

SVEUČILIŠTE U ZAGREBU  
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 805

# **Aplikacija podrške uzgoju i prodaji biljaka**

Jura Starčević

Zagreb, lipanj 2023.

Zagreb, 10. ožujka 2023.

## **ZAVRŠNI ZADATAK br. 805**

Pristupnik: **Jura Starčević (0036531996)**  
Studij: Elektrotehnika i informacijska tehnologija i Računarstvo  
Modul: Računarstvo  
Mentor: prof. dr. sc. Krešimir Fertalj

Zadatak: **Aplikacija podrške uzgoju i prodaji biljaka**

### Opis zadatka:

Provesti analizu procesa uzgoja i prodaje višegodišnjih biljaka. Oblikovati i ugraditi web aplikaciju nad bazom podataka koja će omogućiti evidenciju sjemena i sadnje, planiranje i provedbu uzgoja i presađivanja te praćenje prodaje. Biljke evidentirati prema vrsti i broju te fazi razvoja na pojedinoj lokaciji. Ugraditi cjenike s obzirom na vrstu i veličinu biljke. Evidentirati pripremu, transport, prodaju i zbrinjavanje neprodanih biljaka. Omogućiti korisnički definirana izvješća. Rad aplikacije ilustrirati na primjeru božićnih drvaca.

Rok za predaju rada: 9. lipnja 2023.



## Sadržaj

Uvod .....	1
1. Specifikacija zahtjeva .....	2
1.1. Funkcionalni zahtjevi .....	2
1.2. Nefunkcionalni zahtjevi.....	2
2. Arhitektura sustava .....	4
2.1. Opis arhitekture .....	4
2.2. Baza podataka.....	5
2.2.1. Opis baze podataka .....	5
2.2.2. Opis strukture baze podataka.....	6
2.3. Poslužitelj .....	10
2.3.1. Komunikacija s bazom podataka.....	11
2.3.2. Primanje i obrada HTTP zahtjeva .....	16
2.3.3. Priprema složenijih podataka .....	21
2.3.4. Generiranje web stranica .....	22
2.3.5. Prijava u sustav .....	24
3. Funkcionalnosti aplikacije.....	25
3.1. Prijava u sustav .....	26
3.2. Admin panel .....	27
3.3. Proizvođači sjemena i sadnica .....	28
3.4. Upravljanje sortama.....	31
3.5. Evidentiranje sijanja .....	32
3.6. Evidencija sadnje .....	34
3.7. Evidencija sječe .....	35
3.8. Uređivanje cjenika .....	37
3.9. Upravljanje proizvodima .....	38

3.10.	Upravljanje rezervacijama proizvoda .....	42
3.11.	Izvješće o proizvodima .....	44
3.12.	Pregled i rezervacija proizvoda .....	46
3.12.1.	Pregled proizvoda .....	46
3.12.2.	Rezervacija proizvoda .....	47
4.	Korištene tehnologije .....	48
	Zaključak .....	51
	Literatura .....	52
	Sažetak .....	53
	Summary .....	54

# Uvod

Uzgoj višegodišnjih biljaka, poput smreka, može biti izuzetno zadovoljavajući i dugoročno isplativ. Smreke su popularne zbog svoje ljepote, mirisa i raznovrsnih upotreba poput božićnog drvca. No postupak iza uzgoja je često izazovan i vrlo dugačak. Osim toga je i iterativan jer svake godine je potrebno zasaditi i posijati nove jedinke kako bi imali kontinuiranu zalihu raspoloživih biljnih proizvoda spremnih za distribuciju. Nadodajmo na to još informacije o pojedinim sortama i različite proizvođače koji nam pružaju sjeme i sadnice te dobijemo jednu veliku zbrku informacija koje je teško kvalitetno održavati kroz vrijeme na jednom mjestu, a da one budu korisne i iskoristive u budućoj proizvodnji.

Upravo je to fokus ove web aplikacije, da riješi podatkovnu zavrzlamu te na jednostavan i intuitivan način ponudi unos novih podataka u evidenciju i manipuliranje istima te kao dodatnu funkcionalnost ponudi rezervaciju proizvoda kao oblik prodaje.

Aplikacija treba voditi evidenciju o svim sortama s kojima raspoložemo, proizvođačima od kojih smo ih dobavili, pamtiti informacije o svim sijanjima, presađivanjima i sječama. Potrebno je kreirati konkretne proizvode te cjenike za njih koji će ovisiti o veličinama i sezoni. Sve proizvode treba biti moguće pregledati, filtrirati i rezervirati. Također rezervacije je potrebno moći izmjenjivati, stvarati i brisati po volji.

Sve navedene funkcionalnosti je poželjno ostvariti unutar objektno orijentirane paradigme i načela za pisanje dobrog koda poput načela nadogradnje bez promjena koje stvara naglasak da pisanje sadašnjeg koda treba organizirati na takav način da u budućnosti možemo dodati nove funkcionalnosti bez mijenjanja već napisanog koda (Šegvić and Čupić n.d.).

# 1. Specifikacija zahtjeva

## 1.1. Funkcionalni zahtjevi

- Korisnik ima mogućnost prijave i autorizacije u sustav, nakon koje ima pristup svim podacima u sustavu.
- Korisnik može pregledavati, uređivati, brisati i dodavati nove podatke o sijanju te presađivanju svojih biljaka na proizvoljnim lokacijama
- Korisnik može upravljati podacima o svojim dobavljačima te sortama s kojima raspolaže.
- Korisnik može evidentirati podatke o svakoj provedenoj sječi biljaka.
- Korisnik ima mogućnost dodati nove proizvode te urediti i obrisati postojeće.
- Korisnik može dodavati više različitih cjenika za svaku pojedinu sortu.
- Klijenti korisnika mogu pregledati sve ne rezervirani proizvode.
- Klijenti klikom na proizvod dobiju detalje o proizvodu.
- Klijenti imaju mogućnost rezervacije proizvoda koji se nakon toga uklanja iz ponude.
- Korisnik može pregledati i upravljati svim informacijama vezanih uz rezervaciju proizvoda.
- Korisnik može filtrirati sve postojeće proizvode po stanju i dostupnosti.

## 1.2. Nefunkcionalni zahtjevi

- Aplikacija ima pregledno i intuitivno korisničko sučelje.
- Programski sustav omogućuje nadogradnje bez promjena.
- Osigurati da dva različita klijenta ne mogu rezervirati isti proizvod.
- Provesti autorizaciju korisnika ukoliko želi raditi promjene ili pristupiti svim podacima.

- Aplikacija pruža dodavanje i uređivanje vrijednosti stranih ključeva preko padajuće liste.

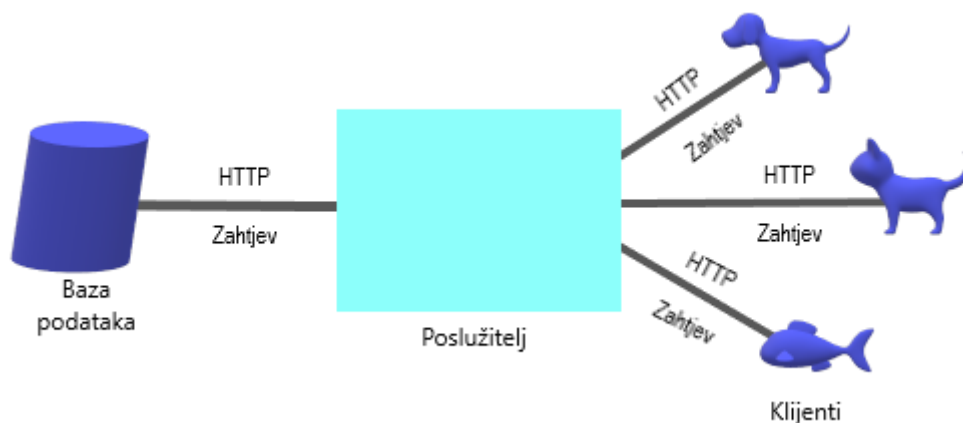


## 2. Arhitektura sustava

### 2.1. Opis arhitekture

U ovom programskom sustavu primijenjen je koncept klijenta i poslužitelja. Klijent je računalo, uređaj ili aplikacija koja zahtijeva neku uslugu ili resurs koji je dostupan putem poslužitelja. Klijent šalje zahtjeve poslužitelju kako bi dobio željene informacije ili izvršio određenu akciju. U kontekstu ove aplikacije klijenti su svi korisnici koji pristupaju web aplikaciji. Poslužitelj je računalo, uređaj ili aplikacija koja je dizajnirana za pružanje određenih usluga klijentima. On prima zahtjeve od klijenata, obrađuje ih i šalje odgovore natrag. Poslužitelj je često sposoban obrađivati veliki broj zahtjeva istovremeno i pružati usluge različitim klijentima istovremeno. Komunikacija između klijenta i poslužitelja u ovom programskom sustavu odvija se preko HTTP zahtjeva.

U kontekstu ove aplikacije, klijenti će komunicirati sa poslužiteljem putem HTTP zahtjeva unutar svog Internet preglednika, poslužitelj će primiti zahtjeve, po potrebi komunicirati s bazom podataka, obraditi zahtjev i vratiti korisniku odgovor u obliku web stranice. Ovaj koncept je ilustriran slikom (Slika 2.1).



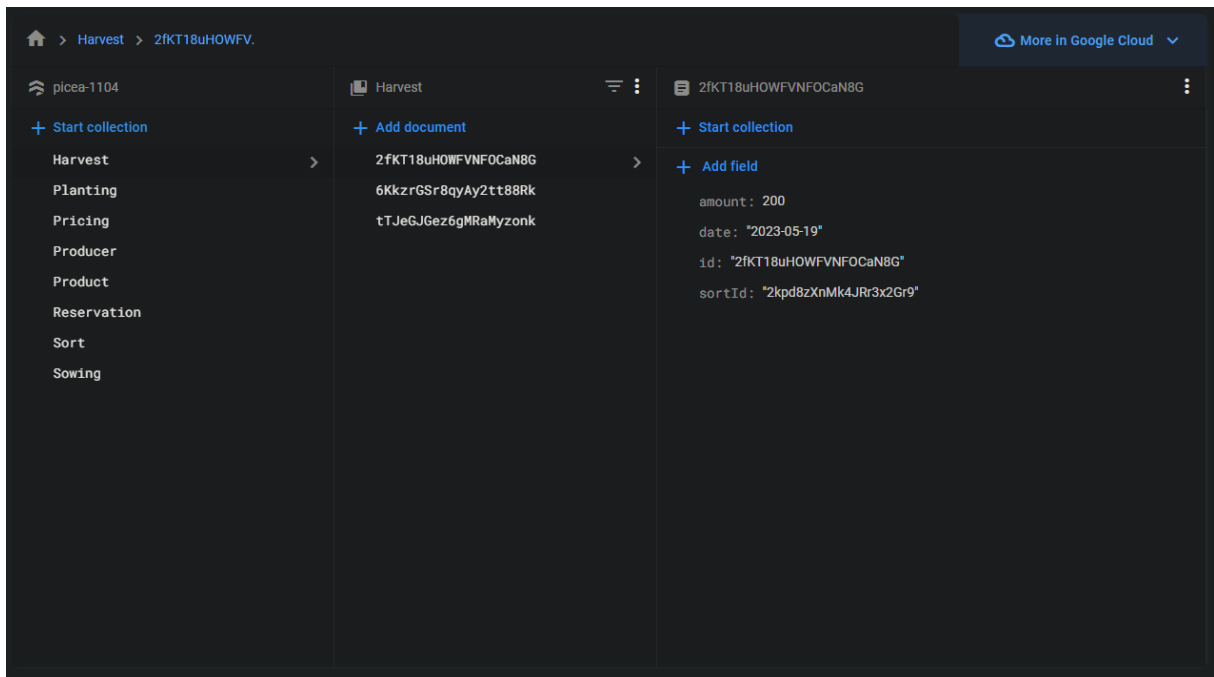
Slika 2.1 Prikaz arhitekture sustava

## 2.2. Baza podataka

### 2.2.1. Opis baze podataka

Baza podataka je sustav se koristi za spremanje, upravljanje i organiziranje informacija. U bazi podataka, podaci su organizirani u tablicama ili kolekcijama koje sadrže redove i stupce. Svaki redak u tablici predstavlja zaseban zapis ili stavku podataka, dok svaki stupac predstavlja određenu karakteristiku ili atribut tog zapisa.

Za bazu podataka korištena je NoSql baza Firebase koja pohranjuje podatke u obliku kolekcija te nije sačinjena od „redova i stupaca“ kao tradicionalne baze (Google 2023). Cijela baza je ustvari jedna kolekcija(lijevi stupac) koji sadrži sve ostale korištene kolekcije(tablice) poput Harvest, Producer, Product i druge.



Slika 2.2 Prikaz baze podataka

Unutar svake pojedinačne kolekcije, primjerice kolekcije Harvest, nalazi se popis dokumenata(srednji stupac) koji su istovremeno i identifikatori svake pojedine instance kolekcije Harvest. U desnom stupcu možemo vidjeti sve atribute od kojih se ta kolekcija sastoji i koje ona informacije pohranjuje.

Baza podataka ove aplikacije sastoji se od 8 kolekcija vidljivih na slici (Slika 2.2).

## 2.2.2. Opis strukture baze podataka

Kolekcija Producer modelira informacije o proizvođačima i dobavljačima te je prikazana tablicom (Tablica 2.1).

Tablica 2.1 Sadržaj kolekcije Producer

Producer		
Vrijednost	Tip vrijednosti	Objašnjenje
producerId	string	Identifikator proizvođača
producerName	string	Ime proizvođača

Kolekcija Sort sadrži podatke o imenima sorte od koje je jedno na latinskom. Prikaz kolekcije nalazi se u tablici (Tablica 2.2).

