Web aplikacija za odabir kozmetičkih proizvoda

Gregurić, Lea

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

Stručni prijediplomski studij Računarstvo

WEB APLIKACIJA ZA ODABIR KOZMETIČKIH PROIZVODA

Završni rad

Lea Gregurić

Osijek, 2024.

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

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Lea Gregurić
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R 4448, 06.10.2022.
JMBAG:	0165081581
Mentor:	doc. dr. sc. Tomislav Galba
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Alfonzo Baumgartner
Član Povjerenstva 1:	doc. dr. sc. Tomislav Galba
Član Povjerenstva 2:	prof. dr. sc. Tomislav Keser
Naslov završnog rada:	Web aplikacija za odabir kozmetičkih proizvoda
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Potrebno je napraviti web aplikaciju koja će omogućiti korisnicima unos podataka na osnovu kojih će se dobiti lista kozmetičkih proizvoda koji najbolje odgovaraju tipu kože korisnika. Web aplikacija treba imati više korisničkih uloga (minimalno admin/korisnik). Potrebno je koristiti neke od dostupnih baza podataka. Napomena: tema rezervirana za Lea Gregurić

Datum ocjene pismenog dijela završnog rada od strane mentora:	17.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	30.09.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	30.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 30.09.2024.

Ime i prezime Pristupnika:	Lea Gregurić
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4448, 06.10.2022.
Turnitin podudaranje [%]:	12

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za odabir kozmetičkih proizvoda**

izrađen pod vodstvom mentora doc. dr. sc. Tomislav Galba

i sumentora

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 pristupnika:

SADRŽAJ

1. Uvod

Sve više se vodi briga o izgledu i zdravlju kože općenito. Na tržištu je dostupan velik broj kozmetičkih proizvoda čiji naziv često sugerira na djelovanje na određeni tip kože, međutim sastav nije pogodan za taj tip kože. Zbog navedenog postoji potreba za analizom sastava proizvoda, a tehnologija može znatno pomoći u tome.

Tema ovog završnog rada je web aplikacija za odabir kozmetičkih proizvoda. Cilj aplikacije je omogućiti korisnicima da dobiju preporučene proizvode za njihov tip kože na temelju sastava proizvoda. To se postiže usporedbom sastava pojedinog proizvoda sa sastojcima koji su preporučeni za određeni tip kože, odnosno sastojcima koji nisu preporučeni. Lakšem odabiru proizvoda također doprinose komentari drugih korisnika koji su isprobali proizvod, što je također omogućeno u ovoj aplikaciji.

U drugom poglavlju rada predstavljena su postojeća rješenja. U trećem poglavlju opisuju se tehnologije korištene pri izradi web aplikacije, a to su HTML, SCSS, Vue.js, Node.js, MySQL . U četvrtom poglavlju opisan je proces razvoja web aplikacije što uključuje kreiranje baze podataka, *backend* i *frontend* dio. U petom poglavlju je prikazano korištenje web aplikacije.

1.1. Zadatak završnog rada

Potrebno je napraviti web aplikaciju koja će omogućiti korisnicima unos podataka na osnovu kojih će se dobiti lista kozmetičkih proizvoda koji najbolje odgovaraju tipu kože korisnika. Web aplikacija treba imati više korisničkih uloga (minimalno admin/korisnik). Potrebno je koristiti neke od dostupnih baza podataka.

