji to su konkretno metode vezane za registraciju i prijavu korisnika te razne metode za upravljanje biljkama, oglasima i komunikacijom.

## 6.1. Registracija i prijava korisnika

U web aplikaciji za ekološki uzgoj biljaka potreban je sustav registracije i prijave korisnika zbog oglašavanja i razgovora o biljkama. Poželjno je znati tko oglašava proizvod, a isto tako i s kim se komunicira.

Neke od implementiranih funkcionalnosti sustava registracije i prijave su:

- registracija: korisnici se mogu registrirati kreiranjem novog korisničkog računa koristeći korisničko ime, adresu e-pošte i zaporku
- autentifikacija: nakon registracije, korisnici se mogu prijaviti u svoj profil koristeći korisničko ime ili adresu e-pošte i zaporku te se odjaviti iz web aplikacije
- zaštićene rute: pojedini dijelovi web aplikacije nisu dostupni, ako korisnik nije prijavljen, odnosno ako korisnik nije administrator
- autorizacija: samo administratori mogu dodavati nove biljke u bazu podataka, prijavljeni korisnici mogu dodavati i brisati samo svoje oglase, a gosti mogu samo pregledavati biljke i oglase

Kako bi se definirao svaki korisnik, potrebno je napraviti model korisnika. Svaki korisnički profil prilikom registracije sprema se u MongoDB bazu podataka i obrađuje logiku vezanu za korisnika kao što je enkripcija zaporke i provjera točnosti podataka. Model korisnika (*User*) definiran je tablicom 6.1.

Tablica 6.1. *User* model

| Ime polja | Tip | Opis |
|---|---|---|
| username | String | Obavezno i jedinstveno polje za spremanje korisničkog imena svakog korisnika |
| email | String | Obavezno i jedinstveno polje za spremanje adrese e-pošte korisnika |
| password | String | Obavezno polje za autentifikaciju. Baza podataka sprema enkriptiranu zaporku, a ne dani *string* zbog sigurnosnih razloga. |
| role | String | Obavezno polje koje se automatski dodjeljuje prilikom registracije. Korisnik može imati ulogu običnog korisnika ili administratora. Koristi *enum* [„user“, „admin“], a zadana vrijednost je „user“. |
| createdAt | Date | Automatski generirano vrijeme kada je korisnički račun kreiran |
| updatedAt | Date | Automatski generirano vrijeme kada je korisnički račun ažuriran |

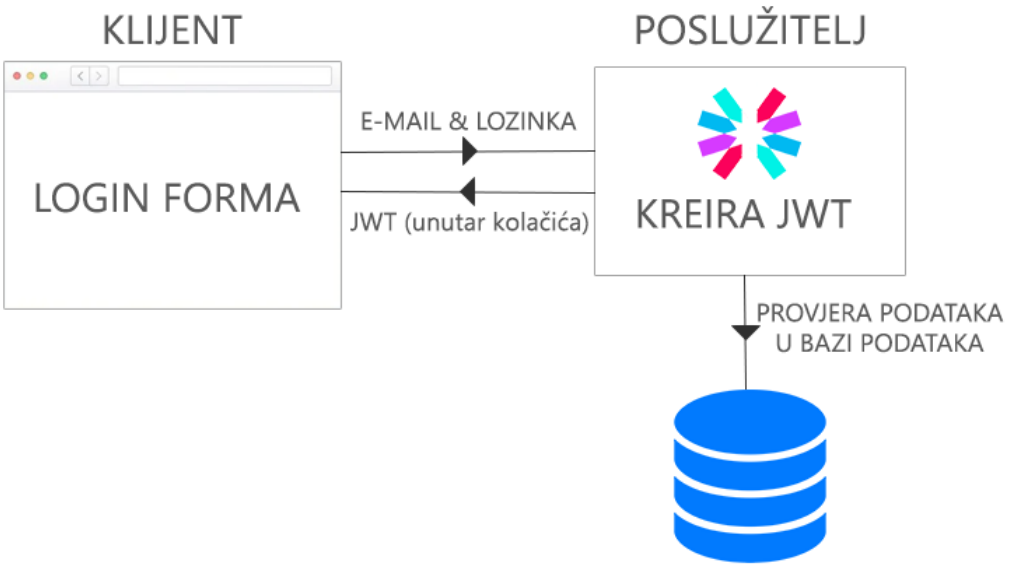
Kako bi se omogućile sve te radnje, na *backendu* je potrebno postaviti rute koje upravljaju tim operacijama. Te rute će se koristiti na *frontend* dijelu aplikacije kako bi se korisnik registrirao ili prijavio u svoj korisnički račun. Rute su definirane tablicom 6.2.

Tablica 6.2. Rute za operacije autentifikacije

| Operacija | API ruta | HTTP metoda |
|---|---|---|
| Kreiranje korisničkog računa | /api/user/register | POST |
| Prijava u korisnički račun | /api/user/login | POST |

Nakon definiranja modela i ruta, potrebno je implementirati funkcionalnost same autentifikacije. To je ostvareno korištenjem JWT (JSON Web Token) mehanizma.

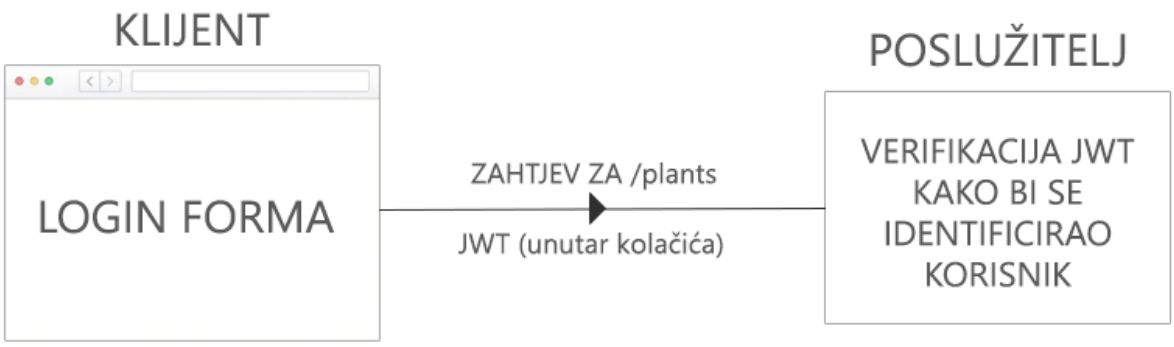
JWT je otvoreni standard (RFC 7519) koji definira kompaktan i samostalan način za siguran prijenos informacija između strana kao JSON objekt. Te informacije se mogu provjeriti i pouzdane su jer su digitalno potpisane. JWT-ovi se mogu potpisati korištenjem tajni (HMAC algoritam) ili pomoću para javnih/privatnih ključeva koristeći RSA ili ECDSA. Također, korištenjem JWT-a nije potrebno pohraniti status korisnika na poslužiteljskoj strani. Rad JWT mehanizma je prikazan na slici 6.1.




**Sl. 6.1.** Princip rada JWT mehanizma prilikom prijave u korisnički račun

Korisnik se prijavljuje upotrebom obrasca na klijentskoj strani pomoću koje se šalje zahtjev na poslužiteljsku stranu zajedno s podacima (adresa e-pošte i zaporka). Poslužitelj zatim provjerava te podatke i uspoređuje ih s onima koji su spremljeni u bazu podataka za tog korisnika. Ako su podaci identični, poslužitelj kreira JWT za korisnika i šalje ga u preglednik gdje se može spremiti lokalno u kolačiće (engl. *cookies*). Na taj način se klijentskoj strani prepušta upravljanje stanjem korisnika. JWT sadrži enkriptirane podatke o tom korisniku kako bi ga identificirao. Dokle god postoji JWT u lokalnoj pohrani kolačićima, korisnik se smatra prijavljenim i autentificiranim.

Kada korisnik ima JWT pohranjen u pregledniku, on ga šalje poslužitelju prilikom svakog zahtjeva (npr. pristup nekoj stranici, API ruti itd.). To omogućava kontrolu pristupa zaštićenim rutama, odnosno autorizaciju. Kada poslužitelj primi JWT od klijenta, on ga može verificirati i dekriptirati kako bi identificirao korisnika. Ukoliko se radi o ispravnom JWT, korisnik se smatra prijavljenim od strane poslužitelja. Prema tome, poslužitelj može „odlučiti“ prikazati zaštićene podatke ili stranice koje zahtijevaju da je korisnik prijavljen. Ako JWT ne postoji ili je neispravan, korisnik nije autentificiran i poslužitelj može vratiti nekakvu pogrešku ili ponovno preusmjeriti na stranicu za prijavu. Taj mehanizam opisan je slikom 6.2.




**Sl. 6.2.** Princip rada JWT mehanizma prilikom zahtjeva zaštićenoj ruti

Prilikom registracije i prijave provjeravaju se moguće pogreške. Prilikom registracije, ukoliko su neka od polja prazna, poslužitelj vraća pogrešku da sva polja moraju biti popunjena. Ta pogreška se prikazuje na klijentskoj strani kada poslužitelj obradi zahtjev. Također se vrši provjera ispravne adrese e-pošte i kompleksne zaporke. Zaporka mora biti kombinacija od najmanje 8 znakova te mora sadržavati najmanje 1 veliko slovo, 1 malo slovo, 1 broj i 1 specijalni znak. Zbog toga što korisničko ime i adresa e-pošte moraju biti jedinstveni, odnosno ne mogu postojati dva korisnika s istim korisničkim imenom ili adresom e-pošte, poslužitelj vraća poruku pogreške da su korisničko ime ili adresa e-pošte već zauzeti. Ukoliko je sve ispravno, poslužitelj enkriptira upisanu zaporku pomoću paketa „bycrypt“ i sprema ju kao takvu u bazu podataka. Ukoliko nema pogrešaka, korisnik se registrira i automatski prijavljuje kreiranjem JWT-a.

Što se tiče prijava korisnika, također se vrši provjera jesu li sva polja ispunjena. Korisnik se može prijaviti ili pomoću korisničkog imena ili pomoću adrese e-pošte te ukoliko ne postoji niti jedno niti drugo, vraća se pogreška da korisničko ime ili adresa e-pošte ne postoji. Ukoliko je korisnik pronađen, uspoređuje se zaporka koju je upisao. Zaporka se enkriptira i uspoređuje s enkriptiranom zaporkom u bazi podataka pomoću funkcije iz paketa „bycrypt“. Ukoliko nema pogrešaka, kreira se JWT i korisnik je prijavljen.

Nakon uspješne prijave, odnosno registracije, preglednik sadrži JWT u svojoj pohrani i onda se na osnovu tog stanja mogu prikazivati ili onemogućiti pristupi nekim stranicama u web aplikaciji, kao i ograničenja nekih funkcionalnosti.

## 6.2. Modeliranje i upravljanje biljkama

Glavni dio web aplikacije, a i sama tema ovog diplomskog su biljke. Stoga, potrebne su određene funkcionalnosti koje upravljaju informacijama o ekološkom uzgoju biljaka. Cilj je korisniku omogućiti pretraživanje baze podataka biljaka te za svaku biljku pružiti detaljne informacije.

Implementirane funkcionalnosti vezane uz biljke su:

- pretraživanje: korisnici mogu pretraživati cijelu bazu podataka biljaka koristeći pretragu pomoću ključne riječi ili filtriranje prema vrstama biljaka
- stranica biljke: odabirom biljke otvara se stranica te biljke koja sadrži sve informacije o toj biljki (slika, opis, metoda ekološkog uzgoja, komentari itd.)
- dodavanje novih biljaka: administratorima je omogućeno dodavanje novih biljaka u bazu podataka. Za svaku biljku potrebno je upisati određene informacije, a podrazumijeva se da vlasnik web aplikacije (administrator) ima točne i dobre informacije

Kako bi se definirala svaka biljka, potrebno je napraviti model biljke. Svaka biljka prilikom dodavanja od strane administratora sprema se u MongoDB bazu podataka. Model biljke (*Plant*) definiran je tablicom 6.3.

Tablica 6.3. *Plant* model

| Ime polja | Tip | Opis |
|---|---|---|
| name | String | Obavezno polje za spremanje imena svake biljke. |
| img | String | Obavezno polje za spremanje slike biljke, odnosno poveznice na sliku. |
| description | String | Obavezno polje za opis biljke. Kratki opis kao općenite informacije o toj biljki i njenim specifičnostima. |
| category | String | Obavezno polje za odabir kategorije, odnosno vrste biljke. |
| farming_method | String | Obavezno polje za unos svih informacija o ekološkom načinu uzgoja pojedine biljke (opis metode na koji se biljka ekološki uzgaja). |
| createdAt | Date | Automatski generirano vrijeme kada je biljka kreirana. |
| updatedAt | Date | Automatski generirano vrijeme kada je biljka ažurirana. |

Rute koje omogućuju sve funkcionalnosti definirane su tablicom 6.4.

Tablica 6.4. Rute za operacije vezane uz biljke

| Operacija | API ruta | HTTP metoda |
|---|---|---|
| Dohvaćanje svih biljaka iz baze podataka | /api/plants/ | GET |
| Dodavanje nove biljke u bazu podataka | /api/plants/ | POST |
| Dohvaćanje pojedine biljke iz baze podataka | /api/plants/:id | GET |
| Brisanje pojedine biljke iz baze podataka | /api/plants/:id | DELETE |

Nakon definiranja modela i ruta, potrebno je definirati metode koje će izvršavati sve navedeno. Metode unutar kontrolera za biljke su: *getPlants, createPlant, getPlant* i *deletePlant*.