Tablica 2.2 Sadržaj kolekcije Sort

Sort		
Vrijednost	Tip vrijednosti	Objašnjenje
sortId	string	Identifikator sorte
sortName	string	Ime sorte
Latin	String	Latinsko ime sorte

Kolekcija Sowing predstavlja sijanje biljaka u našoj bazi podataka. Sadrži informacije o kojoj sorti biljke je riječ, dobavljača od kojeg smo dobili sjeme, godinu sijanja i lokaciju na kojoj je posijano. Ilustrirana je tablicom (Tablica 2.3).

Tablica 2.3 Sadržaj kolekcije Sowing

Sowing		
Vrijednost	Tip vrijednosti	Objašnjenje
sowingId	string	Identifikator sijanja
producerId	string	Identifikator proizvođača
sortId	string	Identifikator sorte
year	Int	Godina sijanja
Location	string	Lokacija sijanja

Kolekcija Planting modelira informacije o presađivanjima. Sastoji se od informacija o dobavljaču, sorti, godine sadnje, broju posađenih biljaka i lokaciji sadnje. Prikazan je tablicom (Tablica 2.4).

Tablica 2.4 Sadržaj kolekcije Planting

Planting		
Vrijednost	Tip vrijednosti	Objašnjenje
plantingId	string	Identifikator sadnje
producerId	string	Identifikator proizvođača
sortId	string	Identifikator sorte
year	Int	Godina sadnje
Amount	Int	Broj posađenih sadnica
Location	string	Lokacija sadnje

U kolekciji Harvest pohranjujemo podatke o svakoj sječi biljaka. Ti podatci uključuju informacije o sorti, datumu sječe i broju posječenih biljaka. Prikaz kolekcije vidljiv je u obliku tablice (Tablica 2.5).

Tablica 2.5 Sadržaj kolekcije Harvest

Harvest		
Vrijednost	Tip vrijednosti	Objašnjenje
harvestId	string	Identifikator sječe
sortId	string	Identifikator sorte
Date	String	Datum sječe
Amount	Int	Broj posječenih biljaka

Kolekcija Pricing predstavlja cjenike temeljem kojih određujemo cijenu proizvoda. Cijena proizvoda ovisi o sezoni, sorti i veličini biljke. Ilustracija kolekcije nalazi se u tablici (Tablica 2.6).

Tablica 2.6 Sadržaj kolekcije Pricing

Pricing		
Vrijednost	Tip vrijednosti	Objašnjenje
pricingId	string	Identifikator cjenika
sortId	string	Identifikator sorte
productSize	String	Veličina proizvoda
pricingPrice	double	Cijena proizvoda
Season	String	sezona

Svaki proizvod u bazi je reprezentiran s kolekcijom Product. Ona sadrži sve informacije vezane uz proizvod i njegov razvoj poput informacija o sijanju, sadnji i sječi. Također sadrži informacije o sorti, njegovu sliku, cijenu, detaljniji opis i informaciju u tome nalazili se u posudi. Prikazna je tablicom (Tablica 2.7).

Tablica 2.7 Sadržaj kolekcije Product

Product		
Vrijednost	Tip vrijednosti	Objašnjenje
productId	String	Identifikator proizvoda
sortId	string	Identifikator sorte
Picture	String	Poveznica na sliku
sowingId	String	Identifikator sijanja
plantingId	String	Identifikator sadnje
productSize	String	Veličina proizvoda
pricingId	String	Identifikator cjenika
Description	String	Opis proizvoda
harvestId	String	Identifikator sječe
inPot	Boolean	Govori o tome je li proizvod u posudi

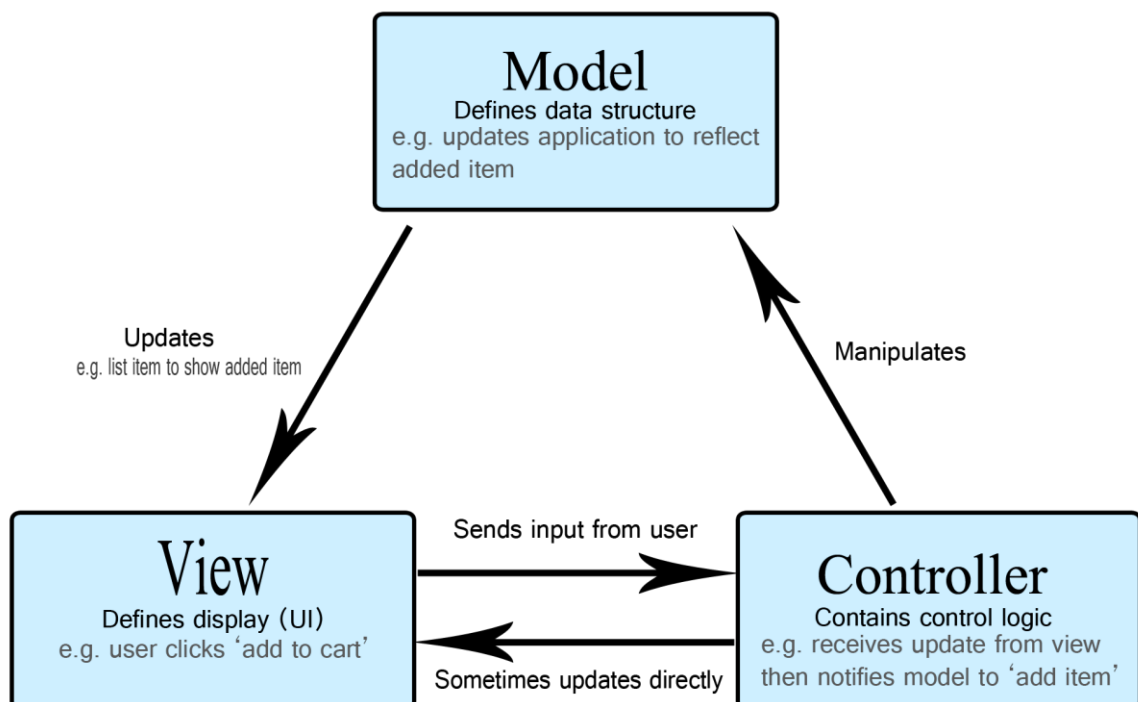
Kolekcija Reservation predstavlja rezervacije proizvoda, stoga sadrži informacije o kojem se konkretno proizvodu radi, osobnim podacima naručitelja te njihovim dodatnim molbama ili napomenama. Prikaz kolekcije vidljiv je u tablici (Tablica 2.8).

Tablica 2.8 Sadržaj kolekcije Reservation

Reservation		
Vrijednost	Tip vrijednosti	Objašnjenje
reservationId	string	Identifikator rezervacije
productId	string	Identifikator proizvoda
reservationName	String	Ime kupca
reservationSurname	String	Prezime kupca
phoneNumber	String	Telefonski broj kupca
Info	String	Dodatne molbe kupca

## 2.3. Poslužitelj

Prilikom izrade programske podrške za poslužitelja primijenjen je arhitekturni obrazac MVC. MVC je kratica za Model-View-Controller (Model-Pogled-Upravljač) i to je arhitekturni obrazac koji se često koristi u razvoju softvera, posebno u razvoju web aplikacija (MDN contributors 2023). Ovaj obrazac pomaže u organizaciji i razdvajanju logike, prikaza i upravljanja podacima u aplikaciji. Prikazan je na slici (Slika 2.3).



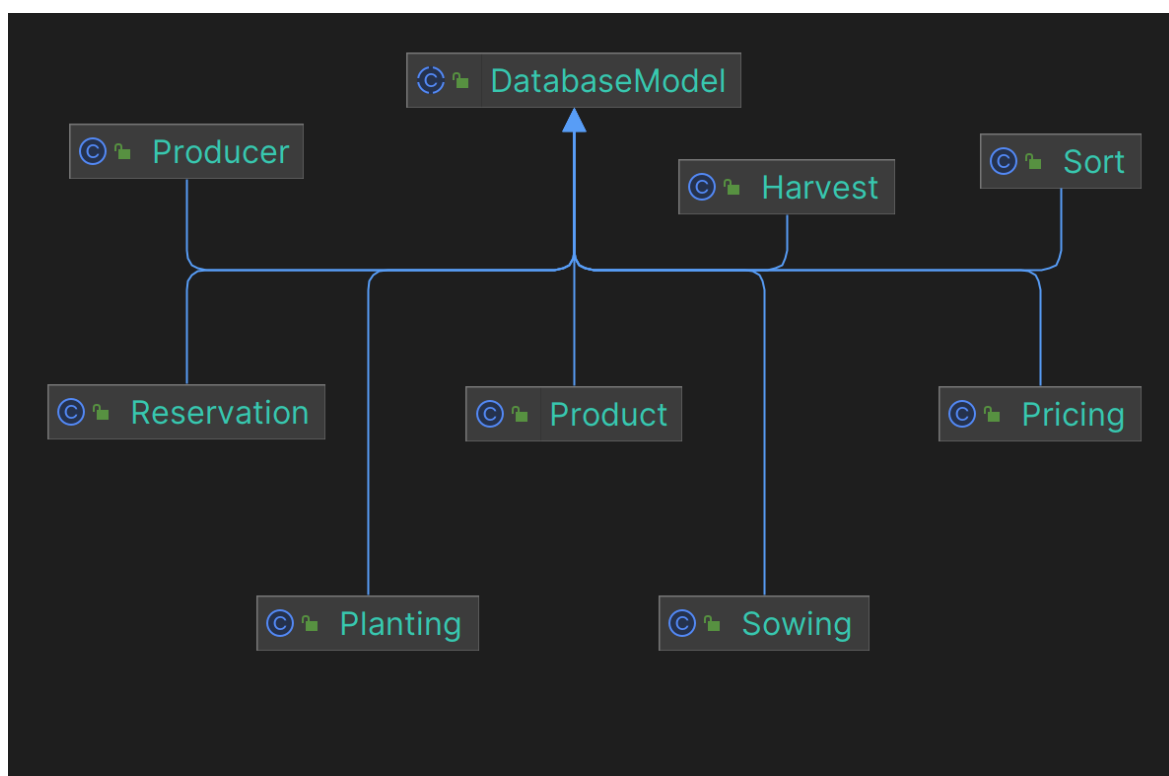
Slika 2.3 Prikaz MVC obrasca (MDN contributors 2023)

Obrazac se sastoji od tri dijela:

- Model - predstavlja podatke i poslovnu logiku aplikacije te je odgovoran za pohranjivanje, obradu i pristup podacima.
- Pogled (View) - predstavlja korisničko sučelje aplikacije koje se koristi za prikazivanje podataka korisniku te je odgovoran za prezentaciju podataka iz modela korisniku na način koji je razumljiv i vizualno privlačan.
- Upravljač (Controller) - djeluje kao posrednik između modela i pogleda i prima korisničke zahtjeve, obrađuje ih i ažurira model i pogled prema potrebi.

### 2.3.1. Komunikacija s bazom podataka

Unutar ovog programskog sustava svu komunikaciju s bazom podataka obavlja Controller. Komunikacija se odvija kroz apstraktni razred *DatabaseModel*. Apstraktni obično imaju apstraktnu metodu ili implementiraju samo neke metode nekog sučelja, a ostale implementacije prepuštaju razredima izvedenim iz njega. U ovoj situaciji ne koristimo ga na taj način nego koristimo svojstvo apstrakcije što znači da objekt tog razreda nije moguće stvoriti što nas štiti od mogućih iznimki. Prikaz ovisnosti je ilustriran dijagramom na slici (Slika 2.4).



Slika 2.4 Dijagram razreda modela

Razred *DatabaseModel* sastoji se od 5 metoda koje pružaju CRUD operacije nad cijelom operacijom. podacima u bazi podataka ili drugom sustavu za upravljanje podacima. Ove operacije su: Create (Stvaranje), Read (Čitanje), Update (Ažuriranje) i Delete (Brisanje). Ovakav pristup ovoj situaciji pruža nam da za svaku postojeću kolekciju u bazi podataka ili potencijalno neku kolekciju koju želimo dodati u budućnosti, samo naslijedimo razred *DatabaseModel* i koristeći njegove metode imamo potpun pristup bazi podataka za taj konkretan razred. Objasnimo detaljnije ovaj pristup na sljedećem primjeru.