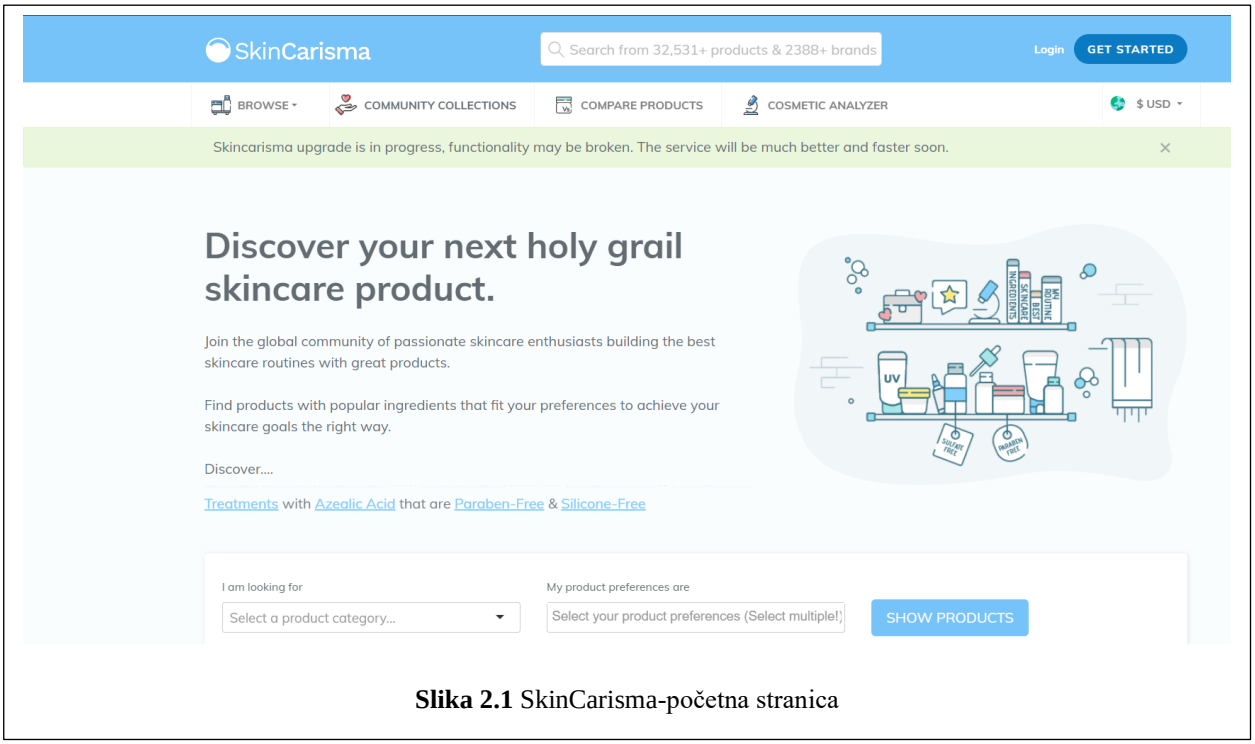
2. Postojeća rješenja

U ovom poglavlju predstavljena su neka od trenutno postojećih rješenja za odabir kozmetičkih proizvoda na temelju njihovog sastava, a to su:

- SkinCarisma
- INCIdecoder
- SkinSAFE.

2.1. SkinCarisma

SkinCarisma je web aplikacija koja korisnicima omogućuje pretraživanje i analizu sastava kozmetičkih proizvoda. Korisnici za željeni proizvod mogu pronaći sastav, funkciju, potencijalne benefite i rizike za kožu. Za svaki proizvod je moguće postaviti komentar. Na slici **2.1** se može vidjeti naslovna stranica navedene web aplikacije [1].