Metoda *getPlants* izvršava funkcionalnost dohvaćanja svih biljaka iz baze podataka. Putem API zahtjeva (GET) s *frontend* strane, šalju se parametri pretrage (ključne riječi koje je korisnik upisao) i kategorije, odnosno vrste koje su odabrane. Ukoliko nije upisan nikakav tekst u polje pretrage, pretraga je jednaka praznom *stringu*, odnosno podaci se ne filtriraju po tom parametru. Ukoliko nije odabrana niti jedna vrsta, kategorije su zapravo sve vrste, odnosno nema filtriranja podataka prema vrstama. Nakon što su ti parametri primljeni s *frontend* strane, izvršava se pretraga i filtriranje podataka iz baze podataka. Pronalaze se sve biljke koje sadržavaju ključne riječi pretrage u svom imenu, a zatim se filtriraju ovisno o kategorijama koje su odabrane. Na taj način usporedno rade pretraga po ključnim riječima i po kategorijama. Nakon toga, vraća se objekt *response* koji sadrži objekte svih biljki koje su pronađene te se on koristi na *frontendu* za prikaz podataka.

Metoda *createPlant* izvršava funkcionalnost dodavanja biljke u bazu podataka. Putem API zahtjeva (POST) s *frontend* strane, šalju se sljedeći parametri: ime, slika, opis, vrsta i metoda uzgoja. Prvo se za svaki parametar provjerava je li uopće primljen jer su svi parametri obavezni da se objekt biljke uopće može kreirati. U polje *emptyFields* dodaju se parametri koji nisu primljeni te ukoliko se nešto nalazi u tom polju, znači da korisnik nije predao sve potrebne parametre prilikom dodavanja. Korisniku se vraća poruka pogreške koja ga obavještava da ispuni sva potrebna polja. Međutim, ukoliko su svi parametri uspješno predani, to znači da se biljka može kreirati u bazi podataka. Koristeći predane parametre, u bazi podataka se dodaje nova biljka s tim vrijednostima.

Metoda *getPlant* izvršava funkcionalnost dohvaćanja pojedine biljke iz baze podataka. Putem API zahtjeva (GET) s *frontend* strane šalje se parametar ID biljke. Taj parametar se šalje u samoj poveznici koja se šalje putem API-a. Pošto je ruta koja rukuje ovom funkcionalnosti definirana kao „/api/plants/:id“, na *frontend* strani se šalje API zahtjev tako da se umjesto „:id“ pošalje određeni ID biljke (npr. „/api/plants/64e0b359e94327630cd58366). Taj ID se preuzima iz poveznice i na temelju njega se biljka pretražuje u bazi podataka. Ukoliko biljka nije pronađena korisnik se obavještava da takva biljka ne postoji u bazi podataka, a ukoliko je uspješno pronađena, metoda vraća objekt te biljke, koji se potom može iskoristiti na *frontend* strani.

Metoda *deletePlant* izvršava funkcionalnost brisanja pojedine biljke iz baze podataka. Također, kao i u prethodnoj metodi, šalje se ID unutar poveznice. Ovdje se koristi metoda koja ne samo da pretražuje biljke prema ID-u, nego odmah i briše objekt biljke nakon što ju pronađe. Ukoliko biljka nije pronađena korisnik se obavještava da takva biljka ne postoji u bazi podataka, a ukoliko je uspješno pronađena, metoda briše objekt biljke iz baze podataka.

### 6.2.1. Modeliranje i upravljanje razgovorima

Razgovor i komunikacija u ovoj web aplikaciji zamišljeni su na način da svaka biljka ima svoj chat, odnosno pripadni razgovor u kojem korisnici dijele savjete, iskustva, pitanja i odgovore. Svaki pojedini razgovor pripada samo jednoj biljci, tako da tema svakog razgovora treba biti isključivo ta biljka.

Razgovor se sastoji od komentara i odgovora na komentare. Na jedan komentar moguće je dodati više odgovora, što izravno određuje na koji komentar se odgovori odnose. Ovakav sustav se često koristi i na dobar način usmjerava razgovor korisnika.

Implementirane funkcionalnosti vezane uz razgovor su:

- dodavanje komentara/odgovora: samo prijavljeni korisnici mogu dodavati komentare/ odgovore
- brisanje komentara/odgovora: samo korisnici koji su autori komentara/odgovora mogu ih i obrisati
- dodavanje oznaka sviđanja: prijavljeni korisnici mogu međusobno označiti komentare/odgovore koji im se sviđaju, što na neki način daje korisnost i važnost pojedinom komentaru/odgovoru

Kako bi se definirao svaki komentar, potrebno je napraviti model komentara. Model komentara (*Comment*) definiran je tablicom 6.5.

Tablica 6.5. *Comment* model

| Ime polja | Tip | Opis |
|---|---|---|
| plantId | Schema.Types.ObjectId | Obavezno polje za povezivanje komentara s biljkom kojoj komentar pripada (preko atributa ID biljke). |
| username | String | Obavezno polje za korisničko ime korisnika koji je napisao komentar. |
| comment | String | Obavezno polje za tekst komentara. |
| likes | [String] | Obavezno polje koje predstavlja polje stringova svih korisnika koji su označili komentar da im se sviđa. |
| replies | [Reply] | Obavezno polje koje predstavlja polje svih odgovora koje su korisnici ostavili na tom komentaru. |
| createdAt | Date | Automatski generirano vrijeme kada je komentar kreiran. |
| updatedAt | Date | Automatski generirano vrijeme kada je komentar ažuriran. |

Model odgovora (*Reply*) definiran je tablicom 6.6.

Tablica 6.6. *Reply* model

| Ime polja | Tip | Opis |
|---|---|---|
| commentId | Schema.Types.ObjectId | Obavezno polje za povezivanje odgovora s komentarom kojem pripada (preko atributa ID komentara). |
| username | String | Obavezno polje za korisničko ime korisnika koji je napisao odgovor. |
| reply | String | Obavezno polje za tekst odgovora. |
| likes | [String] | Obavezno polje koje predstavlja polje *stringova* svih korisnika koji su označili odgovor da im se sviđa. |
| createdAt | Date | Automatski generirano vrijeme kada je odgovor kreiran. |

Rute koje omogućuju sve funkcionalnosti definirane su tablicom 6.7.

Tablica 6.7. Rute za operacije vezane uz komentare

| Operacija | API ruta | HTTP metoda |
|---|---|---|
| Dohvaćanje svih komentara za određenu biljku | /api/comments/:plantId/comments | GET |
| Dodavanje novog komentara za određenu biljku | /api/comments/:plantId/createComment | POST |
| Brisanje pojedinog komentara | /api/comments/:commentId | DELETE |
| Dodavanje novog odgovora na određeni komentar | /api/comments/:commentId/reply | PATCH |
| Dodavanje oznake sviđanja na komentar | /api/comments/:commentId/like | PATCH |
| Brisanje oznake sviđanja na komentar | /api/comments/:commentId/unlike | PATCH |
| Dodavanje oznake sviđanja na odgovor | /api/comments /:commentId/replies/:replyId/like | PATCH |
| Brisanje oznake sviđanja na odgovor | /api/comments /:commentId/replies/:replyId/unlike | PATCH |

Nakon definiranja modela i ruta, potrebno je definirati metode koje će izvršavati sve navedene funkcionalnosti. Metode unutar kontrolera za biljke su: *getAllComments*, *createComment*, *deleteComment*, *addLike*, *deleteLike*, *addReply*, *deleteReply*, *likeReply* i *unlikeReply*.

Metoda *getAllComments* izvršava funkcionalnost dohvaćanja svih komentara za određenu biljku iz baze podataka. Putem API zahtjeva (GET) s *frontend* strane šalje se parametar ID biljke. Taj parametar se šalje u samoj poveznici koja se šalje putem API-a. Ruta koja rukuje ovom funkcionalnosti definirana je kao „/api/plants/:plantId/comments“, pa se na *frontend* strani šalje API zahtjev tako da se umjesto „:plantId“ pošalje određeni ID biljke (npr. „/api/plants/64e0b359e94327630cd58366/comments). Taj ID se preuzima iz poveznice i na temelju njega se pronalaze svi komentari koji su vezani za tu biljku. Nakon što su komentari pronađeni, sortiraju se od najnovijeg prema najstarijem. Metoda vraća polje objekata komentara, koji se potom mogu iskoristiti na *frontend* strani.

Metoda *createComment* izvršava funkcionalnost dodavanja novog komentara u bazu podataka. Putem API zahtjeva (POST) s *frontend* strane, šalju se sljedeći parametri: korisničko ime i tekst komentara. Također, u samoj poveznici se šalje ID biljke uz koju su komentari vezani. Ako su svi

parametri primljeni, kreira se novi objekt komentara s tim parametrima. Ukoliko je slučajno predan krivi ID biljke, vraća se poruka pogreške koja obavještava da ta biljka ne postoji.

Metoda *deleteComment* izvršava funkcionalnost brisanja postojećeg komentara iz baze podataka. Također, kao i u dosadašnjim metodama, šalje se ID unutar poveznice. Ovdje se koristi metoda koja ne samo da pretražuje komentare prema ID-u, nego odmah i briše objekt komentara nakon što ga pronađe u bazi podataka. Ukoliko komentar nije pronađen, korisnik se obavještava da takav komentar ne postoji u bazi podataka, a ukoliko je uspješno pronađen, metoda briše taj komentar.

Metoda *addReply* izvršava funkcionalnost dodavanja odgovora na određeni komentar u bazi podataka. Putem API zahtjeva (PATCH) s *frontend* strane, šalju se sljedeći parametri: korisničko ime i tekst odgovora. Također, šalje se i ID komentara unutar poveznice uz koji je odgovor vezan. Zbog toga što svaki objekt komentara sadrži polje odgovora, ova metoda radi na način da ažurira objekt komentara tako što u to polje (replies) dodaje novi *Reply* objekt. Pomoću atributa ID komentara pronalazi taj komentar, zatim dodaje novi odgovor *push* metodom u to polje.

Metoda *deleteReply* izvršava funkcionalnost brisanja odgovora na određeni komentar u bazi podataka. Funkcionira na sličan način kao i prethodna funkcija za dodavanje novog odgovora, osim što se u ovoj metodi koristi *pull* metoda koja briše odgovor iz polja odgovora (replies). Također, unutar poveznice se šalje i ID odgovora, kako bi se znalo koji točno odgovor treba biti obrisan, a ostali parametri nisu potrebni jer se radi samo o brisanju.

Metoda *addLike* izvršava funkcionalnost dodavanja oznake sviđanja na komentar. Putem API zahtjeva (PATCH) s *frontend* strane šalju se parametri: korisničko ime korisnika i ID komentara (unutar poveznice). Prvo se pronalazi komentar pomoću primljenog atributa ID. Ako je komentar pronađen, vrši se provjera nalazi li se korisničko ime u polju „likes“ (polje koje sadržava sve korisnike koji su označili komentar da im se sviđa). Ako je korisnik već označio komentar da mu se sviđa, ne može ga još jednom označiti, tako da se nova oznaka sviđanja dodaje samo ako korisničko ime nije pronađeno u tom polju.

Metoda *deleteLike* izvršava funkcionalnost brisanja oznake sviđanja na komentar. Cijela funkcionalnost je identična kao prethodna metoda, osim što umjesto dodavanja korisničkog imena u polje "likes“, briše korisničko ime ukoliko se nalazi u tom polju.

Metoda *likeReply* izvršava funkcionalnost dodavanja oznake sviđanja na odgovor. Osim što prima korisničko ime i ID komentara, također prima i ID odgovora. Pomoću tih ID-ova pronalazi se specifičan komentar, a zatim u tom komentaru pronalazi određeni odgovor unutar „replies“ polja.

Nakon što je pronađen određeni odgovor, vrši se provjera nalazi li se korisničko ime u polju „likes“. Ukoliko se ne nalazi, korisničko ime se dodaje u to polje.

Metoda *unlikeReply* izvršava funkcionalnost brisanja oznake sviđanja na odgovor. Funkcionira identično kao prethodna metoda, osim što umjesto dodavanja oznake sviđanja na odgovor, provjerava nalazi li se korisnik u polju „likes“ te ukoliko se nalazi, briše ga iz tog polja.

## 6.3. Modeliranje i upravljanje oglasima

Oglasi u web aplikaciji doprinose tome da korisnici mogu kupovati ili nuditi razne proizvode koji se koriste u svrhu ekološkog uzgoja. Oni su korisni i za potrošače i za prodavače, stoga potrebne su funkcionalnosti za obe strane. Prodavači mogu prodavati svoje proizvode i upravljati sa svojim oglasima, dok potrošači mogu pretraživati i informirati se o proizvodima koji su ponuđeni.

Implementirane funkcionalnosti vezane uz oglase su:

- pretraživanje: korisnici mogu pretraživati cijelu bazu podataka oglasa koristeći pretragu pomoću ključnih riječi ili filtriranje prema kategorijama proizvoda
- stranica oglasa: odabirom oglasa otvara se stranica koja sadrži sve informacije o tom oglasu (slike, tekst oglasa, podaci o prodavaču, cijena itd.)
- dodavanje novih oglasa: svim registriranim korisnicima je omogućeno dodavanje novih oglasa u bazu podataka
- upravljanje oglasima: svaki registrirani korisnik može označiti svoj oglas kao prodan, naknadno ga urediti nakon objave ili ga obrisati

Kako bi se definirao svaki oglas, potrebno je napraviti model oglasa. Model oglasa (*Ad*) opisan je tablicom 6.8.