### 2.3.1.1 Preuzimanje svih elemenata neke kolekcije

```
/**
 * Default implementation of getAll method for all models that extend DatabaseModel
 * @param db Firestore instance
 * @param t Class of the model
 * @return Collection of models from the database
 * @param <T> Type of the model that extends DatabaseModel
 */
public static <T extends DatabaseModel> List<T> getAll(Firestore db, Class<T> t) {
    String collectionName = t.getSimpleName();
    CollectionReference ref = db.collection(collectionName);
    List<T> list;
    try {
        list = ref.get().get().toObjects(t);
    } catch (ExecutionException | InterruptedException e) {
        throw new RuntimeException(e);
    }

    return list;
}
```

Slika 2.5 Implementacija metode *getAll* metode

Na slici (Slika 2.5) vidimo parametriziranu metodu *getAll*. Funkcija prima dva argumenta: referencu na bazu podataka uz pomoć koje radimo operacije nad bazom i tip razreda za kojeg dohvaćamo podatke. Tip razreda je parametrizirana vrijednost koja može poprimiti ime svih razreda koji nasljeđuju naš razred *DatabaseModel*. Preko tipa razreda u prvom redu funkcije dohvaćamo njegovo ime koje koristimo u sljedećem redu kako bi dohvatili referencu na postojeću kolekciju u bazi podataka. Nakon toga koristimo funkcije ponuđene od vlasnika baze podataka koja nam vraća listu svih podataka unutar naše kolekcije koju vraćamo izlaskom iz naše funkcije. Nakon toga možemo u svakom izvedenom razredu, samo nadjačati tu metodu te prilagoditi da izgleda jednostavnije i bude lakša i intuitivnija za uporabu unutar pojedinog Controllera. Primjer nadjačavanja prikazan je na slici (Slika 2.6).

```
public static List<Harvest> getAll(Firestore db) {
    return DatabaseModel.getAll(db, Harvest.class);
}
```

Slika 2.6 Prikaz nadjačavanja metode *getAll* u naslijeđenom razredu

Ovakav pristup pruža svojstvo nadogradnje bez promjena koje je vrlo cijenjeno u razvoju programske podrške jer smanjuje troškove razvoja te pruža vrlo jednostavnu nadogradnju koda ukoliko je to potrebno (Šegvić and Čupić n.d.). Uz to svojstvo finalna metoda koja se koristi u Controllerima prima samo jedan argument koji je referenca na bazu podataka što olakšava korištenje iste funkcije te smanjuje vjerojatnost „bugova“ jer sva logika komunikacije je smještena unutar razreda *DatabaseModel*. U nastavku slijedi implementacija ostalih CRUD elemenata uz kratak opis.

### 2.3.1.2 Preuzimanje specifičnog elementa

```
/**
 * Default implementation of get method for all models that extend DatabaseModel
 * @param db Firestore instance
 * @param t Class of the model
 * @param id Id of the model
 * @return Model from the database
 * @param <T> Type of the model that extends DatabaseModel
 */
@Jura +1 *
public static <T extends DatabaseModel> T get(Firestore db, Class<T> t, String id) {
    String collectionName = t.getSimpleName();
    String path = collectionName + "/" + id;
    T model;

    try {
        model = db.document(path).get().get().toObject(t);
    } catch (InterruptedException | ExecutionException e) {
        throw new RuntimeException(e);
    }

    return model;
}
```

Slika 2.7 Implementacija metode *get*

Ova metoda ima jedan argument više koji sadrži identifikator elementa kojeg želimo dohvatiti iz baze podataka. Zbog toga potrebno je osim imena kolekcije dodati i na nju spomenuti identifikator što će nam pružiti putanju do željenog elementa u bazi podataka. Ilustracija implementacije je prikazana slikom (Slika 2.7).

### 2.3.1.3 Stvaranje novog elementa

```
/**
 * Default implementation of create method for all models that extend DatabaseModel
 * @param db Firestore instance
 * @param model Model to be created
 * @return ApiFuture of the operation
 * @param <T> Type of the model that extends DatabaseModel
 */
@Sarmaguy
public static <T extends DatabaseModel> ApiFuture<WriteResult> create(Firestore db, T model) {
    String collectionName = model.getClass().getSimpleName();

    String id = db.collection(collectionName).document().getId();
    model.setId(id);

    String path = collectionName + "/" + id;

    return db.document(path).set(model);
}
```

Slika 2.8 Implementacije metode *create*

Prilikom stvaranja novog elementa primamo taj isti element kao argument funkcije. Iz njega doznajemo o kojoj se specifičnoj vrsti radi. Nakon toga dohvaćamo referencu na kolekciju njegove vrste unutar baze podataka, generiramo novi identifikator za njega i pod istim ga pohranjujemo u njegovu kolekciju. Prikaz metode nalazi se na slici (Slika 2.8).

Ova funkcija vraća objekt *ApiFuture* tipa *WriteResult*. Ovaj objekt pruža korisne informacije o rezultatu operacije pisanja, poput vremena izvršenja i ažuriranih podataka (Google 2023). Ove informacije mogu se koristiti za praćenje promjena u bazi podataka, upravljanje konfliktima i potvrdu uspješnosti operacija pisanja (Google 2023).

### 2.3.1.4 Izmjena postojećeg elementa

```
/**
 * Default implementation of update method for all models that extend DatabaseModel
 * @param db Firestore instance
 * @param model Model to be updated
 * @param <T> Type of the model that extends DatabaseModel
 */
@Jura +1
public static <T extends DatabaseModel> ApiFuture<WriteResult> update(Firestore db, T model) {
    String collectionName = model.getClass().getSimpleName();
    String path = collectionName + "/" + model.getId();

    return db.document(path).set(model);
}
```

Slika 2.9 Implementacija metode *update*

Ova funkcija prima izmijenjeni element, iz njega izvlači njegov identifikator te ga vraća na njegovo mjesto u kolekciju. Kao rezultat vraća objekt `ApiFuture` tipa `WriteResult` koji pruža informacije o uspješnosti promjene. Prikazana je slikom (Slika 2.9).

### 2.3.1.5 Brisanje elemenata iz baze podataka

```
/**
 * Default implementation of delete method for all models that extend DatabaseModel
 * @param db Firestore instance
 * @param id Id of the model to be deleted
 * @param t Class of the model
 * @return ApiFuture of the operation
 * @param <T> Type of the model that extends DatabaseModel
 */
@Sarmaguy
public static <T extends DatabaseModel> ApiFuture<WriteResult> delete(Firestore db, String id, Class<T> t) {
    String collectionName = t.getSimpleName();
    String path = collectionName + "/" + id;

    return db.document(path).delete();
}
```

Slika 2.10 Implementacija metode *delete*

Ova metoda dobiva informacije o vrsti i identifikator elementa za brisanje. Nalazi njegovu putanju unutar kolekcija te vrši brisanje te vraća informacije o rezultatu postupka. Metoda *delete* vidljiva je na slici (Slika 2.10).

## 2.3.2. Primanje i obrada HTTP zahtjeva

### 2.3.2.1 Autorizacija zahtjeva

Kako bi osigurali da samo ovlaštene osobe imaju pristup osjetljivim podacima filtriramo sve pristigle zahtjeve. Metoda na slici (Slika 2.11) radi upravo to.

```

@Sarmaguy *
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
        .csrf().disable()
        .authorizeHttpRequests((requests) -> requests
            .requestMatchers("/admin/**").hasRole("ADMIN")
            .anyRequest().permitAll()
        )
        .formLogin()
        .loginPage("/login")
        .permitAll()
        .and()
        .logout(LogoutConfigurer::permitAll);

    return http.build();
}

```

Slika 2.11 Implementacija filtera za autorizaciju

Svi zahtjevi koji u sebi imaju putanju „/admin“ trebaju biti autorizirani prijavljivanjem kao *admin*, inače će biti odbijeni. Ovo također služi i kao zaštita od neželjenih i zloćudnih zahtjeva koji bi mogli naštetiti podacima unutar baze podataka. Više o tome u nastavku.

### 2.3.2.2 Primanje zahtjeva

Temeljna zadaća Controllera u MVC obrascu je primanje i obrada HTTP zahtjeva. Unutar ove web aplikacije svaka kolekcija unutar baze ima svoj Controller zadužen za upravljanje tom kolekcijom. Uz njih su još tri druga koji obrađuju zahtjeve poput pristupa početnoj stranici, admin panelu i drugi.

<b>ReservationController</b> <ul style="list-style-type: none"> <li>showCreateReservationForm (Model) String</li> <li>saveReservation (String, String, String, String, String, String) String</li> <li>saveReservation (String, String, String, String, String) String</li> <li>reservations (Model) String</li> <li>editReservation (String, Model) String</li> <li>deleteReservation (String) String</li> </ul>		<b>PricingController</b> <ul style="list-style-type: none"> <li>showCreatePricingForm (Model) String</li> <li>deletePricing (String) String</li> <li>savePricing (Pricing) String</li> <li>updatePricing (String, Model) String</li> <li>pricings (Model) String</li> <li>savePricing (String, String, String, String, String) String</li> </ul>	
<b>HarvestController</b> <ul style="list-style-type: none"> <li>harvests (Model) String</li> <li>showCreateHarvestForm (Model) String</li> <li>updateHarvest (String, Model) String</li> <li>deleteHarvest (String) String</li> <li>saveHarvest (Harvest) String</li> <li>saveHarvest (String, String, String, int) String</li> </ul>		<b>SowingController</b> <ul style="list-style-type: none"> <li>deleteSowing (String) String</li> <li>sowings (Model) String</li> <li>saveSowing (String, String, String, int, String) String</li> <li>saveSowing (String, String, int, String) String</li> <li>showCreateSowingForm (Model) String</li> <li>updateSowing (String, Model) String</li> </ul>	
<b>PlantingController</b> <ul style="list-style-type: none"> <li>savePlanting (Planting) String</li> <li>plantings (Model) String</li> <li>updatePlanting (String, Model) String</li> <li>savePlanting (String, String, String, int, int, String) String</li> <li>showCreatePlantingForm (Model) String</li> <li>deletePlanting (String) String</li> </ul>		<b>ProducerController</b> <ul style="list-style-type: none"> <li>deleteProducer (String) String</li> <li>producers (Model) String</li> <li>saveProducer (String) String</li> <li>updateProducer (String, Model) String</li> <li>showCreateProducerForm (Model) String</li> <li>saveProducer (String, String) String</li> </ul>	
<b>SortController</b> <ul style="list-style-type: none"> <li>deleteSort (String) String</li> <li>sorts (Model) String</li> <li>saveSort (String, String) String</li> <li>updateSort (String, Model) String</li> <li>saveSort (String, String, String) String</li> <li>showCreateSortForm (Model) String</li> </ul>		<b>ProductController</b> <ul style="list-style-type: none"> <li>saveProduct (Product) String</li> <li>showCreateSortForm (Model) String</li> <li>showCreateProductForm (String, Model) String</li> <li>saveProduct (String, String, String, String, String, String, String) String</li> <li>deleteProduct (String) String</li> <li>updateProduct (String, Model) String</li> <li>products (Model) String</li> </ul>	
		<b>AdminPanel</b> <ul style="list-style-type: none"> <li>adminPanel() String</li> </ul>	
		<b>ReportController</b> <ul style="list-style-type: none"> <li>report (Model) String</li> </ul>	
		<b>HomeController</b> <ul style="list-style-type: none"> <li>saveReservation (Reservation) String</li> <li>home (Model) String</li> <li>product (String, Model) String</li> <li>reserve (String, Model) String</li> </ul>	

Slika 2.12 Prikaz svih Controllera i njihovih metoda

Na slici (Slika 2.12) nalaze se svi korišteni Controlleri i metode koji oni pružaju. Primjer primanja zahtjeva za prikaz admin panela. Unutar razreda *AdminPanel* nalazi se metoda *adminPanel()*. Iznad sebe ima modifikator „@GetMapping“ koji govori da prima HTTP GET zahtjev koji sadrži putanju koja se nalazi unutar zagrada u ovom slučaju „/admin/panel“. Kada naš poslužitelj dobije takav zahtjev pozvat će se metoda koja se nalazi na slici ispod. Ta metoda može odraditi neku funkcionalnost te po izlazu iz nje obično vraća pripremljeni pogled koji će biti prikazan korisniku ili preusmjerenje na neku novu putanju. Metoda je prikazana na slici (Slika 2.13).

```

@Sarmaguy
@Controller
public class AdminPanel {

    @Sarmaguy
    @GetMapping("/admin/panel")
    public String adminPanel(){
        return "admin";
    }
}

```

Slika 2.13 Obrada zahtjeva za Admin panel

### 2.3.2.3 Obrada zahtjeva

Svi Controlleri koji obavljaju funkcionalnosti vezane uz kolekcije iz baze podataka imaju 6 osnovnih metoda kroz koje ćemo proći na primjeru razreda *SortController*. Svim tim metodama imaju pristup samo ovlaštene osobe stoga njihove putanje započinju s „/admin“ kako bi aplikacija uspješno filtrirala validne zahtjeve. Na tu putanju se nadodaje putanja usko vezana uz željenu funkcionalnost.