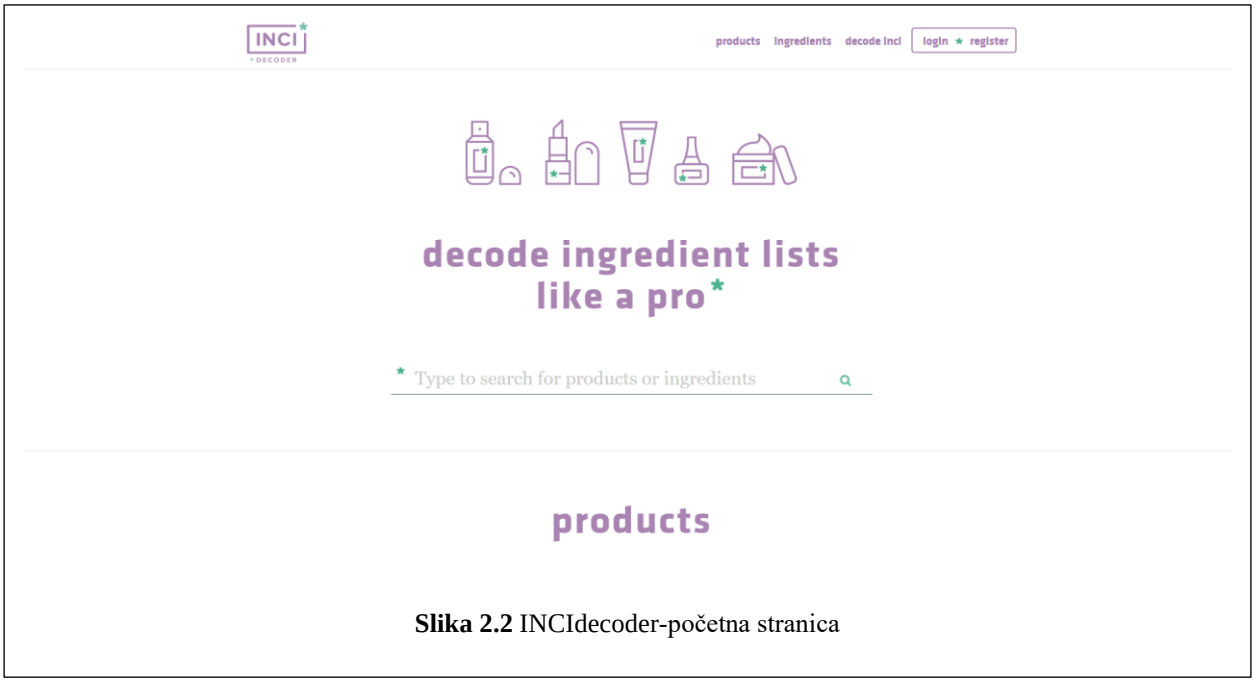


Slika 2.1 SkinCarisma-početna stranica

2.2. INCIdecoder

INCIdecoder je jedna od najpopularnijih web aplikacija za analizu kozmetičkih proizvoda. Pruža mogućnost korisnicima da unesu sastav proizvoda i dobiju informaciju o sastojcima. Također

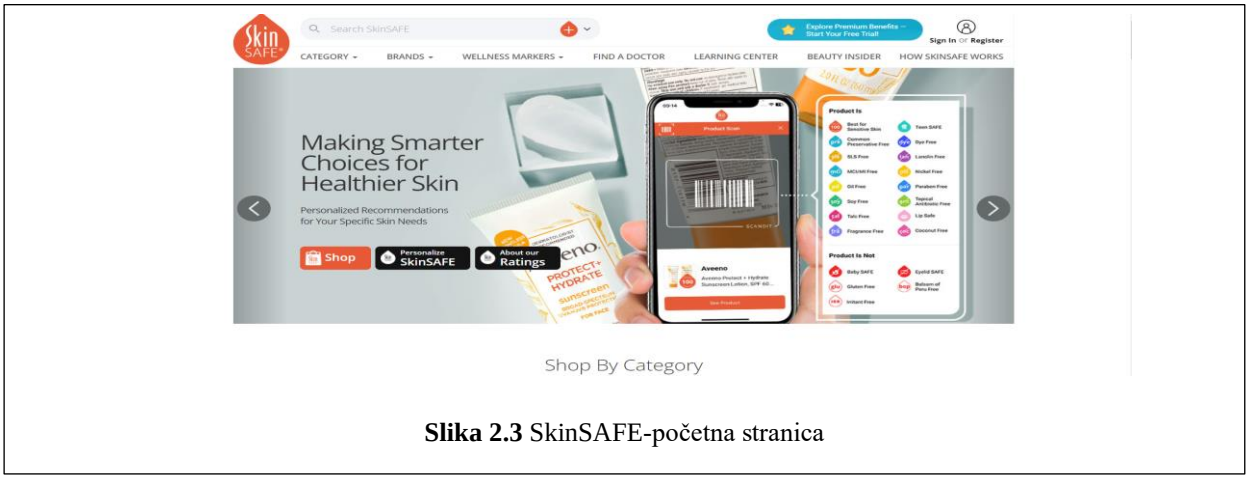
moguće je pretraživanje proizvoda po imenu, imenu proizvođača ili sastavu. Aplikacija sadrži veliki broj kozmetičkih proizvoda iz cijelog svijeta. Na slici **2.2** prikazana je početna stranica [2].



Slika 2.2 INCIdecoder-početna stranica

2.3. SkinSAFE

SkinSAFE je web aplikacija i mobilna platforma koja omogućuje prepoznavanje potencijalno štetnih sastojaka. Svaki proizvod ima ocjenu sigurnosti na temelju prisutnosti štetnih sastojaka. Proizvodi se rangiraju na temelju toga koliko su slobodni od alergena. Korisnici mogu pretraživati proizvode ili skenirati bar *kod* proizvoda. Slika **2.3** prikazuje početku stranicu web aplikacije SkinSAFE [3].