Tablica 6.8. *Ad* model

| Ime polja | Tip | Opis |
|---|---|---|
| name | String | Obavezno polje koje predstavlja naslov oglasa. |
| img | [String] | Obavezno polje za spremanje svih slika oglasa, odnosno poveznica na slike. |
| description | String | Obavezno polje za tekst oglasa. |
| category | String | Obavezno polje koje predstavlja kategoriju kojoj oglas pripada. |
| price | Number | Obavezno polje koje predstavlja cijenu oglasa. |
| contact_number | String | Obavezno polje koje predstavlja kontakt broj prodavača. |
| seller_id | String | Obavezno polje koje predstavlja ID prodavača. |
| seller_name | String | Obavezno polje koje predstavlja korisničko ime prodavača. |
| sold | Boolean | Obavezno polje koje služi za označavanje oglasa prodanim ili dostupnim. |
| createdAt | Date | Automatski generirano vrijeme kada je oglas kreiran. |
| updatedAt | Date | Automatski generirano vrijeme kada je oglas ažuriran. |

Rute koje omogućuju sve funkcionalnosti definirane su tablicom 6.9.

Tablica 6.9. Rute za operacije vezane uz oglase

| Operacija | API ruta | HTTP metoda |
|---|---|---|
| Dohvaćanje svih oglasa iz baze podataka | /api/ads/ | GET |
| Dohvaćanje korisnikovih oglasa iz baze podataka | /api/ads/my | GET |
| Dohvađanje pojedinog oglasa iz baze podataka | /api/ads/:id | GET |
| Dodavanje novog oglasa u bazu podataka | /api/ads/ | POST |
| Ažuriranje pojedinog oglasa u bazi podataka | /api/ads/:id | PATCH |
| Brisanje pojedinog oglasa iz baze podataka | /api/ads/:id | DELETE |

Nakon definiranja modela i ruta, potrebno je definirati metode koje će izvršavati sve navedene funkcionalnosti. Metode unutar kontrolera za oglase su: *getAds*, *getAd*, *getUserAds*, *createAd*, *updateAd* i *DeleteAd*.

Metoda *getAds* izvršava funkcionalnost dohvaćanja svih oglasa iz baze podataka. Putem API zahtjeva (GET) s *frontend* strane, šalju se parametri pretrage (ključne riječi koje je korisnik upisao) i kategorije, odnosno vrste oglasa koje su odabrane. Ukoliko nije upisan nikakav tekst u polje pretrage, pretraga je jednaka praznom *stringu*, odnosno podaci se ne filtriraju po tom parametru. Ukoliko nije odabrana niti jedna kategorija, tada nema filtriranja podataka prema kategorijama. Nakon što su ti parametri primljeni s *frontend* strane, izvršava se pretraga i filtriranje podataka iz baze podataka. Pronalaze se svi oglasi koji sadržavaju ključne riječi pretrage u svom naslovu, a zatim se filtriraju ovisno o kategorijama koje su odabrane. Na taj način, kao i kod pretrage biljaka, usporedno rade pretraga po ključnim riječima i po kategorijama. Nakon toga, vraća se objekt *response* koji sadrži objekte svih oglasa koji su pronađeni te se on koristi na *frontendu* za prikaz podataka.

Metoda *getAd* izvršava funkcionalnost dohvaćanja pojedinog oglasa iz baze podataka. Putem API zahtjeva (GET) s *frontend* strane šalje se parametar ID oglasa (unutar same poveznice). Pošto je ruta koja rukuje ovom funkcionalnosti definirana kao „/api/ads/:id“, na *frontend* strani se šalje API zahtjev tako da se umjesto „:id“ pošalje određeni ID oglasa (npr.

„/api/ads/64e0b359e94327630cd58366). Taj ID se preuzima iz poveznice i na temelju njega se oglas pretražuje u bazi podataka. Ukoliko oglas nije pronađen, korisnik se obavještava da takav oglas ne postoji u bazi podataka, a ukoliko je uspješno pronađen, metoda vraća objekt tog oglasa, koji se potom može iskoristiti na *frontend* strani.

Metoda *getUserAds* izvršava funkcionalnost dohvaćanja svih oglasa određenog korisnika iz baze podataka. Putem API zahtjeva (GET) s *frontend* strane, šalje se parametar kategorije oglasa, odnosno stanje oglasa (Svi oglasi, Aktivni oglasi ili Prodani oglasi) i ID korisnika. Na temelju atributa ID korisnika pronalaze se svi oglasi u kojima je prodavač (seller_id) taj korisnik. Zatim, ovisno o odabranom parametru kategorije, filtrira oglase tog korisnika. Ukoliko je primljen parametar „Svi oglasi“, tada metoda vraća polje svih oglasa koje je korisnik objavio. Ukoliko je primljen parametar „Aktivni oglasi“, tada metoda vraća sve oglase koje je korisnik objavio, a još uvijek nisu prodani (sold = false). Treći slučaj, ukoliko je primljen parametar „Prodani oglasi“, tada metoda vraća sve oglase koje je korisnik objavio, a označeni su kao prodani (sold = true).

Metoda *createAd* izvršava funkcionalnost dodavanja oglasa u bazu podataka. Putem API zahtjeva (POST) s *frontend* strane, šalju se sljedeći parametri: ime, slike, opis, kategorija, cijena i kontakt broj. Osim tih parametara, šalje se i parametar koji je jednak ID-u korisnika, kako bi se znalo tko je prodavač tog oglasa. Prvo se za svaki parametar provjerava je li uopće primljen jer su svi parametri obavezni da se objekt oglasa uopće može kreirati. U polje *emptyFields* dodaju se parametri koji nisu primljeni te ukoliko se nešto nalazi u tom polju, znači da korisnik nije predao sve potrebne parametre prilikom dodavanja. Korisniku se vraća poruka pogreške koja ga obavještava da ispuni sva potrebna polja. Međutim, ukoliko su svi parametri uspješno predani, to znači da se oglas može kreirati u bazi podataka. Koristeći predane parametre, u bazi podataka se dodaje novi oglas s tim vrijednostima.

Metoda *updateAd* izvršava funkcionalnost uređivanja oglasa koji se već nalazi u bazi podataka. Putem API zahtjeva (PATCH) s *frontend* strane, šalju se svi parametri koji su potrebni i za kreiranje oglasa, jer se ti isti parametri i uređuju, odnosno šalju se njihove ažurirane vrijednosti. Osim tih parametara, šalje se i ID oglasa koji se uređuje. Ukoliko je pronađen oglas s tim ID-em, ažurira se oglas s novim vrijednostima parametara.

Metoda *deleteAd* izvršava funkcionalnost brisanja pojedinog oglasa iz baze podataka. Šalje se ID oglasa (unutar poveznice) na temelju kojeg se pronalazi oglas u bazi podataka. Ovdje se koristi metoda koja ne samo da pretražuje oglase prema ID-u, nego odmah i briše objekt oglasa nakon što ga pronađe u bazi podataka. Ukoliko oglas nije pronađen, korisnik se obavještava da takav oglas

ne postoji u bazi podataka, a ukoliko je uspješno pronađen, metoda briše objekt oglasa iz baze podataka.

## 6.4. Međuslojevi (engl. *middlewares*)

Za kontrolu pristupa određenim funkcionalnostima koriste se tzv. međuslojevi (engl. *middlewares*). Oni omogućuju filtriranje HTTP zahtjeva. Obično služe za provjeru nekih uvjeta/parametara te onda na temelju njih dopuštaju ili zabranjuju pristup nekoj funkcionalnosti. U ovoj web aplikaciji implementirano je 5 međuslojeva, a to su: *requireAuth*, *requireAdmin*, *checkSeller*, *checkAuthorComment* i *checkAuthorReply*.

Međusloj *requireAuth* filtrira HTTP zahtjeve ovisno o tome je li korisnik prijavljen, odnosno služi za posluživanje prijavljenih korisnika, a zabranjuje pristup neprijavljenima. Metoda prima JWT token na osnovu kojeg vrši verifikaciju i pokušava pronaći korisnika u bazi podataka. Ukoliko je verifikacija uspješna i korisnik je pronađen, tada metoda dopušta pristup HTTP zahtjevima. Ukoliko nastane nekakva pogreška, vraća se poruka pogreške da je pristup nedopušten i HTTP zahtjev se ne može izvršiti. Ovaj međusloj se koristi za zahtjeve poput upravljanja oglasima i pisanja komentara, jer za korištenje tih funkcionalnosti korisnik mora biti prijavljen.

Međusloj *requireAdmin* filtrira HTTP zahtjeve ovisno o tome je li prijavljeni korisnik administrator, odnosno služi za posluživanje administratora, a zabranjuje pristup svim ostalim korisnicima. Također, metoda prima JWT token i vrši verifikaciju. Nakon toga pronalazi korisnika u bazi podataka i provjerava njegov atribut uloge (role). Samo ukoliko je taj atribut jednak „admin“, tada se dopušta pristup HTTP zahtjevu. Ovaj međusloj se koristi za zahtjeve vezane uz dodavanje i brisanje biljki iz baze podataka.

Međusloj *checkSeller* filtrira HTTP zahtjeve ovisno o tome je li prijavljeni korisnik prodavač pojedinog oglasa. Također, prvo prima JWT token i vrši verifikaciju. Zatim pronalazi tog korisnika u bazi podataka. Osim JWT tokena, metoda prima i parametar ID oglasa (unutar poveznice). Na temelju tog parametra pronalazi određeni oglas te provjerava je li parametar „seller_id“ jednak ID-u korisnika koji šalje HTTP zahtjev. Ukoliko se podudara, to znači da je prijavljeni korisnik ujedno i prodavač oglasa te mu se dopušta pristup zahtjevu. Ukoliko dođe do pogrešaka, metoda vraća sve poruke pogreške kako bi korisnik znao točnu grešku. Ovaj međusloj se koristi za zahtjeve vezane za oglase. Da bi korisnik obrisao ili uredio oglas, potrebno je da on bude prodavač, odnosno autor tog oglasa.

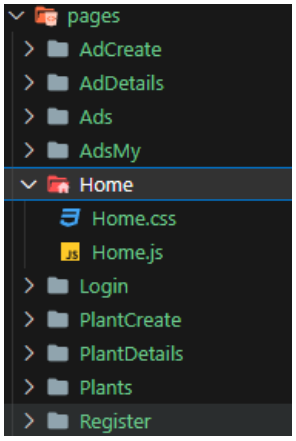
Međuslojevi *checkAuthorComment* i *checkAuthorReply* filtriraju HTTP zahtjeve ovisno o tome je li prijavljeni korisnik autor napisanog komentara/odgovora. Funkcioniraju na identičan način, osim što se kod *checkAuthorComment* koristi ID komentara, a kod *checkAuthorReply* ID odgovora. Naravno, kao i kod svih metoda, prima se JWT token i vrši verifikacija. Zatim se pronalazi određeni korisnik. Na temelju atributa ID komentara/odgovora, pronalazi se određeni komentar/odgovor. Nakon toga vrši se usporedba atributa korisničkog imena (username). Ako je korisničko ime prijavljenog korisnika jednako korisničkom imenu u komentaru/odgovoru, to znači da je prijavljeni korisnik ujedno i autor tog komentara/odgovora. Dopušta mu se pristup HTTP zahtjevima, a ukoliko nastane bilo kakva greška, metoda vraća poruku pogreške. Ovaj međusloj koristi se za funkcionalnost brisanja komentara i odgovora. Svaki korisnik može brisati samo svoje komentare i odgovore.

# 7. DIZAJN KORISNIČKOG SUČELJA

Korisničko sučelje je izrađeno pomoću React biblioteke za korisnička sučelja. Dizajn ove web aplikacije je sličan kao dizajn većine modernih web aplikacija. Web aplikacija za ekološki uzgoj biljaka se sastoji od 3 glavna dijela: navigacijska traka, stranica i podnožje (engl. *footer*).

Pri vrhu je navigacijska traka pomoću koje se korisnik „kreće“ po web aplikaciji. Ona omogućava pristup različitim stranicama, odnosno rutama unutar web aplikacije. Može se reći da je navigacijska traka izbornik koji omogućava povezivanje internih web stranica.

Ispod navigacijske trake nalazi se prikaz trenutne web stranice. Za svaku rutu, odnosno link unutar web aplikacije napravljena je pripadajuća web stranica. Svaka web stranica sastoji se od jedne ili više komponenti jer je React temeljen na komponentama. Može se opisati kao da je svaka stranica „slagalica“ komponenti. Na *frontend* dijelu aplikacije stranice su smještene u mapu „pages“. Svaka stranica ima svoju mapu koja se sastoji od React komponente i pripadne CSS datoteke za tu stranicu. Svaka stranica ima različite komponente, pa zato svaka stranica ima različitu CSS datoteku. Struktura mape i datoteka za stranice je prikazana slikom 7.1.




**Sl. 7.1.** Struktura mape „pages“

Pri dnu web aplikacije nalazi se tzv. *footer*. Ukoliko se korisnik nije uspio snaći kroz navigacijsku traku ili nije uspio pronaći neke informacije, obično se one mogu naći ovdje. Prema tome, za svaku web aplikaciju je važan i *footer* jer omogućava korisnicima sve informacije na jednom mjestu, kao i neke dodatne. Najčešće se sastoji od obavijesti o autorskim pravima, poveznice na politiku privatnosti, karte web-mjesta, ikona društvenih mreža, podataka za kontakt itd. U principu, sadrži informacije koje poboljšavaju sveukupnu upotrebljivost web aplikacije.

## 7.1. Responzivni web dizajn

U današnje vrijeme, ljudi u velikom broju koriste pametne telefone i tablete za pristup internetu, a ne samo stolna i prijenosna računala. Stoga, web aplikacije bi trebale biti optimizirane za sve te uređaje kako bi pružile svim korisnicima na različitim uređajima najbolje iskustvo. Kao odgovor na tu potrebu pojavljuje se responzivni web dizajn.