Primjerice tako sve putanje „/admin/ime\_kolekcije“ od našeg poslužitelja će tražiti ispis svih podataka iz te kolekcije. Metoda koja preuzima taj zahtjev komunicira s bazom te preuzima željene podatke, postavlja ih u model kao pripremu za predaju te vraća pogled u kojem će biti prikazani podatci. Prikaz implementacije nalazi se na slici (Slika 2.14).

```

Sarmaguy
@GetMapping("/admin/sorts")
public String sorts(Model model){
    List<Sort> sorts = Sort.getAll(db);
    model.addAttribute("sorts", sorts);
    return "Sort/sorts";
}

```

Slika 2.14 Primjer obrade zahtjeva za sve podatke unutar kolekcije

Putanja „/admin/new/ime\_kolekcije“ priprema novi model koji ćemo napuniti podacima te vraća pogled koji sadržava formu za preuzimanje informacija o elementu. Obrada zahtjeva ilustrirana je slikom (Slika 2.15).

```

Sarmaguy
@GetMapping("/admin/new/sort")
public String showCreateSortForm(Model model) {
    model.addAttribute("sort", new Sort());
    return "Sort/create-sort";
}

```

Slika 2.15 Priprema forme za unos novog podatka

Putanja „/admin/save/ime\_kolekcije“ preuzima podatke iz forme, stvara novi primjerak elementa kolekcije te ga sprema u bazu podataka. Nakon toga na preusmjerava na stranicu koja sadrži popis svih elemenata te kolekcije. Implementacija je na slici (Slika 2.16).

```

Sarmaguy
@PostMapping("/admin/save/sort")
public String saveSort(@RequestParam("sortName") String sortName, @RequestParam("latin") String latin) {
    Sort sort = new Sort();
    sort.setSortName(sortName);
    sort.setLatin(latin);

    Sort.create(db, sort);

    return "redirect:/admin/sorts";
}

```

Slika 2.16 Prikaz spremanja novog podatka u bazu podataka

Putanja „/admin/delete/ime\_kolekcije/id“ se koristi kada želimo obrisati element s identifikatorom koji je dio putanje. Controller prima zahtjev i poziva odgovarajuću metodu za komunikaciju s bazom podataka. Obrada zahtjeva prikazana je na slici (Slika 2.17). Nakon brisanja vraća nas na popis elemenata.

```

Sarmaguy
@PostMapping("/admin/delete/sort/{id}")
public String deleteSort(@PathVariable("id") String id) {
    Sort.delete(db, id);

    return "redirect:/admin/sorts";
}

```

Slika 2.17 Prikaz obrade zahtjeva za brisanje

Putanja „/admin/update/ime\_kolekcije/id“ priprema podatke za pogled koji će generirati formu za izmjenu podataka te preusmjerava na nju (Slika 2.18).

```

Sarmaguy
@PostMapping("/admin/update/sort/{id}")
public String updateSort(@PathVariable("id") String id, Model model) {
    Sort sort = Sort.get(db, id);
    model.addAttribute("sort", sort);
    return "Sort/edit-sort";
}

```

Slika 2.18 Priprema forme za izmjenu podataka



Putanja „*/admin/save/ime\_kolekcija/id*“ preuzima podatke iz prethodno spomenute forme i sprema ih bazu podataka. Nakon spremanja podataka usmjereni smo na stranicu sa popisom svih elemenata iz kolekcije. Implementacija se nalazi na slici (Slika 2.19).

```
± Samaguy
@PostMapping("@v"/admin/save/sort/{id})
public String saveSort(@PathVariable("id") String id, @RequestParam("sortName") String sortName, @RequestParam("latin") String latin) {
    Sort sort = Sort.get(db, id);
    sort.setSortName(sortName);
    sort.setLatin(latin);

    Sort.update(db, sort);

    return "redirect:/admin/sorts";
}
```

Slika 2.19 Implementacija spremanja uređenog podatka

Controller *AdminPanel* radi samo preusmjeravanje tj. prima zahtjeva na putanju „*/admin/panel*“ i vraća korisniku na zahtjev traženu web stranicu ukoliko je prijavljen. Ukoliko korisnik nije prijavljen preusmjerava ga na stranicu za prijavu. Ova funkcionalnost se događa svaki put kada korisnik zatraži pristup zaštićenim resursima ili aktivnostima(svi zahtjevi koji u svojoj putanji imaju riječ „*admin*“).

Controller *Home* sadrži 4 metode. Osnovna metoda *home()* priređuje sve dostupne proizvode i kao odgovor ih šalje unutar web stranice. Metoda *product()* koja je vezana uz putanju „*/product/id*“ koja komunicira s bazom podataka, preuzima informacije o proizvodu s odgovarajućim identifikatorom i vraća detalje o njemu. Proizvode prikazane na početnoj stranici moguće je rezervirati te o tome brine metoda *reserve()* na putanji „*/reserve/id*“ koja priprema formu koju korisnik treba ispuniti kako bi rezervirao proizvod. Nakon ispunjavanja forme šalje se POST zahtjev koji preuzima metoda *saveReservation()*. Ona brine o kreaciji nove rezervacije te da u međuvremenu neki drugi korisnik nije rezervirao taj isti proizvod tako da prođe kroz već sve postojeće rezervacije i provjeri sadržava li ikoja od njih isti proizvod. Točna implementacija nalazi se na slici (Slika 2.20).

```

Sarmaguy
@PostMapping("/reservation")
public String saveReservation(Reservation reservation) {

    //check if in database is reservation with same productId
    List<Reservation> reservations = Reservation.getAll(db);
    for (Reservation res : reservations) {
        if (res.getProductId().equals(reservation.getProductId())) {
            return "redirect:/";
        }
    }

    Reservation.create(db, reservation);
    return "redirect:/";
}

```

Slika 2.20 Implementacija stvaranja nove rezervacije

### 2.3.3. Priprema složenijih podataka

Određeni tipovi podataka u svojim kolekcijama sadržavaju atribut koji je strani ključ. U kontekstu baza podataka, strani ključ je atribut ili skup atributa u jednoj kolekciji koji se referencira na primarni ključ u drugoj kolekciji. Strani ključevi koriste se za uspostavljanje veza između različitih kolekcija u bazi podataka. Stoga je poželjno povezati podatke iz više kolekcija u jedan objekt koji tada koristimo za prikaz podataka.

ProductVM	HarvestVM	SowingVM
priceInfo String	id String	producerName String
reserved boolean	sortName String	location String
productSize String	date String	sortName String
sortName String	amount int	year int
plantingInfo String		id String
inPot String	PricingVM	
description String	Id String	PlantingVM
harvestInfo String	season String	year int
id String	size String	location String
sowingInfo String	price String	id String
pictureUrl String	sortName String	amount int
		producerName String
		sortName String

Slika 2.21 Prikaz korištenih ViewModela

Na slici (Slika 2.21) je prikaz razreda i njihovih atributa koji obavljaju tu zadaću. Zovemo ih ViewModels. Svaki od tih razreda sadrži metodu *from()* koja na ulaz prima više vrsta podataka i kombinira ih u jedan primjerak tipa razreda u kojemu se nalazi. Popis metoda ovih razreda nalazi se na **slici**. Razred *ProductVM* ima nekoliko metoda *from()*, no jedina razlika među njima je broj i vrsta argumenata koju primaju. Osim toga taj razred sadrži i metodu *isInPot()* koja priređuje informaciju o tome nalazi li se proizvod u posudi ili ne. Prikaz metoda unutar ViewModela nalazi se na slici (Slika 2.22).

© ProductVM	
from(List<Product>, List<Sort>, List<Sowing>, List<Planting>, List<Harvest>, List<Pricing>)	List<ProductVM>
fromHome(List<Product>, List<Sort>, List<Sowing>, List<Planting>, List<Harvest>, List<Pricing>, List<Reservation>)	List<ProductVM>
from(Product, Firestore)	ProductVM
isInPot()	String
from(List<Product>, List<Sort>, List<Sowing>, List<Planting>, List<Harvest>, List<Pricing>, List<Reservation>)	List<ProductVM>
© PlantingVM	
from(List<Planting>, List<Producer>, List<Sort>)	List<PlantingVM>
© SowingVM	
from(List<Sowing>, List<Producer>, List<Sort>)	SowingVM
© HarvestVM	
from(List<Harvest>, List<Sort>)	List<HarvestVM>
© PricingVM	
from(List<Pricing>, List<Sort>)	List<PricingVM>

Slika 2.22 Prikaz metoda unutar ViewModela

### 2.3.4. Generiranje web stranica

Spring Framework omogućuje generiranje web stranica pomoću Thymeleaf template enginea. Thymeleaf je popularan alat za dinamičko generiranje web sadržaja temeljen na predlošcima (The Thymeleaf Team 2023). Predlošci su HTML datoteke koje sadrže Thymeleaf oznake i izraze. Ove oznake i izrazi omogućuju dinamičko generiranje sadržaja i manipuliranje podacima koji se prikazuju na stranici (The Thymeleaf Team 2023). U controlleru smo prethodno pripremili željene podatke za prikaz u obliku modela. Također prilikom obrade zahtjeva u controlleru potrebno je i vratiti model i prikaz(„View“). Spring će automatski povezati predložak s modelom podataka. Model podataka može sadržavati attribute koji će se koristiti u Thymeleaf izrazima kako bi se generirao dinamički sadržaj. Spring će obraditi predložak pomoću Thymeleaf template enginea. Thymeleaf će analizirati predložak, izvršiti izraze i oznake, te generirati konačni HTML sadržaj. U tom procesu, Thymeleaf će koristiti podatke iz modela podataka za generiranje dinamičkog sadržaja (The Thymeleaf Team 2023). Generirani HTML rezultat bit će poslan klijentu i prikazan kao web stranica. Na web stranici će se prikazati dinamički sadržaj koji je generiran pomoću Thymeleaf izraza i podataka iz modela podataka.

```

<h1>List of Sorts</h1>
<table>
  <thead>
    <tr>
      <th>Sort</th>
      <th>Latin Name</th>
      <th>Action</th>
    </tr>
  </thead>
  <tbody>
    <tr th:each="sort : ${sorts}">
      <td th:text="${sort.sortName}"></td>
      <td th:text="${sort.latin}"></td>
      <td>
        <div class="btn-group">
          <form th:action="@{/admin/update/sort/{id}(id=${sort.id})}" method="post">
            <input type="hidden" name="_method" value="post"/>
            <button type="submit" class="btn btn-primary">Update</button>
          </form>
          <form th:action="@{/admin/delete/sort/{id}(id=${sort.id})}" method="post">
            <input type="hidden" name="_method" value="post"/>
            <button type="submit" class="btn btn-danger">Delete</button>
          </form>
        </div>
      </td>
    </tr>
  </tbody>
</table>

```

Slika 2.23 Implementacija pogleda koji prikazuje sve sorte

Na slici (Slika 2.14) smo na određeni HTTP zahtjev kontaktirali bazu podataka, pripremili podatke, spremili ih u model i vratili pogled. Taj pogled se nalazi na slici (Slika 2.23) gdje ispisujemo tablicu svih podataka. U Controlleru smo pripremili listu sortova, ovdje koristeći Thymeleaf engine iteriramo kroz listu i ispisujemo ime te latinski naziv za svaku sortu. Uporabu unutar HTMLa možemo prepoznati po tome što se u tome retku nalazi kratica „th:“.

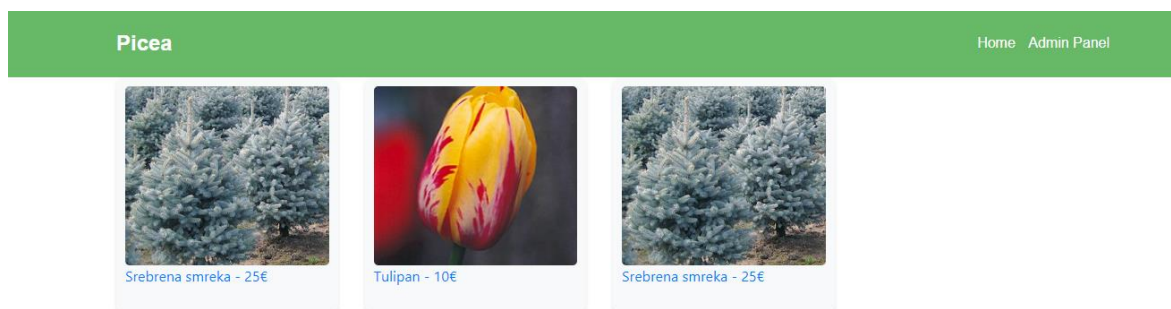
### **2.3.5. Prijava u sustav**

Prijava u sustav ostvarena je preko Spring Securitya. Spring Security je Java okvir koji pruža podršku za sigurnost i autorizaciju u aplikacijama koje se temelje na Spring okviru. Spring Security omogućuje programerima da lako implementiraju različite mehanizme zaštite, kao što su autentifikacija korisnika, autorizacija pristupa resursima i zaštita od raznih sigurnosnih prijetnji (VMware 2023).

Za potrebe ovog programskog sustava dovoljno je imati samo jednog administratora stoga ovaj sustav nema registracijski proces te sadrži samo jednog administratora.

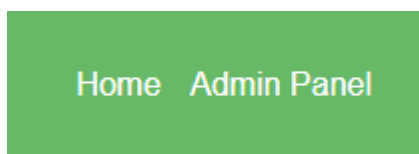
### 3. Funkcionalnosti aplikacije

Na početnoj stranici prikazanoj na slici (Slika 3.1) korisnici mogu pregledati postojeće dostupne proizvode. Klikom na sliku ili opis proizvoda otvorit će se web stranica s dodatnim informacijama o tome proizvode i mogućnost rezervacije.



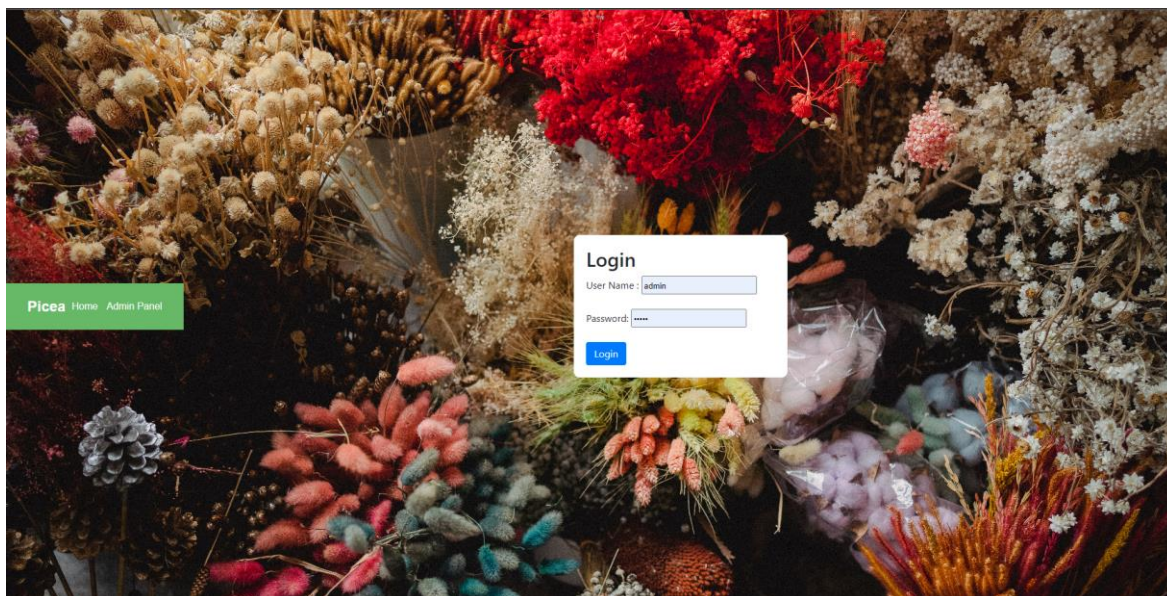
Slika 3.1 Prikaz početne stranice

Osim toga na početnoj stranici nalaze se i dvije poveznice vidljive na slici (Slika 3.2). Klikom na „Home“ vraćamo se na početnu stranicu. Ako smo prijavljeni u sustav pritiskom na „Admin Panel“ biti ćemo prebačeni na admin panel koji sadržava opcije za upravljanje podacima. Ako nismo prijavljeni, biti ćemo preusmjereni na stranicu za prijavu. Ovaj izbornik će se pojavljivati kroz web aplikaciju kao navigacijsko sredstvo.