Slika 2.3 SkinSAFE-početna stranica

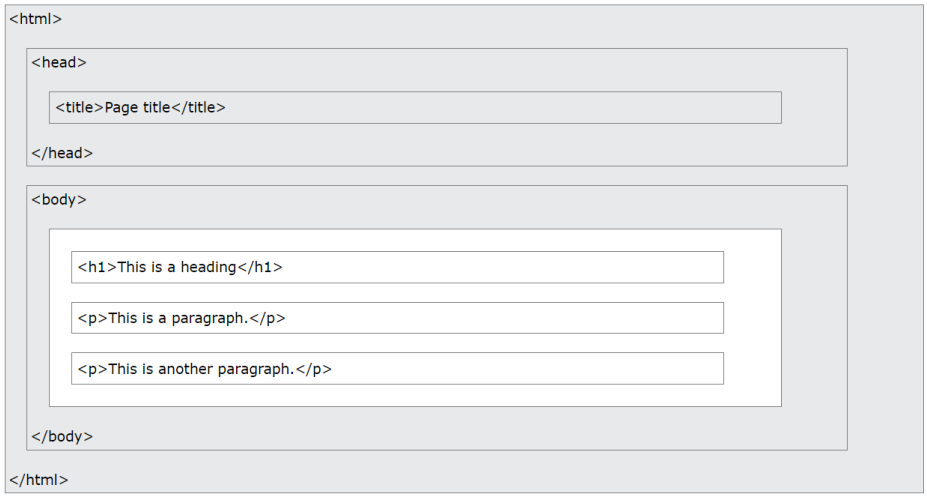
3. Tehnologije korištene prilikom izrade web aplikacije

Za izradu web aplikacije potrebno je korištenje brojnih tehnologija. U izradi ove web aplikacije korišteni su:

- HTML
- SCSS
- Vue.js
- Node.js
- MySQL

3.1. HTML

HTML (engl. *Hyper Text Markup Language*) je deskriptivni jezik za web stranice. Opisuje kako je web stranica strukturirana te se sastoji od niza elemenata. Elementi govore pregledniku kako prikazati sadržaj [4]. Element je odijeljen od drugog teksta oznakom. Neke od oznaka su <head>, <body>, <header>, <footer>, <article>, <section>, <p>, , <img>, <ul>, <ol>, <li> [5]. Na slici **3.1** prikazana je struktura HTML-a [4].



Slika 3.1 HTML struktura

3.2. SCSS

SCSS je kratica za "*Sassy CSS*", što je CSS predprocesorski skriptni jezik koji standardnom CSS-u dodaje dodatne značajke i mogućnosti. SCSS je prvi razvio Hampton Catlin 2006. godine. Jezik je dizajniran za rješavanje ograničenja i izazova pisanja složenog i održivog CSS-a. SCSS je

također poznat kao SASS (engl. *Syntactically Awesome Stylesheets*) [6]. Sintaksa SCSS-a vrlo je slična CSS-u, ali dopušta upotrebu varijabli, ugnježđivanja, miješanja i drugih programskih konstrukcija [7].

3.3. Vue.js

Vue je razvojni okvir za razvoj sučelja temeljen na JavaScriptu. Izgrađen je na standardnom HTML-u, CSS-u i JavaScriptu i pruža intuitivne i integrirane softverske komponente Većina Vue projekata stvara Vue komponente koristeći format datoteke sličan HTML-u. On sažima logiku komponente (JavaScript), predloške (HTML) i stil (CSS) u jednoj datoteci [8]. Na slici **3.3** je prikazan primjer Vue komponente korištene u ovom projektu.

```
<script setup>
import { useUser, clearUser } from '../../store/auth.js';
import { useRouter } from 'vue-router';

const router = useRouter();
const { user } = useUser();

const logout = () => {
  clearUser();
  router.push('/login');
};
</script>

<template>
    <header class="header-container">
      <RouterLink to="/" class="header-action-btn">Home</RouterLink>
      <RouterLink to="/products" class="header-action-btn">Products</RouterLink>

      <template v-if="user">
        <span class="header-action-btn">Hello, {{ user.username }}!</span>

        <RouterLink v-if="user.role === 'admin'" to="/admin" class="header-action-btn">Admin Dashboard</RouterLink>

        <button @click="logout()" class="header-action-btn">Logout</button>
      </template>

      <template v-else>
        <RouterLink to="/login" class="header-action-btn">LOGIN</RouterLink>
        <RouterLink to="/register" class="header-action-btn">REGISTER</RouterLink>
      </template>
    </header>
</template>
```