Responzivni web dizajn daje web aplikaciji fleksibilnost da se prilagodi bilo kojem uređaju tj. njihovim razlučivostima zaslona. Osim različitih veličina zaslona i razlučivosti, različitih preglednika i platformi, postoje i neke razlike u načinu na koji korisnici interaktiraju sa svojim uređajima; pomoću miša, dodirivanja zaslona ili pokreta. [24]

Pojam „responzivni dizajn“ je prvotno korišten i objašnjen od strane web dizajnera imena Ethan Marcotte 2010. godine. „Responzivni web dizajn je pristup koji podrazumijeva da bi dizajn i okruženje korisnika trebali odgovarati zahtjevima na temelju veličine zaslona, platforme i orijentacije zaslona.“ [24, str.14]

Responzivni web dizajn implicira na različit način razmišljanja npr. mala promjena u filozofiji web dizajna. Reponzivna web stranica ima jedan URL, jedan HTML kod i njen sadržaj se prikazuje prema definiranim CSS medijskim upitima (engl. *media queries*) na više različitih uređaja (stolna računala, prijenosna računala, pametni telefoni, tableti i televizori). Automatski će skalirati i prilagoditi sadržaj ovisno o uređaju. Cilj je da se postigne što bolja čitljivost i navigacija na bilo kojem uređaju uz minimalnu promjenu veličine i pomicanja stranice. Princip responzivnog web dizajna prikazan je slikom 7.2. Na slici se može vidjeti razlika u rasporedu sadržaja, ovisno o vrsti uređaja i veličini zaslona. [25]

**Sl. 7.2.** Princip responzivnog web dizajna. Preuzeto iz izvora [26]

Prema tome, glavni ciljevi responzivnog dizajna su:

- prilagodba prikaza tako da odgovara različitim veličinama zaslona

- promjena veličina slika kako bi odgovarale razlučivosti zaslona
- posluživanje slika manje propusnosti na mobilne uređaje
- pojednostavljenje elemenata stranice za mobilno korištenje
- skrivanje neobaveznih/nebitnih elemenata na manjim zaslonima
- pružanje većih poveznica i gumbova za mobilne korisnike koji se mogu lakše pritisnuti prstom
- detektiranje i reagiranje na mobilne značajke kao što su geo-lokacija i orijentacija uređaja [25, str.6]

Tri glavna elementa responzivnog dizajna su:

- fluidni raspored koji koristi fleksibilnu mrežu, što zauzvrat osigurava da se web stranica može skalirati na punu širinu preglednika
- slike koje rade u fleksibilnom kontekstu, bilo da su same fluidne ili kontrolirane mehanizmima prelijevanja (engl. *overflow*)
- medijski upiti koji optimiziraju dizajn za različite kontekste gledanja i ispravljaju pogreške koje se javljaju pri različitim rasponima razlučivosti [27, str.1207-1208]

Za izradu web aplikacije s fluidnim rasporedom, dizajner specificira širinu područja sadržaja kao postotak prozora preglednika. Kod fiksnog rasporeda, širina je statična i dana je u pikselima, što znači da se ne prilagođava veličini zaslona preglednika, a to može ostaviti prazne prostore na zaslonima visoke rezolucije i poremetiti cijeli sadržaj. Fluidni rasporedi su dinamični i osjetljivi na korisnike, odnosno prilagođavaju se raspoloživom prostoru na korisničkom sučelju i pružaju povećanu dostupnost sadržaja. [27]

Rasporedi temeljeni na postocima odlično mijenjaju veličinu prema veličini preglednika koji ih prikazuje. Međutim, problematično je osigurati da se i sadržaj unutar stranice istovremeno mijenja. Iako će se tekst prelamati prema širini svog kontejnera, medijski objekti (npr. slike ili videozapisi) imaju postavljenu zadanu veličinu. Postavljanje medijskog objekta unutar rasporeda fluidne širine spriječit će da se to područje ikada smanji ispod širine medijskog objekta. CSS rješava ovaj problem korištenjem svojstva „max-width“. Skaliranje medijskog objekta prema njegovom nadređenom kontejneru jednostavno se može postići korištenjem ovog svojstva koje osigurava da preglednik skalira objekt na njegovu veličinu kontejnera, ako širina kontejnera padne ispod izvorne veličine objekta. [28]

Nažalost, fluidni rasporedi dolaze i s određenim problemima prilikom korištenja. Na primjer, ukoliko se neka stranica sastoji od 2 stupca sadržaja. Lijevi sadržaj, koji je dizajniran kao bočna

navigacija, ima širinu od 20%, što je u redu za veće zaslone. Međutim, na mobilnom telefonu, koji ima zaslon širine otprilike 320px, taj stupac je širine 64px. Riječi unutar te bočne navigacije će vjerojatno biti presječene u dva retka, što će umanjiti korisničko iskustvo čitanja bočne navigacije. CSS medijski upiti rješavaju te probleme dopuštajući preglednicima posluživanje različitih stilova za različite kontekste gledanja. Oni omogućuju izgradnju više različitih rasporeda koristeći isti HTML dokument. Predstavljaju uvjetne izjave koje mogu identificirati ne samo vrste medija (zaslon), nego i fizičke karakteristike uređaja i preglednika (npr. širina preglednika, orijentacija i sl.). [28]

Prema izvoru [24, str.16], mogu se definirati određene granice za medijske upite. Najčešće korištene granice između veličina zaslona ovisno o uređaju prikazane su tablicom 7.1.

Tablica 7.1. Primjer granica za medijske upite. Izradio autor prema izvoru [24, str.16]

| **Vrsta uređaja** | **Širina razlučivosti ili orijentacija** |
|---|---|
| Pametni telefon | Manje od 570px |
| Tablet | Podržava orijentaciju |
| Računalo s manjim zaslonom | 570px – 1280px |
| Monitor širokog zaslona | Veće od 1280px |

## 7.2. Navigacijska traka

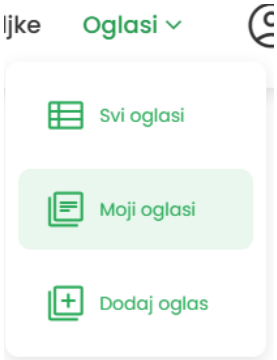
Navigacijska traka web aplikacije mijenja se ovisno o stanju korisnika i ulozi korisnika. Korisnik može imati dva stanja: neprijavljen (gost) i prijavljen. Ukoliko je korisnik prijavljen, navigacijska traka mu nudi posebne opcije, ovisno o tome je li običan korisnik ili administrator. Dakle, postoje 3 različita dizajna navigacijske trake: kada je korisnik gost, kada je korisnik prijavljen te kada je korisnik prijavljen i ima ulogu administratora.

U slučaju da je korisnik neprijavljen (gost), nudi mu se najmanje opcija. Osim poveznica za prijavu i registraciju korisnika, postoje poveznice za naslovnicu, pregled svih oglasa i pregled svih biljaka. Budući da korisnik koji nije prijavljen ne može postaviti oglas, nudi mu se samo opcija da pregledava sve oglase. Navigacijska traka u slučaju kada korisnik nije prijavljen, prikazana je slikom 7.3.

**Sl. 7.3.** Navigacijska traka kada je korisnik neprijavljen

U slučaju kada je korisnik prijavljen, glavne opcije ostaju, ali poveznice za registraciju i prijavu korisnika se zamjenjuju padajućim izbornikom za upravljanje korisničkim računom. Pritiskom na ikonu profila otvara se padajući izbornik koji sadrži korisničko ime korisnika i gumb za odjavu korisnika. Prijavljeni korisnik ima dopuštenje za postavljanje oglasa, pa tako običnu poveznicu za pregled oglasa zamjenjuje padajući izbornik s 3 opcije: Svi oglasi, Moji oglasi i Dodaj oglas. Navigacijska traka u slučaju kada je korisnik prijavljen prikazana je slikom 7.4, a na slici 7.5. prikazan je primjer dizajna padajućeg izbornika.

**Sl. 7.4.** Navigacijska traka kada je korisnik prijavljen




**Sl. 7.5.** Padajući izbornik za oglase

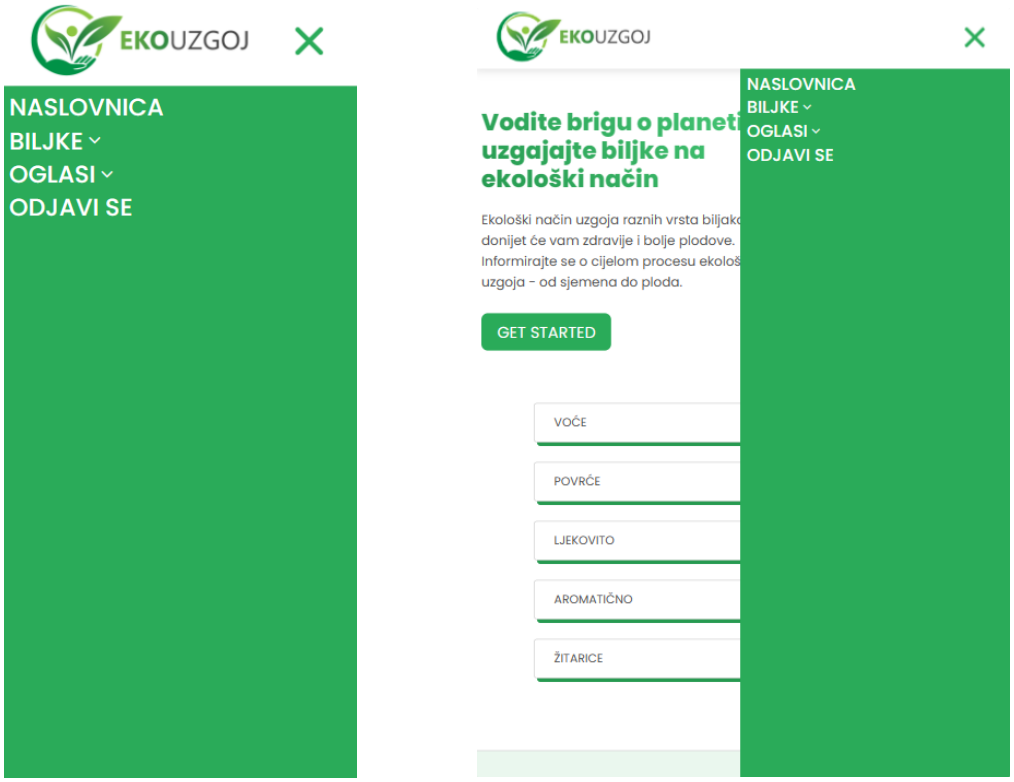
U slučaju kada je korisnik prijavljen i ima ulogu administratora, navigacijska traka se nadograđuje u odnosu na prijavljenog korisnika. Administrator ima dopuštenje za dodavanje novih biljaka u bazu podataka, pa se tako obična poveznica za pregled biljaka zamjenjuje padajućim izbornikom koji nudi 2 opcije: Pregledaj i Dodaj novu biljku. Navigacijska traka u slučaju kada je korisnik prijavljen i ima ulogu administratora prikazana je na slici 7.6.

**Sl. 7.6.** Navigacijska traka kada korisnik ima ulogu administratora

Responzivni dizajn navigacijsku traku izmjenjuje na način da poveznice (Naslovnica, Biljke, Oglasi i Korisnik) nisu više vidljive, već je umjesto njih vidljiva ikona izbornika (tzv. hamburger menu). Klikom na tu ikonu otvara se izbornik koji nudi iste opcije kao što su i poveznice na većem

zaslonu. Dok je izbornik otvoren, onemogućeno je pomicanje u web aplikaciji sve dok se ne odabere neka poveznica ili ne zatvori izbornik ponovnim klikom na gumb.

Izgledi navigacijske trake na mobilnim uređajima i tabletima prikazani su na slici 7.7.



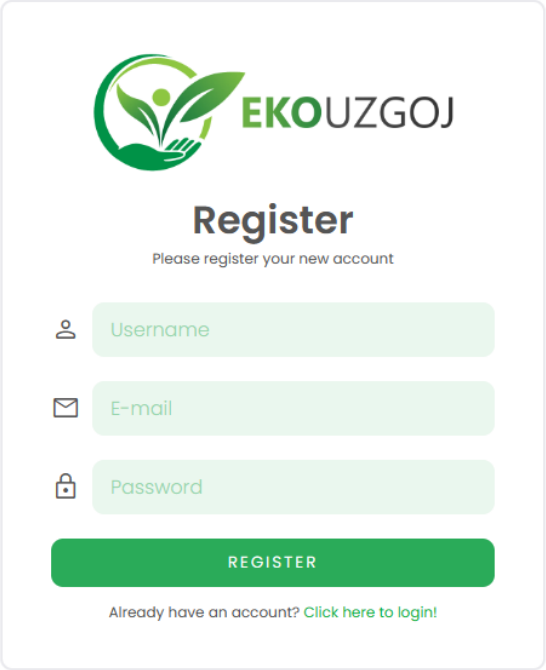

**Sl. 7.7.** Navigacijska traka na mobilnim uređajima (lijevo) i tabletima (desno)

## 7.3. Stranice za registraciju i prijavu korisnika

Glavni dio stranica za registraciju i prijavu korisnika je forma. U formu se upisuju korisnički podaci i podnošenjem forme se ti upisani podaci šalju putem API-a na poslužitelj, gdje se vrši proces autentifikacije korisnika.