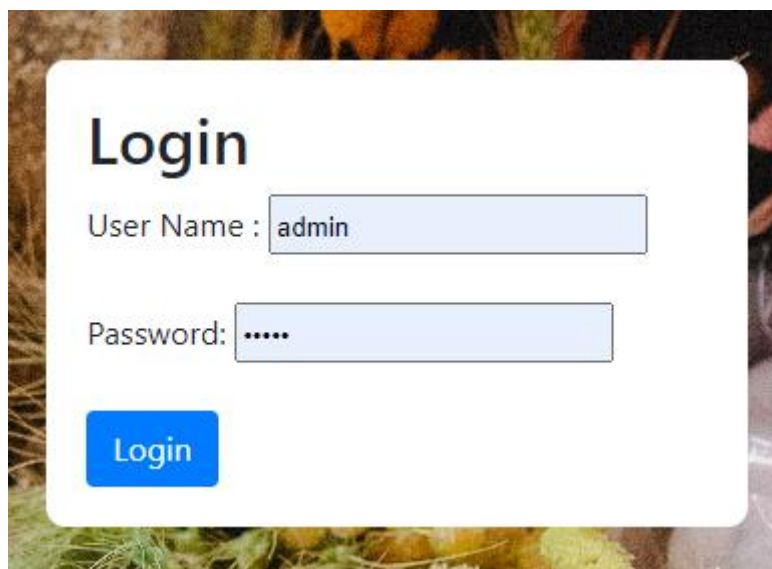
Slika 3.2 Prikaz poveznica

### 3.1. Prijava u sustav



Slika 3.3 Stranica za prijavu

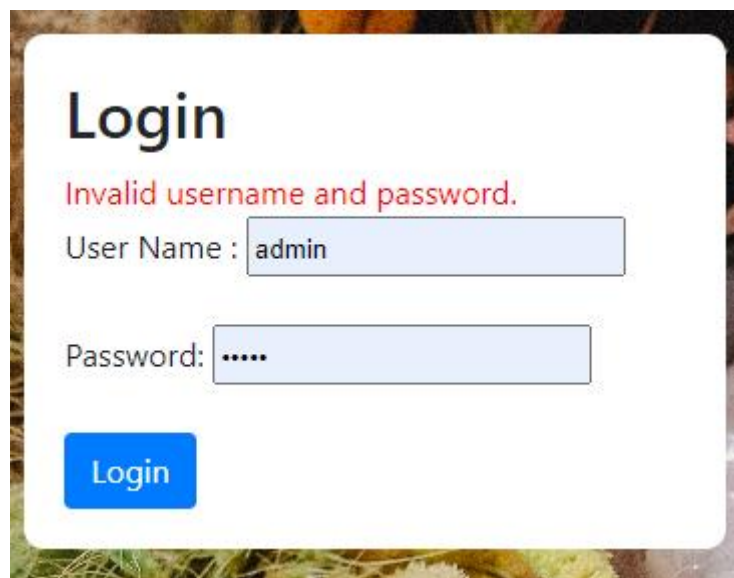
Stranica za prijavu prikazana je na slici (Slika 3.3). Ona se sastoji od navigacije u lijevom kutu i forme za prijavu vidljive na slici (Slika 3.4).



Slika 3.4 Forma za prijavu

Ukoliko su podatci ispravni preusmjereni smo na web stranicu admin panela, no ukoliko nisu vratit ćemo se na ovu stranicu i dobiti obavijest o neispravnom unosu podataka vidljivo na slici (Slika 3.5).



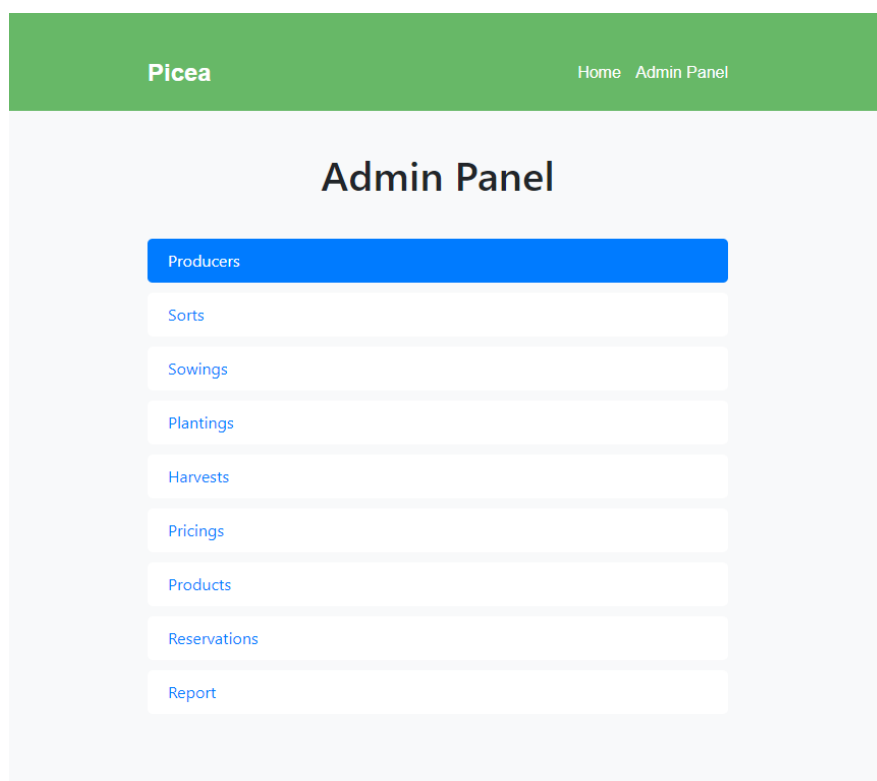


Slika 3.5 Dojava o grešci

Iz sigurnosnih razloga aplikacija ne specizira koja vrijednost nije valjana nego vraća ovu generaliziranu poruku o neuspjehu.

## 3.2. Admin panel

Nakon uspješne pripreme prikazuje nam se sljedeća stranica prikazana na slici (Slika 3.6).



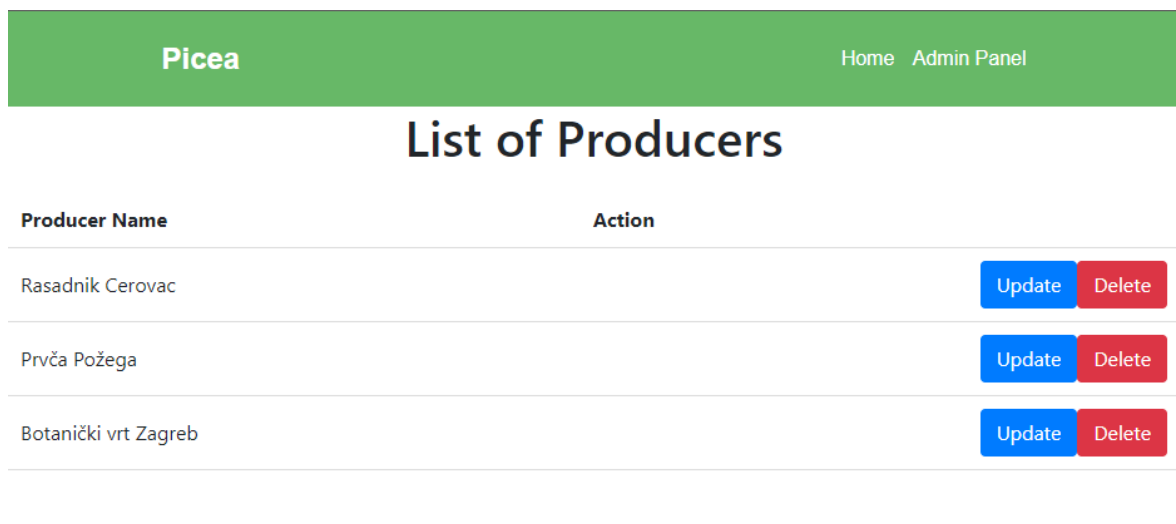
Slika 3.6 Prikaz Admin Panela



Stranica se sastoji od navigacije u gornjem dijelu stranice te poveznica na svaku posebnu funkcionalnost. Ukoliko prijedemo pokazivačem preko okvira jedne poveznice ona će promijeniti boju u plavu. Administratorski dio funkcionalnosti ćemo prijeći na primjeru dodavanjem novog proizvoda u naš sustav. Taj proizvod će biti obična smreka.

### 3.3. Proizvođači sjemena i sadnica

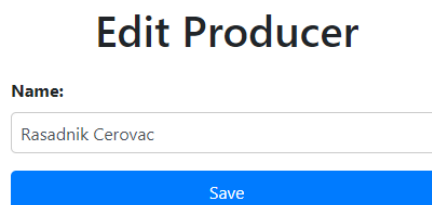
Ova funkcionalnost aplikacije modelira informacije o proizvođačima od kojih smo dobavili sjeme za sijanje naših biljaka ili sadnice koje su naknadno posađene. Klikom na opciju *Producers* na slici (Slika 3.6) otvara nam se glavna stranica proizvođača koja sadrži pregled postojećih unutar aplikacije. Stranica je prikazana na slici (Slika 3.7).



Producer Name	Action
Rasadnik Cerovac	<button>Update</button> <button>Delete</button>
Prvča Požega	<button>Update</button> <button>Delete</button>
Botanički vrt Zagreb	<button>Update</button> <button>Delete</button>

Slika 3.7 Popis svih proizvođača

Svaku postojeću vrijednost moguće je odraditi operaciju brisanja koja ju briše s popisa i iz baze podataka. Ukoliko želimo promijeniti informacije o proizvođaču to činimo pritiskom na plavo dugme *Update*. Nakon klika na to dugme otvorit će nam se nova forma za izmjenu podataka. Ona će sadržavati trenutne vrijednosti podataka koje želimo mijenjati. Ova funkcionalnost ilustrirana je na slici (Slika 3.8).



### Edit Producer

Name:

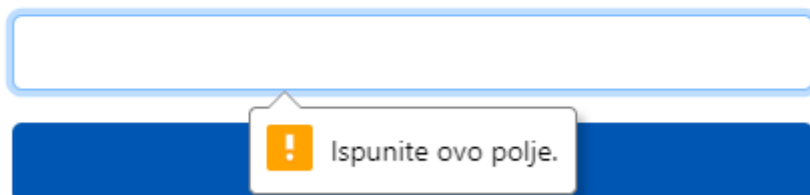
Save

Slika 3.8 Forma za izmjenu proizvođača

Forma provodi validaciju podataka stoga je nemoguće spremiti prazan podatak. Ukoliko se to pokuša dobit ćemo upozorenje o istome prikazano na slici (Slika 3.9).

## Edit Producer

Name:



Slika 3.9 Primjer validacije podataka

Radi demonstracije funkcionalnosti vrijednost će biti promjenjena na „Zeleni rasadnik Cerovac“. Nakon promjene klikom na dugme *Save* biti ćemo vraćeni na stranicu proizvođača gdje će promjena biti vidljiva što je ilustrirano slikom (Slika 3.10).

Picea		Home	Admin Panel
List of Producers			
Producer Name	Action		
Zeleni rasadnik Cerovac	Update	Delete	
Prvča Požega	Update	Delete	
Botanički vrt Zagreb	Update	Delete	

Slika 3.10 Popis svih proizvođača nakon izmjene podatka

Ukoliko želimo dodati novog proizvođača to činimo pritiskom na dugme „Create new producer“ koje se nalazi ispod popisa prikazano na slici (Slika 3.11).



Slika 3.11 Dugme za dodavanje novog proizvođača

Pritiskom na to dugme otvara se forma na novoj stranici ilustrirano slikom (Slika 3.12).

## Create new producer

Producer name:

Submit

Slika 3.12 Forma za stvaranje novog proizvođača

Za potrebe našeg demonstracijskog primjera stvorit ćemo novog proizvođača „Smreka d.o.o.“ koji će nakon klika na *Submit* biti vidljiv na popisu svih proizvođača vidljivo na slici (Slika 3.13). Ukoliko pokušamo unijeti praznu formu dobit ćemo jednaku obavijest kao na slici (Slika 3.9).

Picea		Home	Admin Panel
List of Producers			
Producer Name	Action		
Zeleni rasadnik Cerovac	<a href="#">Update</a>	<a href="#">Delete</a>	
Prvča Požega	<a href="#">Update</a>	<a href="#">Delete</a>	
Botanički vrt Zagreb	<a href="#">Update</a>	<a href="#">Delete</a>	
Smreka d.o.o.	<a href="#">Update</a>	<a href="#">Delete</a>	

Slika 3.13 Prikaz popisa proizvođača nakon dodavanja novog podatka

### 3.4. Upravljanje sortama

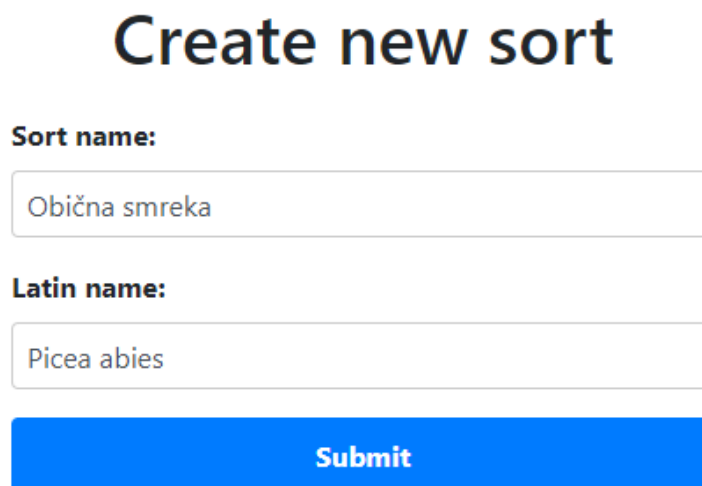
Ukoliko na admin panelu odaberemo opciju *Sorts* biti ćemo prebačeni na stranicu s popisom sorti i informacijama o njima vidljivo na slici (Slika 3.14). Ovaj popis ima jednake funkcionalnosti i kao popis proizvođača. Ovdje ćemo dodati novu sortu(srebreu smreku) kako bismo pokazali daljnje funkcionalnosti aplikacije.

Picea		Home	Admin Panel
List of Sorts			
Sort	Latin Name	Action	
Srebreu smreka	Picea Pungens	<a href="#">Update</a>	<a href="#">Delete</a>
Tulipan	Tulipa	<a href="#">Update</a>	<a href="#">Delete</a>
Beavertail cactus	Opuntia basilaris	<a href="#">Update</a>	<a href="#">Delete</a>

Create new sort

Slika 3.14 Prikaz evidencije sorti

Nakon klika na *Create new sort* i ispunjavanja podataka forma za unos na toj stranici izgledat će ovako:



**Create new sort**

**Sort name:**

Obična smreka

**Latin name:**

Picea abies

**Submit**

Slika 3.15 Prikaz forme za unos nove sorte

Klikom na *Submit* unosimo novu sortu u bazu podataka te smo spremni za nastavak evidentiranja našeg proizvoda.

### 3.5. Evidentiranje sijanja

Evidenciji sijanja pristupamo klikom na *Sowings* na izborniku spomenutom u [Admin panel](#). Web stranica evidencije sijanja sastoji se od tabličnih podataka koji sadržavaju podatke o sijanju poput godine, sorte, proizvođača sjemena i lokacije na kojoj je sorta posijana. Prikaz ove stranice nalazi se na slici (Slika 3.16).



Year	Sort Name	Producer Name	Location	Action
1999	Srebrna smreka	Prvča Požega	Dervišaga	<a href="#">Update</a> <a href="#">Delete</a>
2021	Beavertail cactus	Zeleni rasadnik Cerovac	Lakušija	<a href="#">Update</a> <a href="#">Delete</a>
2013	Srebrna smreka	Zeleni rasadnik Cerovac	Požega	<a href="#">Update</a> <a href="#">Delete</a>
1999	Tulipan	Botanički vrt Zagreb	Zagreb	<a href="#">Update</a> <a href="#">Delete</a>

Slika 3.16 Prikaz evidencije sijanja

Ove podatke je moguće uređivati i brisati po želji pritiskom na *Update* odnosno *Delete*. Za potrebe demonstracije stvorit ćemo novi unos sijanja običnih smreki.

## Create new sowing

**Producer name:**

Prvča Požega

**Sort name:**

Obična smreka

Srebrna smreka

Tulipan

Obična smreka

Beavertail cactusa

**Location:**

**Submit**

Slika 3.17 Evidentiranje novog sijanja

Tijekom unosa nove evidencije sijanja koristimo podatke iz drugih kolekcija iz baze podataka, stoga te vrijednosti biramo putem padajućeg izbornika kako je prikazano na slici (Slika 3.17). Nakon unosa podataka u formu i klikom na *Submit* prebačeni smo na stranicu evidencija sijanja gdje je vidljiva novonastala promjena:

Picea

HomeAdmin Panel

List of Sowings

Year	Sort Name	Producer Name	Location	Action
2010	Obična smreka	Prvča Požega	Požega	<div>UpdateDelete</div>
1999	Srebrna smreka	Prvča Požega	Dervišaga	<div>UpdateDelete</div>

Slika 3.18 Popis sijanja nakon promjene

### 3.6. Evidencija sadnje

Nakon određenog vremena rasta i razvoja koje ovisi o svakoj pojedinoj biljci, potrebno ih je presaditi i pružiti im više prostora za nastavak njihovog razvoja. Također postoji opcija kupnje sadnica od proizvođača ukoliko smatramo da je to sigurnija opcija. Evidencija sadnji sadržava popis svih dosadašnjih unosa ilustrirano na slici (Slika 3.19).

Picea						Home	Admin Panel
List of Plantings							
Producer Name	Sort Name	Amount	Year	Location	Action		
Zeleni rasadnik Cerovac	Beavertail cactus	223	1999	Zagreb	Update	Delete	
Zeleni rasadnik Cerovac	Tulipan	200	2007	Dervišaga	Update	Delete	
Zeleni rasadnik Cerovac	Srebrena smreka	22	2007	Požega	Update	Delete	
Botanički vrt Zagreb	Tulipan	51	2007	Dervišaga	Update	Delete	
Create new planting							

Slika 3.19 Evidencija svih sadnji

Sve ove podatke je moguće mijenjati, dodavati i brisati. Evidentirajmo sada presađivanje naših običnih smreki pritiskom na *Create new planting*.

Nakon klika na gumb pristupiti ćemo stranici sa slike (Slika 3.20).

**Create new planting**

**Producer name:**

Zeleni rasadnik Cerovac

**Sort name:**

Obična smreka

**Amount:**

300

**Year:**

2013

**Location:**

Lakušija

**Submit**

Slika 3.20 Forma za stvaranje novog unosa sadnje

Ukoliko ostavimo prazno polje aplikacija vraća upozorenje kao i na prethodnim formama.

### 3.7. Evidencija sječe

Nakon što naše biljke narastu do poželjne veličine potrebno ih je pripremiti za prodaju. Ova funkcionalnost modelira informacije o tome koje biljke smo posjekli, količinu i kojeg dana se je to dogodilo. Datum sječe je vrlo bitna stavka koja zanima mnoge mušterije. Prikaz tih informacija vidljiv je na slici (Slika 3.21).



Picea				Home	Admin Panel
List of Harvests					
Sort Name	Amount	Date	Action		
Srebrna smreka	200	2023-05-19	Update	Delete	
Tulipan	21	2023-05-04	Update	Delete	
Create new harvest					

Slika 3.21 Evidencija sječa

Evidentirajmo sada sječ u običnih smreka. Novina u formu za evidenciju smreka je odabir datuma. Kada kliknemo na polje za unos datuma iskočit će nam interaktivni izbornik sastavljen od kalendara. Pritiskom na željeni dan u mjesecu pohranjujemo taj podatak u željeno mjesto unutar forme. Prikaz ovaj akcije je ilustriran na slici (Slika 3.22).

## Create new Harvest

Sort name:

Obična smreka

Amount:

200

Date:

06.06.2023.

lipanj, 2023

po	ut	sr	če	pe	su	ne
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Izbriši

Danas

Slika 3.22 Stvaranje novog podataka o sadnji

### 3.8. Uređivanje cjenika

Potražnja i cijena biljki značajno se mijenja kroz vrijeme te ovisi o trenutnoj sezoni stoga proizlazi potreba za funkcionalnosti koja će ponuditi modeliranje cijene ovisne o takvim okolnostima. Osim sezona na cijenu biljke ovisi i njeno trenutno stanje u razvoju što se može pratiti njenom veličinom. Ovim informacijama pristupamo ukoliko na [Admin panelu](#) pritisnemo na opciju *Pricings*. Prikaz web stranice s ovom funkcionalnosti je na slici (Slika 3.23).

Picea

HomeAdmin Panel

List of Pricings

Sort Name	Price	Size	Season	Action
Tulipan	10€	10cm	Proljeće	<div>UpdateDelete</div>
Srebrna smreka	25€	2m	Zima	<div>UpdateDelete</div>

Slika 3.23 Prikaz svih cjenika

Izmjena i dodavanje podataka funkcionira kao i na web stranicama spomenutim prije. Dodajmo sada podatke za naš novi proizvod ispunjavanjem forme sa slike (Slika 3.24).

### Create new Pricing

Size:

2m+

Sort name:

Obična smreka

Price:

10€

Season:




Zima

Submit

Slika 3.24 Stvaranje novog cjenika


### 3.9. Upravljanje proizvodima

Nakon što smo popunili sve prethodne podatke možemo kreirati konačni proizvod spreman za prodaju. Web stranica koja modelira funkcionalnosti na proizvodima prikazana je na slici (Slika 3.25).

Picea								
Home Admin Panel								
List of Products								
Image	Sort Name	Year of Sowing	Year of Planting	Harvested On	Price	Description	In Pot	Actions
	Srebrna smreka	1999	2007	2023-05-19	25€	Srebrna smreka s jako oštrim iglicama	Product is not in pot	<a href="#">Update</a> <a href="#">Delete</a>
	Tulipan	1999	2007	2023-05-04	10€		Product is in pot	<a href="#">Update</a> <a href="#">Delete</a>
	Srebrna smreka	1999	2007	2023-05-19	25€		Product is in pot	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">Create new product</a>								

Slika 3.25 Pregled svih postojećih proizvoda

Svaki proizvod sadržava informacije o njegovoj sorti, godini sijanja i sadnje, datum sječe njegovu cijenu, opis i informaciju o tome nalazi li se proizvod u posudi. Ako neki od tih podataka nije dostupan, na njegovom mjestu pisat će skraćenica „N/A“ (not available). Primjer te funkcionalnosti ilustriran je na slici (Slika 3.26).

	Beavertail cactusa	2021	1999	N/A	20€	Ljubičasti kaktus	Product is in pot
---	--------------------	------	------	-----	-----	-------------------	-------------------

Slika 3.26 Primjer praznog atributa

Proces stvaranja novog proizvoda malo je drugačiji od stvaranja instanci drugih podataka. Pritiskom na dugme *Create new product* učitava se padajući izbornik koji sadržava sve sorte iz baze podataka. Izbornik je prikazan na slici (Slika 3.27).

## Choose sort of Product

Sort name:

Obična smreka

Submit

Slika 3.27 Izbornik za biranje sorte

Ovaj međukorak služi za pripremu svih podataka povezanih uz odabranu sortu. Nakon odabira željene sorte web aplikacija nas prebacuje na sljedeću formu za ispunjavanje ostalih informacija o tome proizvodu. Ako sorta nema evidentirane informacije o sijanju, sadnji, sječi ili cijeni ti podatci neće moći biti ispunjeni. Primjer je ilustriran na slici (Slika 3.28).

### Create new Obična smreka

Sort name:

Obična smreka

Sowing year:

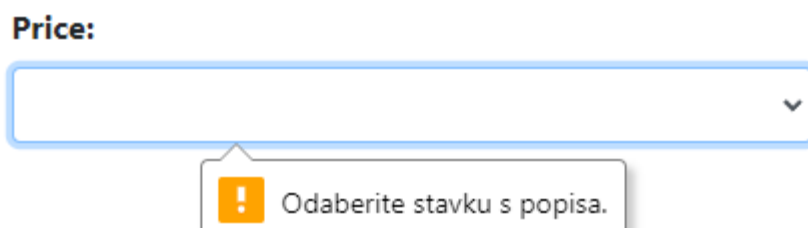
Planting year:

Harvest year:

Price:

Slika 3.28 Prikaz praznih padajućih listi

Također bitno je naglasiti da nema smisla dodati proizvod bez cijene stoga je to polje potrebno ispuniti inače je proizvod nemoguće stvoriti. Prikaz ove validacije funkcionalnosti nalazi se na slici (Slika 3.29).



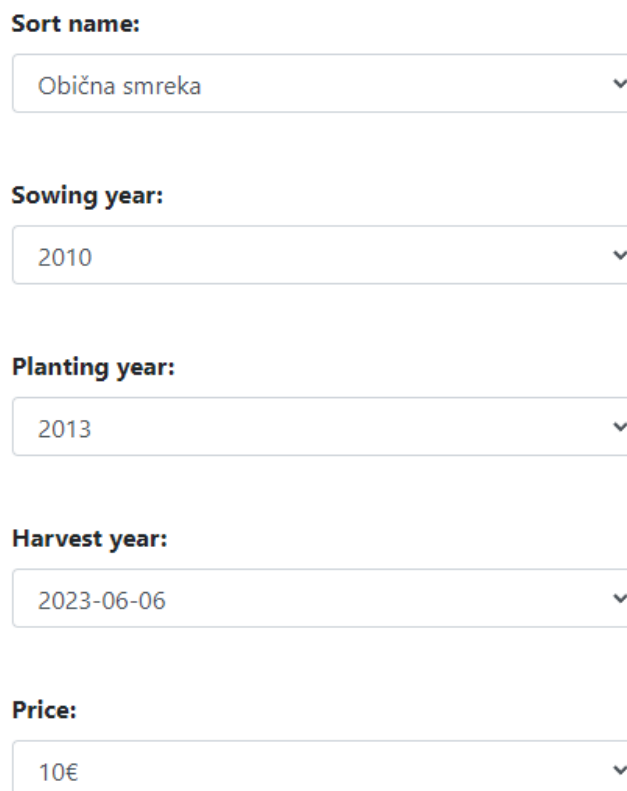
**Price:**

Odaberite stavku s popisa.