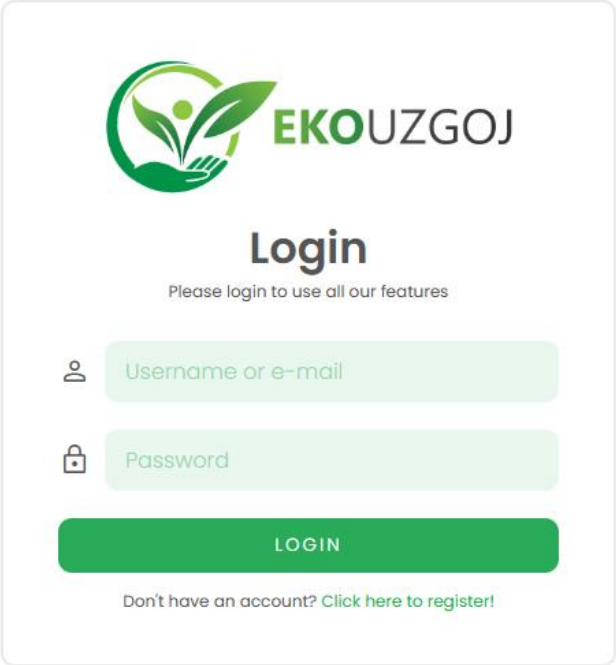
Stranica za registraciju korisnika sadrži samo formu koja ima tri polja: korisničko ime, adresa e-pošte i zaporka. Pritiskom na gumb za registraciju se podnosi forma. Ukoliko postoje pogreške, poslužitelj neće kreirati korisnika u bazi podataka te će vratiti i prikazati poruku pogreške unutar forme crvenom bojom. Dakle, poslužitelj rukuje s pogreškama, dok ih klijentski dio samo prikazuje. Ukoliko je registracija uspješna, odnosno odgovor poslužitelja je prošao bez pogrešaka, automatski se prijavljuje korisnik. To funkcionira tako da se korisnik sprema u lokalnu pohranu preglednika i na taj način se rukuje autentifikacijom na klijentskoj strani. Ispod gumba za

registraciju nalazi se poveznica koja upućuje korisnika na stranicu za prijavu, ukoliko već ima korisnički račun. Forma za registraciju korisnika prikazana je na slici 7.8.




**Sl. 7.8.** Forma za registraciju korisnika

Stranica za prijavu korisnika sadrži formu koja ima dva polja: korisničko ime ili adresa e-pošte i zaporka. Zbog toga što se prijava može izvršiti ili pomoću korisničkog imena ili pomoću adrese e-pošte, koristi se samo jedno polje za ta dva podatka. Kao i kod forme za registraciju korisnika, pritiskom na gumb podnosi se forma, odnosno šalju se korisnički podaci na poslužitelj. Ukoliko postoje poruke pogreške, također će biti ispisane unutar forme crvenom bojom. Ukoliko je prijava uspješna, kao i kod registracije, korisnik se sprema u lokalnu pohranu preglednika. Ispod gumba za prijavu nalazi se poveznica koja vodi na stranicu za registraciju, ukoliko korisnik nema kreiran korisnički račun. Forma za prijavu korisnika prikazana je na slici 7.9.

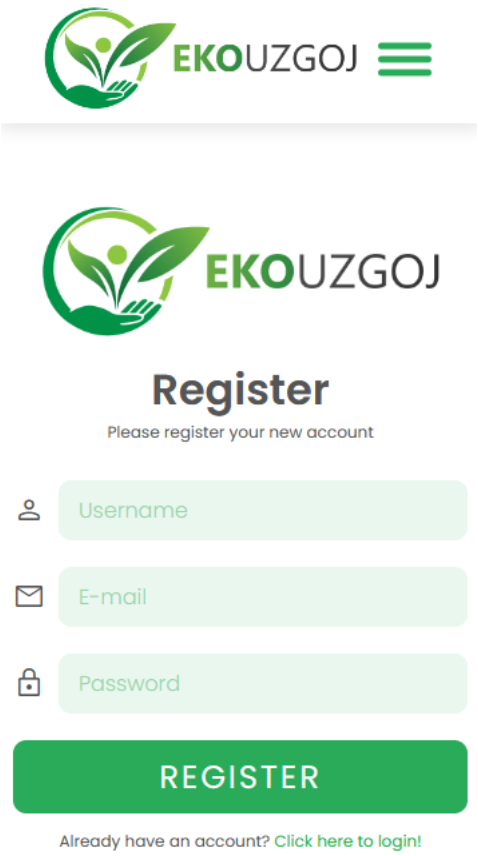



**Sl. 7.9.** Forma za prijavu korisnika

Nakon uspješne registracije ili prijave, korisnik se preusmjerava na naslovnu stranicu te se mijenja izgled navigacijske trake (ovisno o tome je li običan korisnik ili administrator).

Responzivni dizajn izmjenjuje forme za registraciju i prijavu na način da forma zauzima skoro cijelu širinu zaslona kako bi bila bolje vidljiva na mobilnim uređajima. Na tabletima je izgled forme ostao u potpunosti isti jer nema potrebe za responzivnosti.

Izgled forme na mobilnim uređajima prikazan je na slici 7.10.

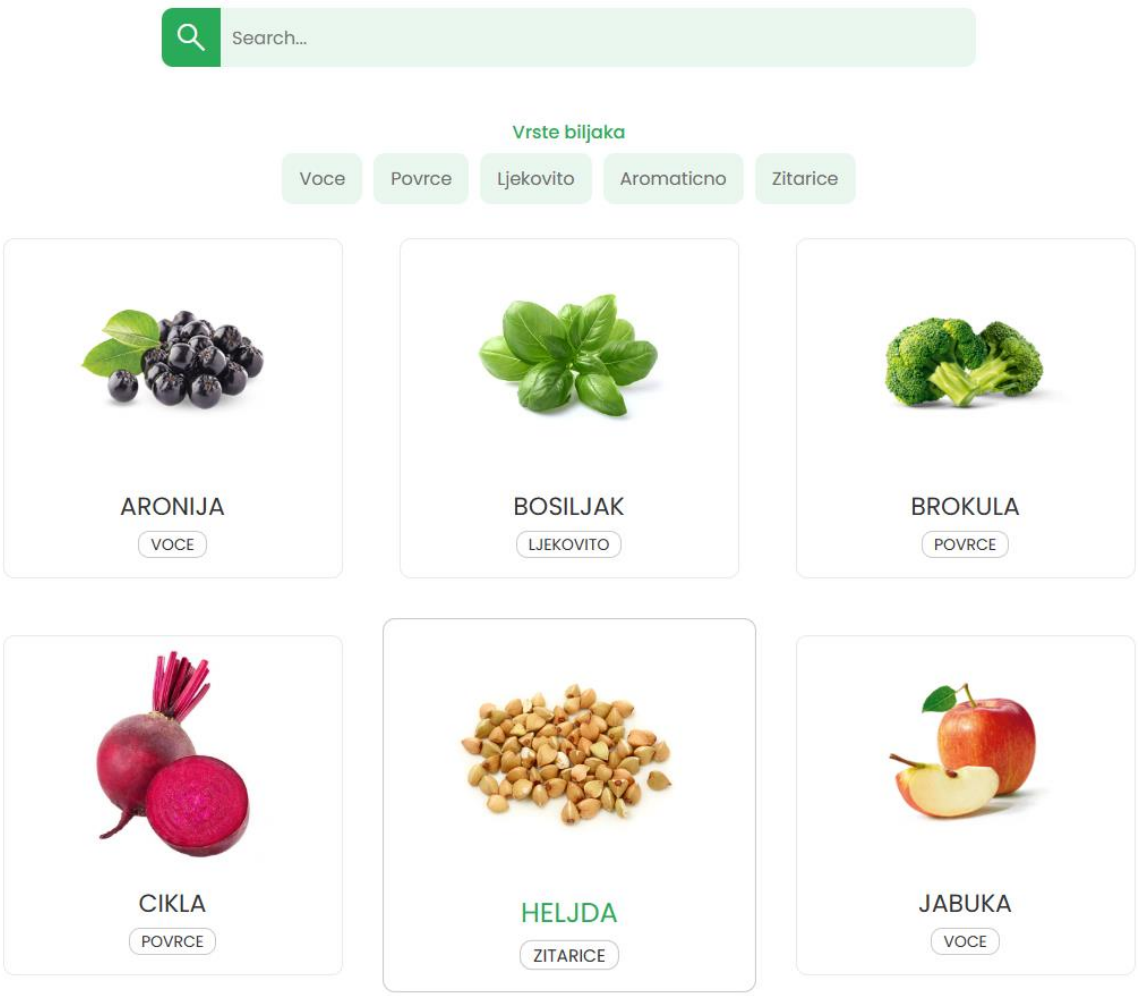



**Sl. 7.10.** Forma za registraciju korisnika na mobilnim uređajima

## 7.4. Stranice za pregled i dodavanje biljaka

Stranice unutar web aplikacije koje su vezane uz biljke su sljedeće: pregled svih biljaka, detalji o pojedinoj biljci i dodavanje nove biljke.

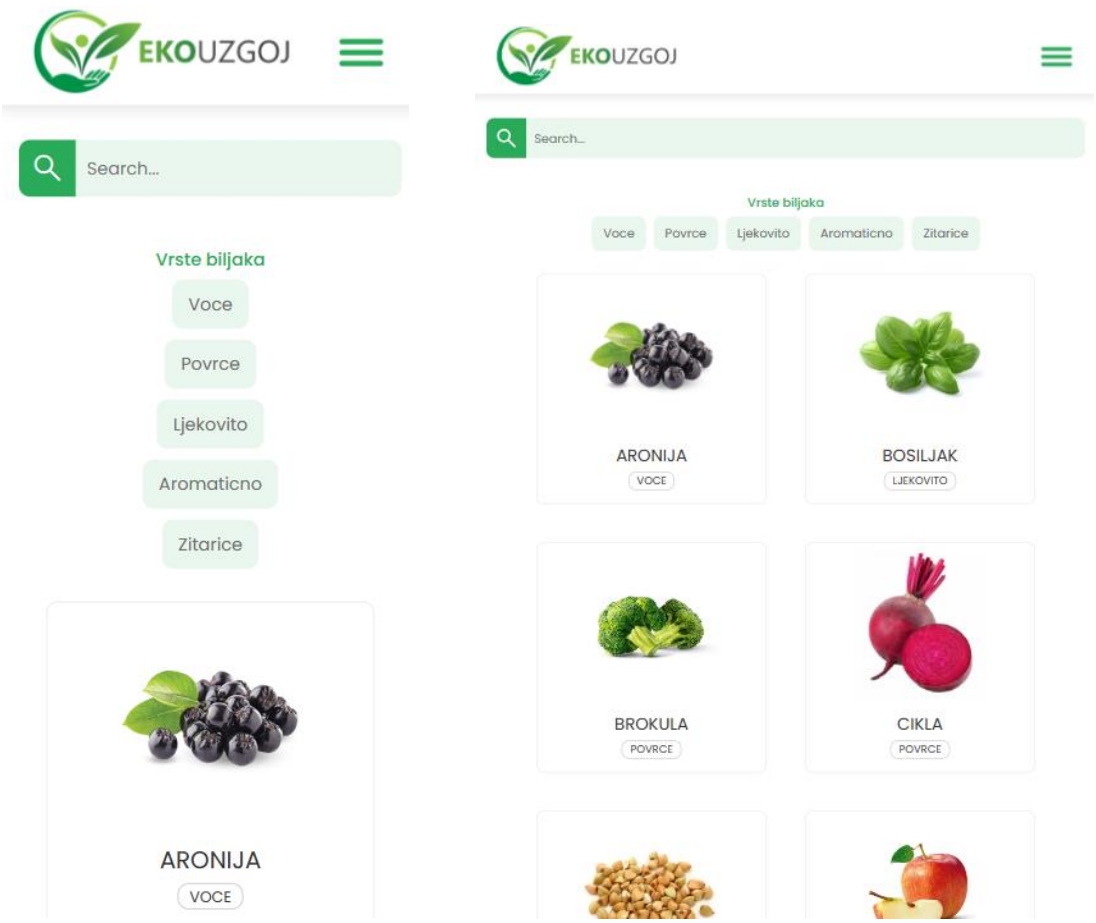
Stranica za pregled svih biljaka sadrži komponente: polje za pretragu biljaka, filtriranje prema vrsti biljaka te prikaz svih biljaka unutar mreže (engl. *grid layout*). Ovisno o tekstu unesenom u polje za pretragu te o odabiru filtera, mreža prikaza biljaka se ažurira prilikom svake promjene. Prilikom prvog otvaranja stranice povlače se podaci iz baze podataka s poslužitelja. Zbog toga što se ne šalju nikakvi parametri pretrage i filtriranja, prikazuju se sve biljke unutar baze podataka. Metoda koja se koristi za zahtjevanje podataka s poslužitelja putem API-a naziva se GET metoda. Dakle, prilikom prvog otvaranja stranice se poziva GET metoda bez ikakvih dodatnih parametara, ali prilikom unosa u polje pretrage ili odabira filtera, poziva ista metoda, ali s odabranim parametrima. Funkciju pretraživanja i filtriranja u bazi podataka obrađuje poslužitelj, dok klijentska strana prima podatke i prikazuje ih. Stranica za pregled svih biljaka prikazana je na slici 7.11.




**Sl. 7.11.** Stranica za pregled svih biljaka

Responzivni dizajn na mobilnim uređajima izmjenjuje pregled svih biljaka na način da mijenja raspored ponuđenih vrsta biljaka. Raspored ponuđenih vrsta biljaka mijenja se iz retka u stupac. Također, ovisno o širini zaslona, automatski se smanjuje broj kartica biljaka po retku tako da stane unutar dostupne širine. Na mobilnim uređajima smanjuje se na 1 karticu po retku, a na tabletima na 2 kartice po retku.

Izgled zaslona za pregled svih biljaka na mobilnim uređajima i tabletima prikazan je na slici 7.12.




**Sl. 7.12.** Stranica za pregled svih biljaka na mobilnim uređajima (lijevo) i tabletima (desno)

Unutar mreže biljaka može se odabrati tj. kliknuti na određenu biljku kako bi se otvorila stranica s detaljima o toj biljki te informacijama o ekološkom uzgoju te biljke. Stranica za detalje o pojedinoj biljci sastoji se od komponenti: naziv, slika, kategorija, opis, detaljan opis načina uzgoja te biljke na ekološki način i komentari. Kada se klikne na biljku, dohvaća se ID te biljke (ID dokumenta u bazi podataka) te se preusmjerava na poveznicu koja sadrži taj ID (npr. "*/plant/6489ce64dcc53f6cea8f916a*"). Kako bi stranica s detaljima mogla zahtjevati podatke o svakoj pojedinoj biljci, šalje se ID iz te poveznice putem API-a. Također se koristi GET metoda, ali ovaj put za dohvaćanje podataka o samo jednoj biljci. Poslužitelj zatim pronalazi biljku u bazi podataka putem atributa ID i vraća podatke na klijentski dio koji ih prikazuje. Stranica s detaljima prikazana je slikom 7.13.

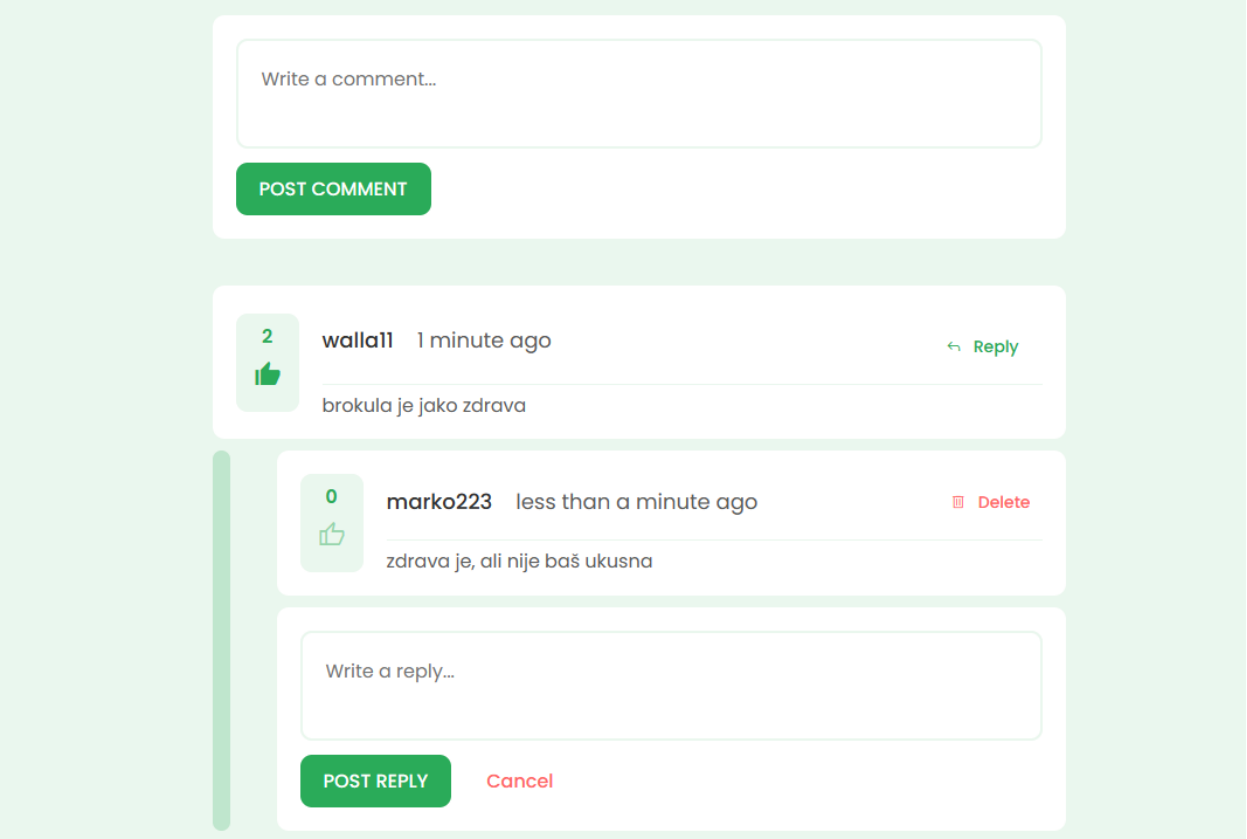
**Sl. 7.13.** Stranica za detalje o biljci

Responzivni dizajn na mobilnim uređajima i tabletima izmjenjuje pregled biljke na način da mijenja cjelokupni raspored. Svi elementi su raspoređeni u jedan stupac koji zauzima skoro pa cijelu širinu zaslona. Na taj način su elementi vidljivi i čitki na manjim uređajima.

Izgled zaslona za pregled biljke na mobilnim uređajima i tabletima prikazan je na slici 7.14.

**Sl. 7.14.** Stranica pojedine biljke na mobilnim uređajima (lijevo) i tabletima (desno)

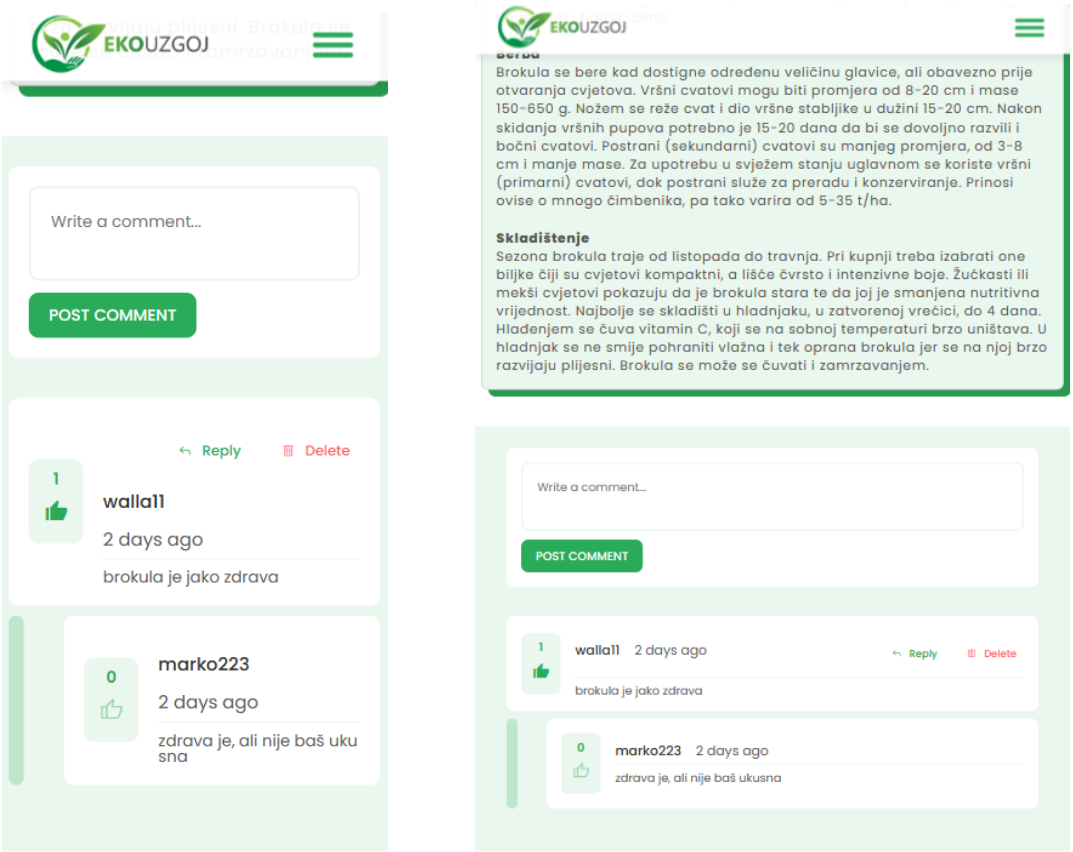
Na stranici svake biljke pri dnu se nalazi dio stranice vezan za komentare. Da bi korisnik mogao napisati komentar, potrebno je da bude prijavljen. Ukoliko je korisnik neprijavljen (gost), ispisuje mu se poruka koja ga upućuje na prijavu ili registraciju. Dio za komentare sastoji se od forme za dodavanje komentara i prikaza mreže samih komentara. Ukoliko je korisnik prijavljen, putem forme za komentare može napisati bilo kakav komentar, savjet ili pitanje vezano za uzgoj određene biljke. Svaki pojedini komentar se sastoji od komentara i odgovora na taj komentar (može ih biti više). Na svaki komentar se mogu dodati novi odgovori odabirom gumba “Odgovori”, čime se otvara forma za dodavanje odgovora. Također, svaki komentar i odgovor sadrži broj ljudi koji su označili da im se komentar/odgovor sviđa. Ukoliko je korisnik prijavljen i napisao je neke komentare/odgovore, nudi mu se opcija za brisanje svog komentara/odgovora. Svakom korisniku je vidljiva ta opcija samo za njegove vlastite komentare/odgovore. Dio za komentare prikazan je na slici 7.15.




**Sl. 7.15.** Dio stranice biljke vezan uz komentare

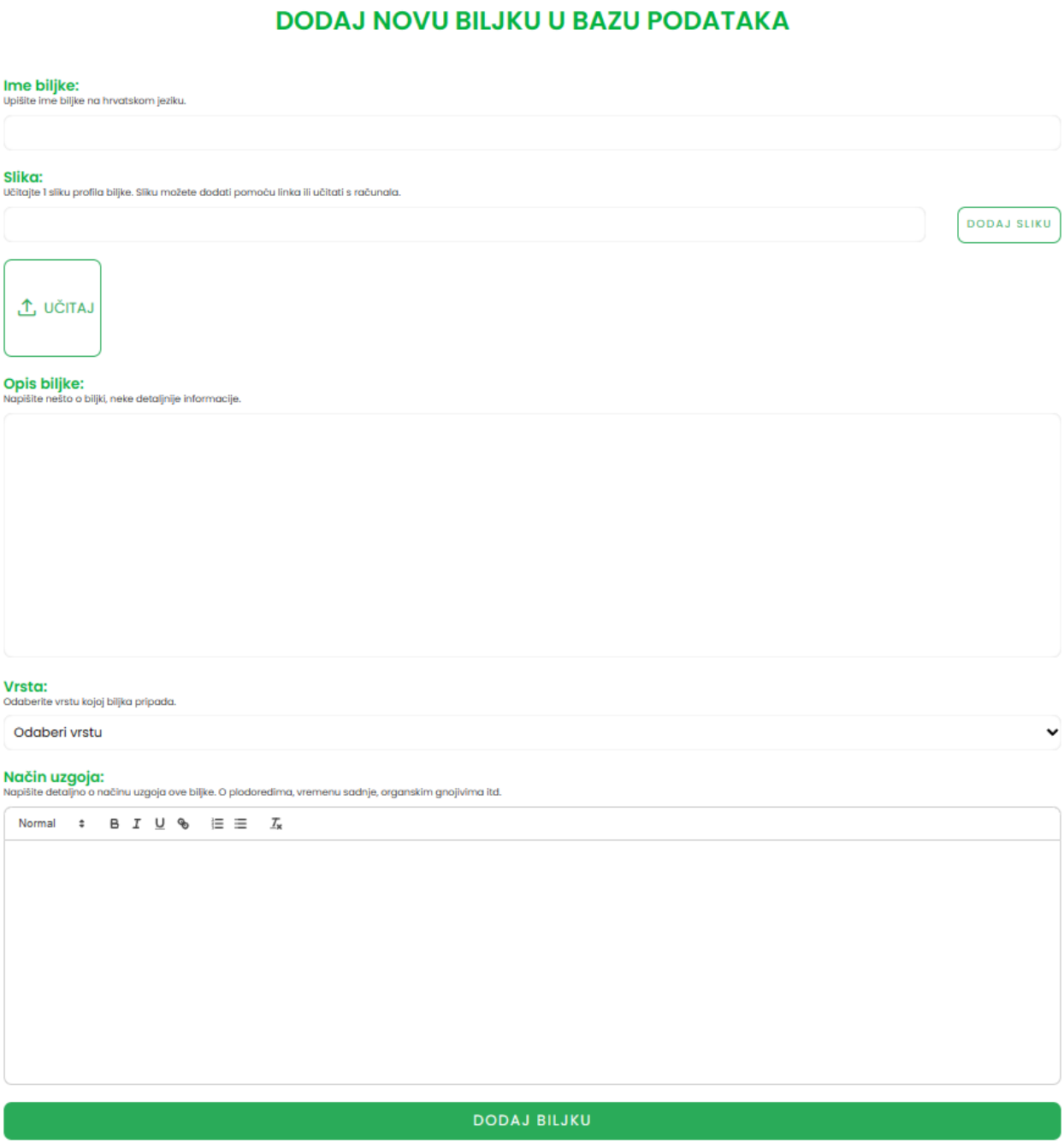
Responzivni dizajn izmjenjuje dio za komentare na način da komentari na manjim uređajima zauzimaju punu širinu zaslona, ali uz isti raspored elemenata. Međutim, elementi koji se nalaze unutar jedne kartice komentara na mobilnim uređajima prelaze iz retka u stupac, dok na tabletima to nije potrebno jer ipak imaju nešto širi zaslon.