Slika 3.29 Validacija podatka o cijeni

Ispunjavanje forme za stvaranje novog proizvoda prikazano je na slikama (Slika 3.30) i (Slika 3.31).

## Create new Obična smreka



**Sort name:**

Obična smreka

**Sowing year:**

2010

**Planting year:**

2013

**Harvest year:**

2023-06-06

**Price:**

10€

Slika 3.30 Prvi dio unosa novog proizvoda

**Price:**

10€

**Size:**

220cm

**Description:**

Lijepa obična smreka

**Picture URL:**

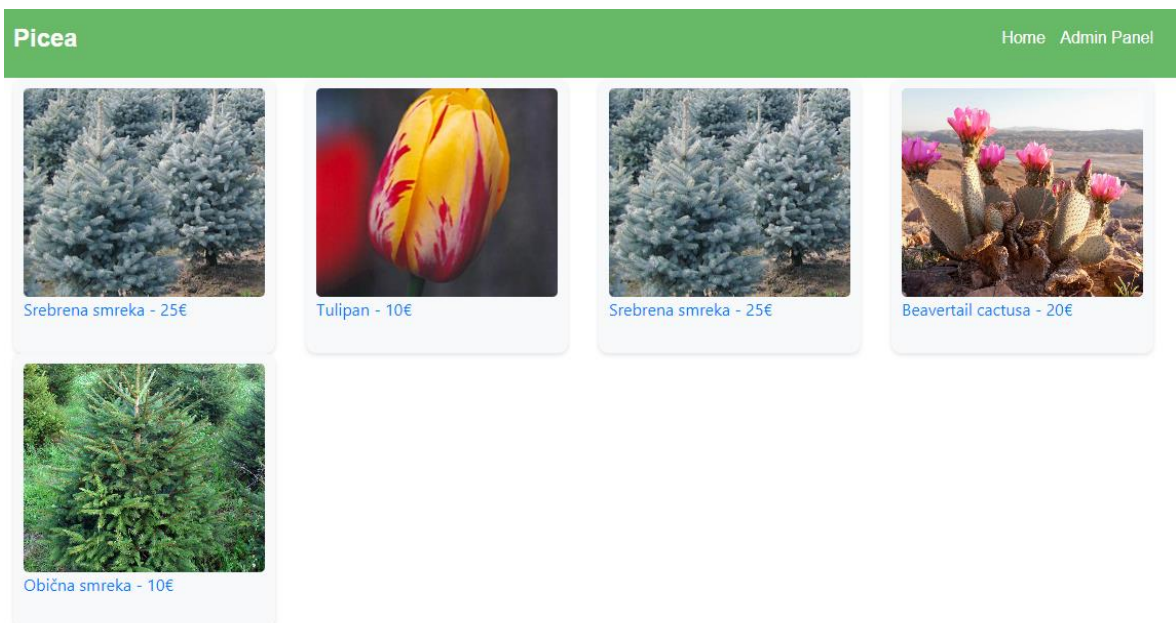
https://bozicna-drvca-jambrisak.com/images/smreka-1

**In Pot:** ☐

**Submit**

Slika 3.31 Drugi dio unosa novog proizvoda

Nakon pritiska na dugme *Submit* naš proizvod je stvoren i dodan u bazu podataka. Također proizvod je automatski dodan i na početnu stranicu web aplikacije gdje je spreman za rezervaciju. Ova funkcionalnost je ilustrirana na slici (Slika 3.32).



Slika 3.32 Prikaz novog proizvoda na početnoj stranici

### 3.10. Upravljanje rezervacijama proizvoda

Pogreške tijekom rezerviranja su očekivane iz čega proizlazi potreba za proizvoljnim upravljanjem rezervacijama poput uređivanja, brisanja i dodavanja novih. Pregled postojećih rezervacija prikazan je na slici (Slika 3.33).

Picea						Home	Admin Panel
List of Reservations							
Product ID	Name	Surname	Phone Number	Info	Action		
ihgG2nb79Xt32GWTYyw0	Jura	Starčević	12345667			Update	Delete

Create new reservation

Slika 3.33 Prikaz svih rezervacija

Svaka rezervacija sadrži identifikator proizvoda, ime i prezime klijenta te njegov broj mobitela. Opcionalno može sadržavati i neku dodatnu napomenu. Recimo da je klijent zaboravio dodati napomenu. Postupak izmejene će biti klik na dugme *Update* koje će otvoriti sljedeću formu na (Slika 3.34).

Prethodno uneseni podatci će već biti uneseni te mi ćemo samo dodati novu napomenu. Pritiskom na dugme *Save* spremamo promjenu i vraćamo se na web stranicu s rezervacija. Promjena je ilustrirana na slici (Slika 3.35).

## Edit Reservation

Name:

Jura

Surname:

Starčević

Phone Number:

12345667

Info:

napomena

Save

Slika 3.34 Uređivanje podataka o rezervaciji

Picea						Home	Admin Panel
List of Reservations							
Product ID	Name	Surname	Phone Number	Info	Action		
ihgG2nb79Xt32GWTYyw0	Jura	Starčević	12345667	napomena	Update	Delete	

Slika 3.35 Prikaz rezervacija nakon promjena







### 3.11. Izvješće o proizvodima

Važan aspekt prodaje je analiza prodaje kako bismo dobili više informacija o tome što više interesira naše kupce, a što manje. Stoga web aplikacija nudi pregled svih proizvoda s raznim kontekstima. Sljedeća funkcionalnost je ostvarena kao tablični prikaz svih proizvoda i informacijama o njima. Redovi koji sadržavaju već rezervirane proizvode označeni su svjetlo zelenom bojom. Slika (Slika 3.36) ilustrira ovu funkcionalnost.

**Picea**Home Admin Panel

## Product Report

☐ Filter by Not Reserved ☐ Filter by Reserved ☐ Filter by No Pot ☐ Filter by In Pot

Sort Name	Sowing Info	Planting Info	Harvest Info	Price Info	Product Size	Description	State of product	Picture
Srebrna smreka	1999	2007	2023-05-19	25€	2m	Srebrna smreka s jako ostrim iglicama	Product is not in pot	
Tulipan	1999	2007	2023-05-04	10€	10cm		Product is in pot	
Srebrna smreka	1999	2007	2023-05-19	25€	100Cm		Product is in pot	
Obična smreka	2010	2013	2023-06-06	10€	220cm	Lijepa obična smreka	Product is not in pot	

Slika 3.36 Pregled izvještaja svih proizvoda

Također ovaj izvještaj pruža mogućnost filtracije popisa proizvoda. Filteri su ostvareni uz pomoć JavaScripta stoga se osvježavanje popisa događa isti trenutak bez

potrebe za komunikaciju sa poslužiteljem. Slika (Slika 3.37) prikazuje filtraciju popisa ukoliko želimo vidjeti sve nerezervirane proizvode.

## Product Report

☒ Filter by Not Reserved ☐ Filter by Reserved ☐ Filter by No Pot ☐ Filter by In Pot


Sort Name	Sowing Info	Planting Info	Harvest Info	Price Info	Product Size	Description	State of product	Picture
Srebrna smreka	1999	2007	2023-05-19	25€	2m	Srebrna smreka s jako ostrim iglicama	Product is not in pot	
Tulipan	1999	2007	2023-05-04	10€	10cm		Product is in pot	
Obična smreka	2010	2013	2023-06-06	10€	220cm	Lijepa obična smreka	Product is not in pot	

Slika 3.37 Prikaz svih nerezerviranih proizvoda

Izveštaj pruža i kombiniranje filtera što je ilustrirano slikom (Slika 3.38) ukoliko odaberemo sve nerezervirane proizvode koji se nalaze u posudi.

## Product Report

☒ Filter by Not Reserved ☐ Filter by Reserved ☐ Filter by No Pot ☒ Filter by In Pot

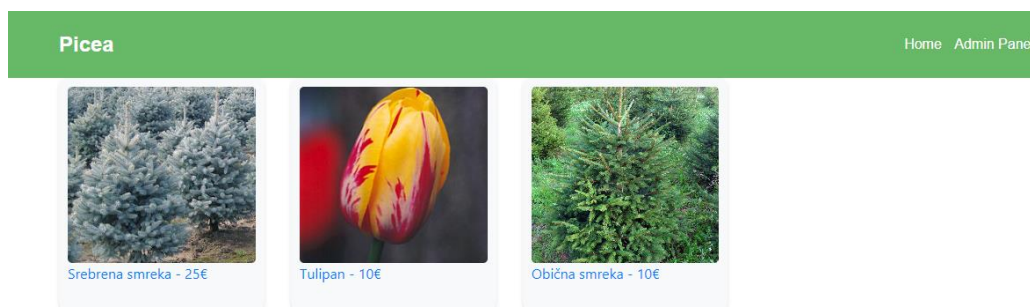
Sort Name	Sowing Info	Planting Info	Harvest Info	Price Info	Product Size	Description	State of product	Picture
Tulipan	1999	2007	2023-05-04	10€	10cm		Product is in pot	

Slika 3.38 Prikaz svih nerezerviranih proizvoda koji su u posudi

## 3.12. Pregled i rezervacija proizvoda

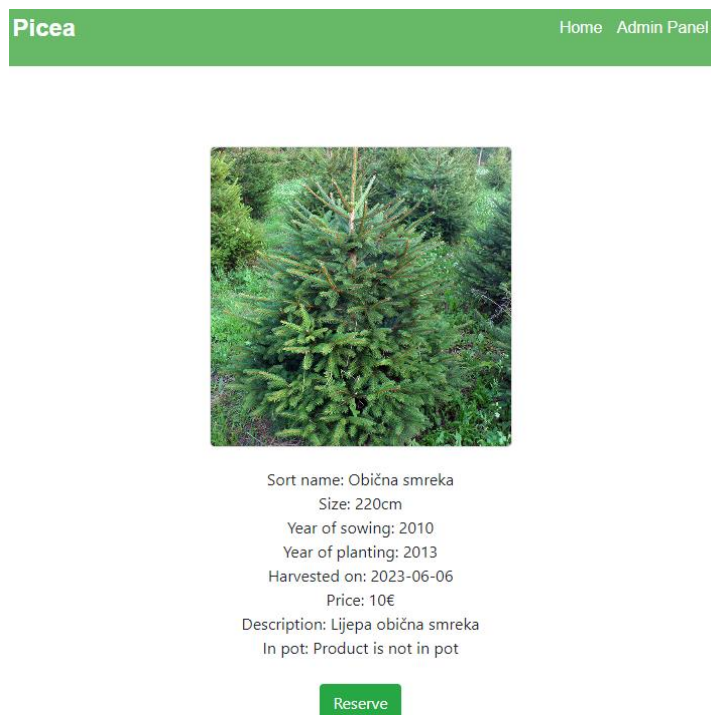
### 3.12.1. Pregled proizvoda

Početna stranica se sastoji od svih dostupnih proizvoda. Prikaz početne stranice vidljiv je na slici (Slika 3.39).



Slika 3.39 Prikaz svih dostupnih proizvoda

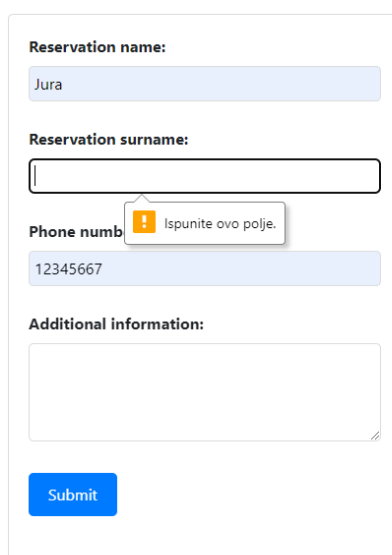
Pritiskom na sliku ili kratki opis proizvoda pristupamo stranici koja pruža više informacija o pojedinom proizvodu. Ilustracije te funkcionalnosti prikazana je na slici (Slika 3.40).



Slika 3.40 Prikaz detalja o proizvodu

### 3.12.2. Rezervacija proizvoda

Proces rezervacije proizvoda započinjemo pritiskom na dugme *Reserve* vidljivo na slici (Slika 3.40). Nakon klika na dugme otvara se nova forma u koju je potrebno unijeti podatke poput imena i prezimena naručitelja, broj mobitela i dodatne napomene ili želje. Ime, prezime te broj mobitela su obavezni podatci pa nije moguće stvoriti rezervaciju bez tih podataka. Forma upozorava korisnika ukoliko ti podatci nisu ispunjeni što je vidljivo na slici (Slika 3.41).