Izgled dijela za komentare na mobilnim uređajima i tabletima prikazan je na slici 7.16.




**Sl. 7.16.** Dio za komentare na mobilnim uređajima (lijevo) i tabletima (desno)

Osim te dvije stranice, postoji još i stranica za dodavanje novih biljaka u bazu podataka. Toj stranici može pristupiti samo korisnik koji ima ulogu administratora. Sastoji se od forme koja sadrži sljedeća polja za upis podataka: ime biljke, slika, opis biljke, vrsta i način uzgoja. Slika se može učitati s računala ili putem linka. Polje za unos teksta o načinu uzgoja nudi i oblikovanje teksta. Administrator istražuje kako se pojedina biljka uzgaja na ekološki način i njegov je zadatak napisati i oblikovati taj tekst. Zbog toga mu se nude opcije za dodavanje podnaslova, ulomaka, nabrajanja, veličinu teksta itd. Za podnošenje forme koristi se POST metoda, koja šalje upisane podatke na poslužitelj. Poslužitelj obrađuje taj zahtjev, a ukoliko su neka polja prazna ili je došlo do pogreške, ispod forme se prikazuje poruka crvenom bojom. Stranica za dodavanje nove biljke prikazana je slikom 7.17.



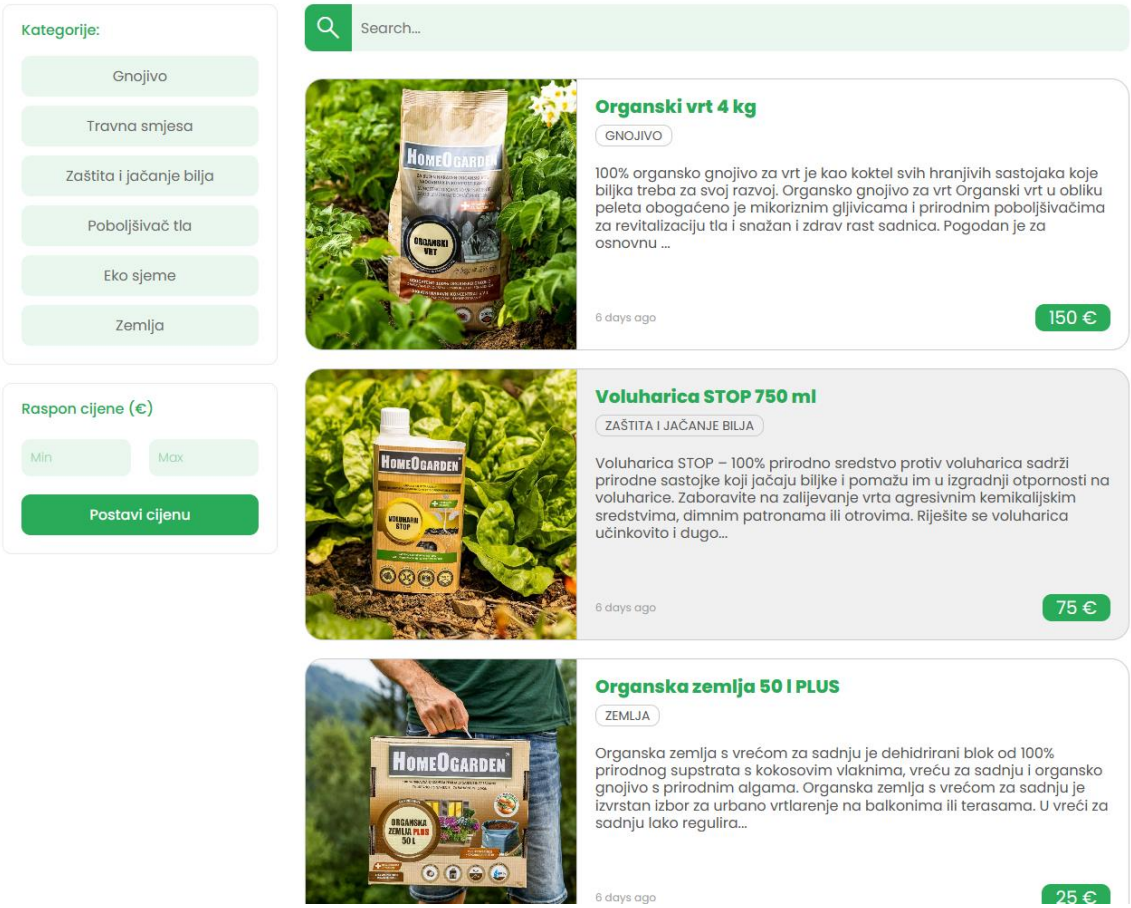

**Sl. 7.17.** Stranica za dodavanje nove biljke

Responzivni dizajn stranice za dodavanje novih biljaka je identičan kao i na velikim zaslonima. Nije prijeko potrebna responzivnost jer svi elementi zauzimaju punu širinu na manjim uređajima, što omogućuje dobru vidljivost i čitkost teksta.

## 7.5. Stranice za pregled i dodavanje oglasa

Stranice vezane uz pregled i dodavanje oglasa su sljedeće: pregled svih oglasa, moji oglasi i dodavanje novog oglasa.

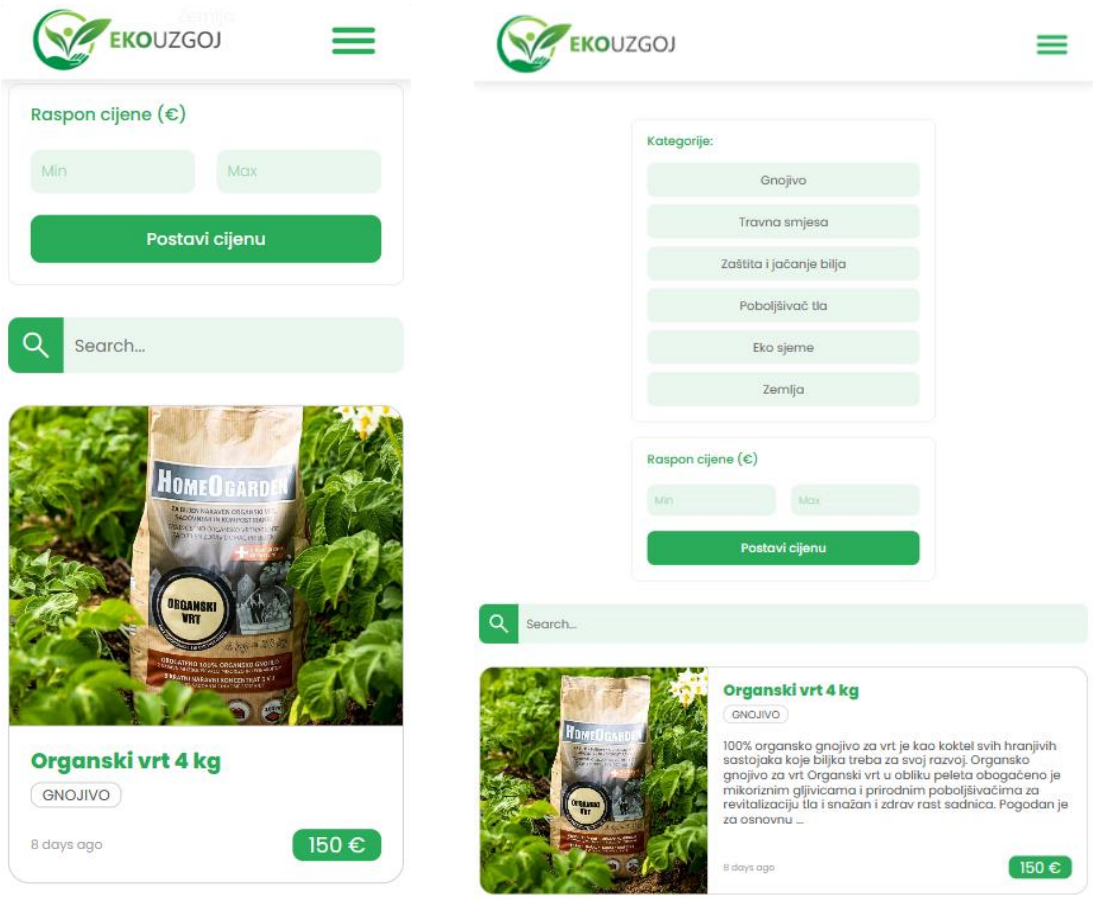
Stranica za pregled svih oglasa sadrži sljedeće komponente: polje za pretragu, filtriranje oglasa, raspon cijena i prikaz kartica oglasa. Kao i kod stranice za prikaz svih biljaka, oglasi se ažuriraju prilikom svake promjene teksta unesenog u polje za pretragu, odabira filtera i ranga cijene. Također se koristi GET metoda koja zahtjeva podatke s poslužitelja putem API-a. Dakle, princip je isti kao i kod stranice za prikaz biljki, samo se ovdje radi o oglasima. Svaka kartica oglasa sastoji se od naslova i kategorije proizvoda, kratkog opisa proizvoda, vremena kada je oglas objavljen i cijene. Stranica za pregled svih oglasa prikazana je na slici 7.18.




**Sl. 7.18.** Stranica za pregled svih oglasa

Responzivni dizajn izmjenjuje stranicu za pregled svih oglasa na način da izmjenjuje cjelokupni raspored. Na manjim zaslonima filtri za kategorije i raspon cijene postavljaju se iznad samih kartica oglasa. Time se ostvaruje bolji pregled oglasa jer tada kartice oglasa mogu zauzeti svu dostupnu širinu zaslona. Na tabletima kartice oglasa prikazuju skraćeni tekst oglasa, dok se na mobilnim uređajima taj tekst ne prikazuje zbog preglednosti i estetike jer nije od prevelikog značaja. Važnije je da se vidi naslov oglasa jer će korisnika prvo zainteresirati sami naslov, a svakako klikom na određeni oglas može vidjeti više informacija.

Izgled zaslona za pregled svih oglasa na mobilnim uređajima i tabletima prikazan je na slici 7.19.




**Sl. 7.19.** Pregled svih oglasa na mobilnim uređajima (lijevo) i tabletima (desno)

Stranica koja prikazuje oglase korisnika (Moji oglasi) identičnog je dizajna kao i stranica za pregled svih oglasa. Međutim, umjesto prikaza svih oglasa, poslužitelj traži oglase i filtrira bazu podataka na osnovu atributa ID korisnika koji je prijavljen. ID korisnika se dohvaća iz lokalne pohrane preglednika, zatim se u zahtjevu prema poslužitelju šalje ID korisnika. Poslužitelj u bazi podataka pronalazi sve oglase koje je kreirao korisnik s tim ID-om, vraća taj skup oglasa te prikazuje kartice oglasa tog korisnika.

Odabirom oglasa tj. klikom na karticu pojedinog oglasa, otvara se stranica s detaljima o oglasu. Također funkcionira na istom principu kao i kod biljaka, samo se dohvaćaju podaci o jednom oglasu s određenim ID-om. Ova stranica se sastoji od prikaza slika, odnosno komponente pomoću koje se pregledavaju slike oglasa (engl. *image slider*), naslova oglasa, kategorije, punog opisa oglasa i cijene. S desne strane nalaze se podaci o prodavaču, odnosno njegovo korisničko ime i kontakt broj. Primjer stranice s detaljima o oglasu prikazan je na slici 7.20.

**PODACI O PRODAVAČU**

**Korisničko ime:**
tajana121

**Kontakt broj:**
0922231314

## Organski vrt 4 kg

GNOJIVO

100% organsko gnojivo za vrt je kao koktel svih hranjivih sastojaka koje biljka treba za svoj razvoj. Organsko gnojivo za vrt Organski vrt u obliku peleta obogaćeno je mikoriznim gljivicama i prirodnim poboljšivačima za revitalizaciju tla i snažan i zdrav rast sadnica. Pogodan je za osnovnu gnojidbu, prihranu i presađivanje svih vrsta biljaka.

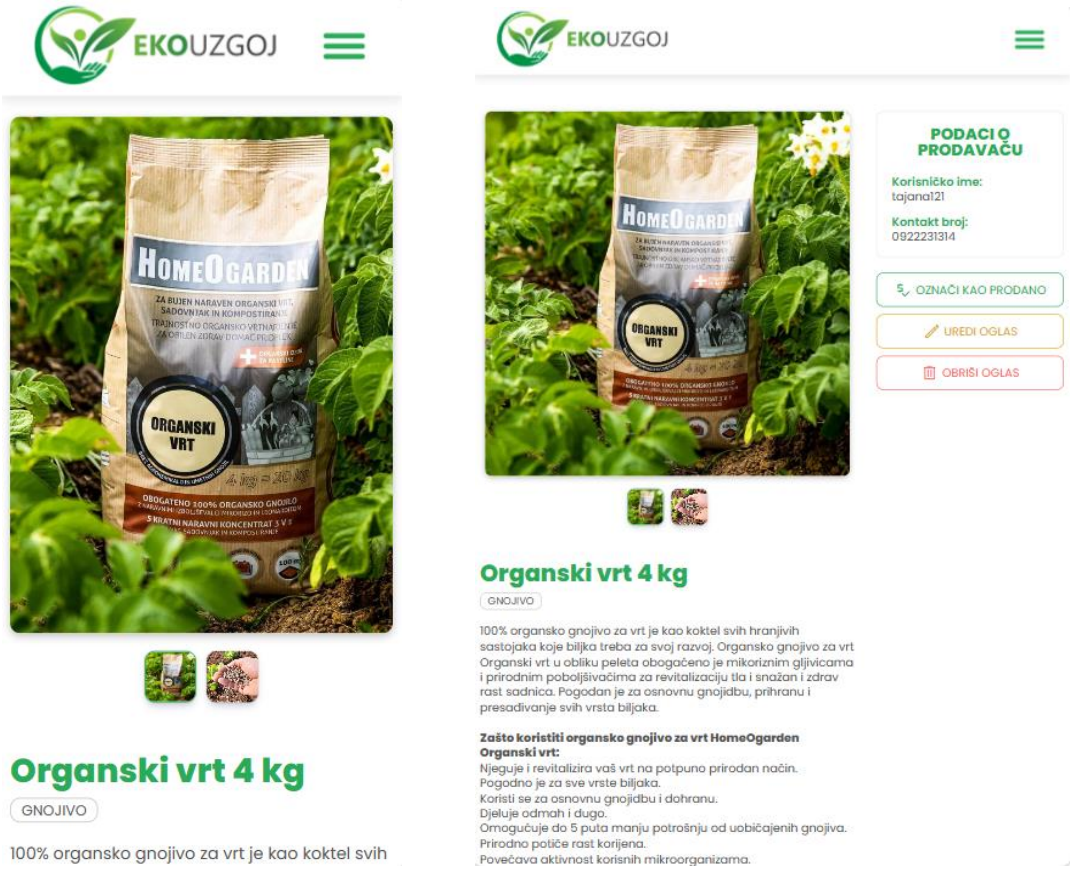
**Zašto koristiti organsko gnojivo za vrt HomeOgarden Organski vrt:**
Njeguje i revitalizira vaš vrt na potpuno prirodan način.
Pogodno je za sve vrste biljaka.
Koristi se za osnovnu gnojidbu i dohranu.
Djeluje odmah i dugo.
Omogućuje do 5 puta manju potrošnju od uobičajenih gnojiva.
Prirodno potiče rast korijena.
Povećava aktivnost korisnih mikroorganizama.
Sadrži hranjive sastojke.
Uz pomoć mikoriznih gljiva biljke dobivaju više vode i hranjivih sastojaka.
Smanjuje kiselost tla.
Bez umjetnih ili kemijski obrađenih sastojaka.
Primjeren je za kompostiranje.
Ima ECO certifikat.

150 €

**Sl. 7.20.** Stranica za prikaz detalja oglasa

Responzivni dizajn izmjenjuje stranicu za prikaz detalja oglasa na način da na mobilnim uređajima mijenja cijeli raspored u jedan stupac koji onda zauzima punu širinu zaslona, dok na tabletima ostaje identičan kao i na većem zaslonu jer je još uvijek dovoljno pregledan i čitljiv.

Izgled zaslona za pregled detalja o oglasu na mobilnim uređajima i tabletima prikazan je na slici 7.21.




**Sl. 7.21.** Stranica za prikaz detalja oglasa na mobilnim uređajima (lijevo) i tabletima (desno)

# 8. ZAKLJUČAK

Briga za okoliš i održivost postaju sve važniji aspekti u današnjem svijetu. Činjenica je da je ekološki uzgoj utjecajan faktor za održivost i očuvanje prirodnih resursa. Samim time što je zabranjena upotreba umjetnih gnojiva, kemikalija i raznih umjetnih materijala dolazi do smanjenja štetnih tvari u zraku i zemlji. Upravo zbog toga, zdravije je tlo, a samim time i plodovi koji nastaju.

Razvoj novih rješenja koja doprinose brizi za okoliš i održivosti također imaju određeni značaj. Kroz ovaj diplomski rad, postavljeni su temelji jednog takvog rješenja u obliku web aplikacije koja ne samo da educira korisnike o ekološkim metodama uzgoja, već i povezuje zajednicu ljudi kojima je stalo do očuvanja prirode. Kroz teorijsko istraživanje o ekološkom uzgoju biljaka i analizu sličnih postojećih web aplikacija, prikupljene su ideje i potrebno znanje za razvoj ove web aplikacije. Razvijena web aplikacija pruža pregledan, pristupačan i jednostavan pristup informacijama te svojom interakcijom s korisnikom olakšava cjelokupno korištenje. Osim toga, web aplikacija je responzivna za sve uređaje, što omogućava svim korisnicima podjednako iskustvo.

Ideja je da se širi svijest o ovakvom načinu uzgoja biljaka stvarajući zajednicu unutar ove web aplikacije. To omogućuje implementirani sustav komunikacije korisnika kojemu je cilj da korisnici pričaju, savjetuju jedni druge, dijele svoja iskustva, pitaju pitanja itd. Također je implementiran i sustav oglašavanja proizvoda vezanih za ekološki uzgoj s ciljem lakšeg pronalaska potrebnih proizvoda, a s druge strane prodavači mogu ponuditi vlastite proizvode. Dakle, sve vezano za ekološki uzgoj biljaka se može pronaći na jednom mjestu, unutar ove web aplikacije.

Razvoj aplikacije su omogućile tehnologije koje pripadaju MERN (MongoDB, Express, React, Node.js) skupu tehnologija. Taj skup tehnologija je široko korišten te postoji mnogo dokumentacije, što je pridonijelo lakšem i bržem razvoju. Aplikacija ima dobre performanse, reaktivno ažuriranje korisničkog sučelja i efikasnu komunikaciju s bazom podataka.

Međutim, usprkos svim implementiranim funkcionalnostima, postoje mogućnosti za razvoj i unaprjeđenje aplikacije. Budući razvoj može implementirati funkcionalnosti kao što su personalizirani planeri uzgoja biljaka za svakog korisnika, informacije na temelju odabrane lokacije i klime, automatizirani podsjetnici za sadnju i održavanje biljaka itd.

Može se zaključiti da ovaj diplomski rad predstavlja temelj za stvaranje digitalnog okruženja koje podržava i promovira ekološki uzgoj biljaka. S obzirom na klimatske promjene, promicanje

ekološkog uzgoja biljaka postaje važnije, a ova web aplikacija ima potencijal doprinijeti očuvanju prirode i usmjeriti budućnost poljoprivrede u smjeru ekologije.

# LITERATURA

[1] Ministarstvo poljoprivrede i šumarstva, „Pravilnik o ekološkoj proizvodnji u uzgoju bilja i u proizvodnji biljnih proizvoda“, Narodne Novine, listopad 2001., eli: /eli/sluzbeni/2001/91/1558

[2] S. Voća, D. Bilandžija, S. Radman, i J. Šic Žlabur, „Ekološki uzgoj ljekovitog i aromatičnog bilja“, Impressum, prosinac 2020.

[3] Fred Magdoff, „Ecological agriculture: Principles, practices, and constraints“, *Renewable Agriculture and Food Systems*, sv. 22, str. 109–117, sij. 2007, doi:10.1017/S1742170507001846

[4] Miguel A. Altieri, Clara I. Nicholls, Marlene A. Fritz, „MANAGE INSECTS On Your Farm: A Guide to Ecological Strategies“, SARE Outreach, 2005.

[5] Ann Larkin Hansen, „The Organic Farming Manual: A comprehensive Guide to Starting and Running a Certified Organic Farm“, Storey Publishing, 2010.

[6] Kathleen Merrigan, Estève G. Giraud, Nadia El-Hage Scialabba, Lena Brook, Allison Johnson, Sarah Aird, „Grow organic: the climate, health, and economic casse for expanding organic agriculture“, The Natural Resources Defense Council, R: 22-10-A., listopad 2022.

[7] Peter V. Fossel, „Organic farming: how to raise, certify, and market organic crops and livestock“, Voyageur Press, 2007.

[8] Minnesota Department of Agriculture, „Overview: Experiences and Outlook of Minnesota Organic Farmers – 2007“, 2007.

[9] Smart Gardener, dostupno na: https://www.smartgardener.com/ [22.8.2023.]

[10] The Old Farmer's Almanac Garden Planner, dostupno na: https://gardenplanner.almanac.com/ [23.8.2023.]

[11] Gardenate, dostupno na: https://www.gardenate.com/ [23.8.2023.]

[13] Agrivi, dostupno na: https://www.agrivi.com/hr/ [15.9.2023.]

[13] Agroklub, dostupno na: https://www.agroklub.com/ [15.9.2023.]

[14] Haroon Shakirat Oluwatosin, „Client-Server Model“, IOSR Journal of Computer Engineering, str. 67-71, veljača 2014.

[15] Rishi Vyas, „Comparative Analysis on Front-End Frameworks for Web Applications“, International Journal for Research in Applied Science & Engineering Technology (IJRASET), str. 298., srpanj 2022., doi: https://doi.org/10.22214/ijraset.2022.45260

[16] Marin Kaluža, Marijana Kalanj, Bernard Vukelić, „A comparison of back-end frameworks for web application development“, Zbornik Veleučilišta u Rijeci, str. 317-332., ožujak 2019., doi: https://doi.org/10.31784/zvr.7.1.10

[17] Alan Beaulieu, „Learning SQL, Second Edition“, O'Reilly Media, travanj 2009.,

[18] Deka Ganesh Chandra, „BASE analysis of NoSQL database“, Future Generation Computer Systems, str. 13-21., studeni 2015., doi: https://doi.org/10.1016/j.future.2015.05.003

[19] Maria Maleshkova, Carlos Pedrinaci, John Domingue, „Investigating Web APIs on the World Wide Web“, Knowledge Media Institute (KMi), str. 107-109., 2010.

[20] Prateek Rawat, Archana N. Mahajan, „ReactJS: A Modern Web Development Framework“, International Journal of Innovative Science and Research Technology, str. 698-699., studeni 2020., ISSN No:-2456-2165

[21] Shama hoque, „Learn MERN stack development by building modern web apps using MongoDB, Express, React, and Node.js“, Packt Publishing, travanj 2020.

[22] MongoDB, „MERN Stack Explained“, prosinac 2022., dostupno na: https://www.mongodb.com/mern-stack [27.6.2023.]

[23] Sitalakshmi Venkatraman, Kiran Fahd, Samuel Kaspi, Ramanathan Venkatraman, „SQL Versus NoSQL Movement with Big Data Analytics“, I.J. Information Technology and Computer Science, prosinac 2016., DOI: 10.5815/ijitcs.2016.12.07

[24] Brett S. Gardner, „Responsive Web Design: Enriching the User Experience“, Noblis, str. 13-16., listopad 2011.

[25] Eva Harb, Paul Kapellari, Steven Luong, Norbert Spot, „Responsive Web Design“, str. 1-6., prosinac 2011.

[26] Interaction Design Foundation, „Responsive Design: Best Practices“, prosinac 2022., dostupno na: https://www.interaction-design.org/literature/article/responsive-design-let-the-device-do-the-work [27.6.2023.]

[27] S. Mohorovičić, „Implementing Responsive Web Design for Enhanced Web Presence“, University of Rijeka, Faculty of Maritime Studies, svibanj 2013.

[28] Eva Harb, Paul Kapellari, Steven Luong, Norbert Spot, „Responsive Web Design“, prosinac 2011.

# SAŽETAK

**Uzgoj biljaka na ekološki način uz potporu web-aplikacije**

U ovom diplomskom radu zadatak je razvoj web aplikacije za podršku ekološkom načinu uzgoja biljaka. Istražene su prakse, metode i ciljevi ekološkog uzgoja. S naglaskom na ekološki uzgoj biljaka, istražena je i potreba za rješenjem u smislu web aplikacije koja educira korisnike o ekološkom uzgoju i potiče stvaranje zajednice ljudi koji podržavaju takav način uzgoja.

Korištene su moderne tehnologije kao što su React, Express, Node.js i MongoDB. Za *frontend* dio je zaslužan React, za *backend* su zaslužni Node.js i Express, dok je MongoDB zaslužan za bazu podataka. Prije samog razvoja su istražene te tehnologije što je omogućilo brže i bolje razumijevanje. U konačnici, razvijena je aplikacija koja korisnicima daje informacije, omogućuje komunikaciju između samih korisnika, omogućuje kupovinu/prodaju proizvoda za ekološki uzgoj te pruža responzivnost za sve uređaje.

KLJUČNE RIJEČI: biljke, ekološki uzgoj, okoliš, web aplikacija

# ABSTRACT

**Growing plants in an ecological way with the support of a web application**

In this master's thesis, the task is to develop a web application to support the ecological way of growing plants. Practices, methods and goals of ecological farming are explored. With an emphasis on ecological plant growing, the need for a solution in terms of a web application that educates users about ecological growing and encourages the creation of a community of people who support such a way of growing was also explored.

Modern technologies such as React, Express, Node.js and MongoDB were used. React is used for the frontend part, Node.js and Express are used for the backend, while MongoDB is used for the database. These technologies were researched before development, which enabled faster and better understanding. Ultimately, an application was developed that provides users with information, enables communication between the users themselves, enables the purchase/sale of ecological farming products and provides responsiveness for all devices.

KEYWORDS: ecological farming, environment, plants, web application

# PRILOZI

PRILOG 1 – sve datoteke i kodovi web aplikacije - https://github.com/vmatokanovic/eco-uzgoj-webapp

# ŽIVOTOPIS

Autor ovog diplomskog rada, Valentin Matokanović, rođen je 14.2.1998.g. u Novoj Gradiški. Osnovnu školu završava u osnovnoj školi Stara Gradiška. Upisuje srednju školu Građevinsku Tehničku Školu u Zagrebu gdje završava i dobiva zvanje građevinskog tehničara. Nakon srednje škole upisuje stručni studij „Automatika“ na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Godine 2020. završava stručni studij, nakon čega polaganjem razlikovne godine, upisuje sveučilišni diplomski studij, smjer „Komunikacije i informatika“.

_____

Potpis autora