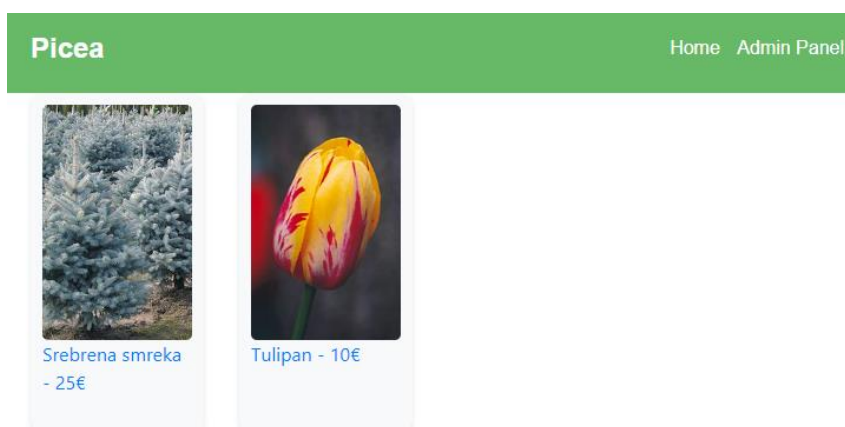
The screenshot shows a reservation form with the following fields and values:

- Reservation name:** Jura
- Reservation surname:** (empty)
- Phone number:** 12345667
- Additional information:** (empty text area)

A blue **Submit** button is at the bottom. A yellow tooltip with an exclamation mark icon and the text "Ispunite ovo polje." points to the empty surname field.

Slika 3.41 Validacija unosa podataka za rezervaciju

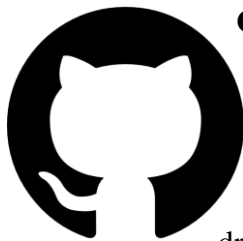
Klikom na dugme *Submit* stvara se rezervacija i preusmjereni smo na početnu stranicu gdje sada možemo primjetiti da proizvod koji smo rezervirali više nije prikazan. Promjena je vidljiva na slici (Slika 3.42).



Slika 3.42 Prikaz početne stranice nakon procesa rezervacije

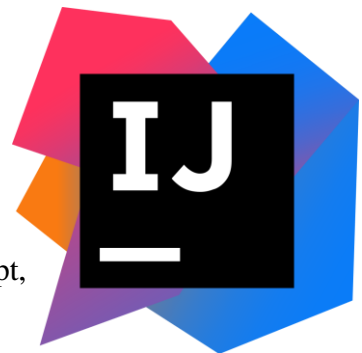
## 4. Korištene tehnologije

**Firebase** je platforma koju je razvio Google i koja pruža razne usluge i alate za razvoj mobilnih i web aplikacija. Firebase pruža set alata koji olakšavaju izradu, testiranje, upravljanje i skaliranje aplikacija, bez potrebe za održavanjem vlastitog infrastrukturnog okruženja (Google 2023).



**GitHub** je web platforma koja pruža usluge za upravljanje i suradnju na razvoju softvera koristeći Git, distribuirani sustav za verzioniranje koda. To je popularno odredište za programere i timove za upravljanje izvornim kodom njihovih projekata, praćenje promjena, suradnju s drugim programerima i objavljivanje njihovog koda (Github 2023).

**IntelliJ IDEA** je integrirano razvojno okruženje (IDE) koje je razvio JetBrains. Ono pruža programerima bogat skup alata i značajki za razvoj softvera u različitim programskim jezicima, uključujući Java, Kotlin, Scala, Groovy, JavaScript, TypeScript, PHP, Python, i mnoge druge (JetBrains 2023).



**Java** je visokorazinski, objektno orijentirani programski jezik koji je razvijen od strane Sun Microsystemsa (sada dio Oracle Corporation) u kasnim 1990-ima. Java je popularan programski jezik koji se koristi za razvoj različitih vrsta aplikacija, uključujući web aplikacije, mobilne aplikacije, desktop aplikacije, igre, serverske aplikacije i mnoge druge (Oracle 2023).



**Spring** je popularni okvir (framework) otvorenog koda za razvoj Java aplikacija. Razvijen je kako bi pojednostavio razvoj složenih aplikacija i pružio fleksibilnost, skalabilnost i lakoću održavanja (VMware 2023). Spring pruža brojne module i komponente koji olakšavaju razvoj, testiranje i upravljanje Java aplikacijama (VMware 2023).



**Thymeleaf** je templating framework za izradu dinamičkih web stranica koristeći Java i XML sintaksu. To je popularan alat u svijetu Java web razvoja, koji omogućava programerima da generiraju HTML, XML, CSV i druge formate datoteka koristeći podatke iz backend aplikacija (The Thymeleaf Team 2023).

**Apache Maven** je alat za upravljanje projektima koji se koristi u razvoju softvera. On pruža konzistentan način za upravljanje ovisnostima projekta, izgradnju softvera, testiranje i distribuciju (Apache 2023). Maven je vrlo popularan u svijetu Java razvoja, iako se može koristiti i za druge programske jezike (Apache 2023).



**Lucidchart** je alat za izradu dijagrama i vizualizacija koji se koristi za stvaranje profesionalnih dijagrama, flowchartova, organizacijskih shema, mrežnih dijagrama, ER dijagrama, UML dijagrama, i još mnogo toga. To je web bazirana platforma koja omogućava korisnicima da stvaraju, dijele i surađuju na dijagramima u stvarnom vremenu (Lucid Software 2023).

**HTML, CSS i JavaScript** su tri osnovne tehnologije koje se koriste u web razvoju za stvaranje i dizajniranje web stranica i web aplikacija. HTML je standardni označivački jezik koji se koristi za strukturiranje i prikazivanje sadržaja na

webu (Kolowich Cox 2021). On opisuje strukturu sadržaja, ali nema moć za interakciju ili stiliziranje. CSS je stilski jezik koji se koristi za stiliziranje i izgled web stranica. On definira izgled i prezentaciju elemenata definiranih u HTML-u (Kolowich Cox 2021). JavaScript je programski jezik koji se koristi za stvaranje interaktivnih elemenata na web stranicama. To je skriptni jezik koji se izvršava u pregledniku korisnika i omogućava dodavanje dinamičkog ponašanja i funkcionalnosti na web stranicu (Kolowich Cox 2021).



**Bootstrap** je popularan otvoreni CSS framework za brzi razvoj responzivnih web stranica i web aplikacija. To je skup preddefiniranih stilova, komponenti i skripti koje omogućavaju programerima da brzo i jednostavno stvaraju moderna i responzivna korisnička sučelja (Bootstrap team 2023).

**Microsoft Word** je popularan tekstualni procesor razvijen od strane tvrtke Microsoft. To je dio Microsoft Office paketa i pruža korisnicima širok spektar alata za stvaranje, uređivanje i oblikovanje tekstualnih dokumenata. Microsoft Word je dostupan za operacijske sustave Windows, macOS, iOS i Android (Microsoft 2023).



**Zotero** je besplatni softver za upravljanje referencama i alat za organizaciju istraživačkih materijala. On omogućuje korisnicima da jednostavno prikupljaju, organiziraju i citiraju bibliografske podatke iz različitih izvora, kao što su akademski članci, knjige, web stranice i drugi materijali (Corporation for Digital Scholarship 2023).

## Zaključak

Aplikacija omogućuje korisnicima pregled dostupnih proizvoda, informacija o proizvodima i mogućnost rezervacije. Također, programski sustav ima administratorski panel koji omogućuje upravljanje podacima.

Na početnoj stranici, korisnici mogu vidjeti postojeće dostupne proizvode i klikom na sliku ili opis proizvoda mogu otvoriti web stranicu s dodatnim informacijama i rezervacijom. Osim toga, na početnoj stranici nalaze se poveznice "Home" i "Admin Panel". Pritiskom na "Home" korisnici se vraćaju na početnu stranicu, dok pritiskom na "Admin Panel" prelaze na admin panel s opcijama za upravljanje podacima. Ako korisnici nisu prijavljeni, bit će preusmjereni na stranicu za prijavu.

Administratorski panel sadrži navigaciju i poveznice na posebne funkcionalnosti. Jedna od funkcionalnosti je upravljanje proizvođačima sjemena i sadnica. Na toj stranici korisnici mogu vidjeti popis postojećih proizvođača, a mogu izvršiti operaciju brisanja ili ažuriranja podataka proizvođača. Također, mogu dodati novog proizvođača putem forme. Upravljanje sortama je još jedna funkcionalnost koja omogućuje korisnicima pregled, dodavanje i ažuriranje podataka o sortama biljaka. Programski sustav omogućuje evidentiranje sijanja, sadnje i sječe biljaka. Korisnici mogu unijeti podatke o tim aktivnostima putem obrazaca, a zatim pregledavati, uređivati ili brisati te podatke. Također postoji mogućnost uređivanja cjenika biljaka, ovisno o sezoni i veličini. Korisnici mogu pregledavati, dodavati i mijenjati podatke o cjenicima.

Programski sustav omogućuje upravljanje proizvodima, gdje korisnici mogu stvarati nove proizvode, unoseći informacije o sorti, godini sijanja i sadnje, datumu sječe, cijeni, opisu i informaciji o posudi. Proizvodi se mogu rezervirati putem aplikacije, a rezervacije mogu biti uređivane, brisane ili dodavane.



# Literatura

Apache. 2023. 'Maven'. <https://maven.apache.org/> (June 6, 2023).

Bootstrap team. 2023. 'Bootstrap'. <https://getbootstrap.com/> (June 6, 2023).

Corporation for Digital Scholarship. 2023. 'Zotero'. <https://www.zotero.org/> (June 6, 2023).

Github. 2023. 'GitHub'. <https://github.com/> (June 6, 2023).

Google. 2023. 'Firebase'. <https://firebase.google.com> (June 6, 2023).

JetBrains. 2023. 'IntelliJ'. <https://www.jetbrains.com/idea/> (June 6, 2023).

Kolowich Cox, Lindsay. 2021. 'Coding for Web Design 101: How HTML, CSS, and JavaScript Work'. <https://blog.hubspot.com/marketing/web-design-html-css-javascript> (June 6, 2023).

Lucid Software. 2023. 'Lucidchart'. <https://www.lucidchart.com/> (June 6, 2023).

MDN contributors. 2023. 'MVC'. <https://developer.mozilla.org/en-US/docs/Glossary/MVC> (June 7, 2023).

Microsoft. 2023. 'Microsoft Word'. <https://www.microsoft.com/en-ww/microsoft-365> (June 6, 2023).

Oracle. 2023. 'Java'. <https://www.java.com/> (June 6, 2023).

Šegvić, Siniša, and Marko Čupić. 'Oblikovni Obrasci u Programiranju'. <https://www.fer.unizg.hr/predmet/ooup> (June 6, 2023).

The Thymeleaf Team. 2023. 'Thymeleaf'. <https://www.thymeleaf.org/> (June 6, 2023).

VMware. 2023. 'Spring'. <https://spring.io/> (June 6, 2023).

## Sažetak

**Naslov:** Aplikacija podrške uzgoju i prodaji biljaka

**Sažetak:** Ova aplikacija omogućuje korisnicima da pregledaju dostupne proizvode, dobiju informacije o njima te obave rezervacije. Također, aplikacija sadrži administratorski panel koji omogućuje upravljanje podacima. Na početnoj stranici korisnici mogu vidjeti dostupne proizvode te otvoriti web stranicu s dodatnim informacijama i rezervacijom klikom na sliku ili opis proizvoda. Navigacija na početnoj stranici uključuje poveznice "Home" i "Admin Panel", koje korisnike vraćaju na početnu stranicu odnosno prebacuju na administratorski panel. Neprijavljeni korisnici bit će preusmjereni na stranicu za prijavu. Administratorski panel omogućuje upravljanje podacima, uključujući proizvođače sjemena i sadnica te sorte biljaka. Korisnici mogu pregledavati, ažurirati i brisati informacije o proizvođačima te dodavati nove proizvođače putem posebne forme. Također, korisnici mogu upravljati podacima o sortama biljaka, uključujući pregled, dodavanje i ažuriranje. Programski sustav omogućuje evidentiranje aktivnosti sijanja, sadnje i sječe biljaka, s mogućnosti unosa i uređivanja tih podataka. Upravljanje cjenikom biljaka omogućuje korisnicima pregledavanje, dodavanje i mijenjanje podataka o cjenicima. Također, korisnici mogu stvarati nove proizvode unoseći informacije o sorti, godini sijanja i sadnje, datumu sječe, cijeni, opisu i informaciji o tome nalazili se u posudi. Proizvodi se mogu rezervirati putem aplikacije, a rezervacije mogu biti uređivane, brisane ili dodavane.

**Ključne riječi:** aplikacija, proizvodi, informacije, rezervacija, administratorski panel, proizvođači, sorte, sijanje, sadnja, sječa, cjenik, pregledavanje, dodavanje, uređivanje, brisanje.

# Summary

**Title:** Plant Cultivation and Sales Support Application

**Abstract:** This application allows users to browse available products, obtain information about them, and make reservations. Additionally, the application includes an administrative panel that enables data management. On the homepage, users can view the available products and click on an image or short product description to open a webpage with additional information and reservation options. The homepage also features links for "Home" and "Admin Panel." Clicking on "Home" takes users back to the homepage, while clicking on "Admin Panel" redirects them to the admin panel with data management options. Unauthorized users will be redirected to the login page. The admin panel provides functionalities such as managing seeds and plantings producers and plant sorts. Users can view, update, and delete information about producers and add new producers through a dedicated form. They can also manage data related to plant sorts, including browsing, adding, and updating. The software allows tracking activities such as sowing, planting, and harvesting plants, with the ability to enter and edit such data. Price list management for plants enables users to view, add, and modify pricing information. Additionally, users can create new products by entering details about the sort, sowing and planting year, harvest date, price, description, and information about product being in pot. Products can be reserved through the application, and reservations can be edited, deleted, or added.

**Keywords:** application, products, information, reservation, administrative panel, producers, sort, sowing, planting, harvesting, price list, browsing, adding, editing, deleting.