Slika 3.3 *Header* komponenta

3.4. Node.js

Node.js je besplatno JavaScript programsko okruženje otvorenog *koda* za više platformi koje programerima omogućuje stvaranje web aplikacija, alata i skripti na strani poslužitelja [9]. Memorijski je vrlo učinkovit. Node.js može generirati dinamički sadržaj stranice, kreirati, otvarati, čitati i pisati, obrisati, zatvarati datoteke na poslužitelju, prikupljati podatke obrazaca, dodavati,

brisati i mijenjati podatke baze podataka. Datoteke Node.js sadrže zadatke koji se izvršavaju na određene događaje. Node.js datoteke imaju nastavak ".js"[10]. Express je također korišten za razvoj ovog programa. Express je fleksibilan okvir web aplikacije Node.js koji pruža skup značajki za web i mobilne aplikacije. [11].

3.5. MySQL

MySQL je najpopularnija baza podataka otvorenog *koda*. Budući da je MySQL otvorenog *koda*, uključuje mnoge značajke razvijene u bliskoj suradnji s korisnicima. Baza podataka je osnovna pohrana podataka za sve aplikacije. Baze podataka pohranjuju podatke u zasebne tablice. Baze podataka su organizirane u fizičke datoteke, što povećava brzinu. Logički model uključuje objekte kao što su podatkovne tablice, pogledi, retci i stupci i pruža jednostavnu metodu programiranja. MySQL je idealan i za male i za velike aplikacije. Vrlo je brz, skalabilan i pouzdan. MySQL razvija, distribuira i podržava Oracle Corporation. Koriste ga velike web stranice kao što su Facebook, Twitter i YouTube [13].

4. Implementacija programskog rješenja

Proces razvoja web aplikacija sastoji se od kreiranja baze podataka, razvoja *backenda* te razvoja *frontenda*. U ovom poglavlju opisuje se svaki od navedenih koraka.

4.1. Baza podataka

Kako bi se podaci mogli pohraniti potrebna je baza podataka. Za izradu ove web aplikacije je korišten MySQL. Izrađena je baza podatka pod nazivom „skinprotect“ koja se sastoji od tablica „comments“, „products“, „skin_type_ingredients“ te „user“.

4.1.1. Tablica „user“

Tablica „user“ sadrži podatke o korisnicima. Sastoji se od pet stupaca, a to su „ID“, „email“, „username“, „password“ i „role“. Struktura tablice prikazana je na slici **4.1**.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ID	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
2	email	varchar(150)	utf8mb4_general_ci		No	*None*			Change Drop More
3	username	varchar(20)	utf8mb4_general_ci		No	*None*			Change Drop More
4	password	varchar(100)	utf8mb4_general_ci		No	*None*			Change Drop More
5	role	varchar(30)	utf8mb4_general_ci		No	*None*			Change Drop More

Slika 4.1 Struktura tablice "user"

ID	email	username	password	role
12	admin@admin.com	admin	admin	admin
11	user3@gmail.com	user3	$2b$10$BdwE/5fxyQg0UoeeGfhKyO/NABaipylMh2k7e5/57Hw...	user
10	user2@gmail.com	user2	$2b$10$Nzt0/7Oyt3DuxkdvmcKSdO0nZLyX4F4wpRgegB4So2e...	user
9	user1@gmail.com	user1	$2b$10$U/FarSHviBbEM9oQf7HAze1Rm4S7aA6kMxYyS/680Fh...	user
7	lea123@gmail.com	lea123	$2b$10$EpMgDtCRNcDD/4L9l7n8/u3xYwXneXLtkM/yKgyf/48...	user

Slika 4.2 Podaci u tablici "user"

Stupac „ID“ je primarni ključ i označava identifikacijski broj korisnika. Stupac „email“ sadrži mail adresu korisnika, „username“ korisničko ime, „password“ zaporku te „role“ korisničku ulogu (korisnik/admin). Na slici **4.2** je prikazan primjer podataka unutar tablice .

4.1.2. Tablica „skin_type_ingredients“

Tablica „skin_type_ingredients“ sadrži sastojke koji su poželjni ili nepoželjni u kozmetičkim proizvodima za određeni tip kože. Sastoji se od stupaca „id“ koji je primarni ključ i označava identifikacijski broj sastojka, „skin_type“ koji govori za koji tip kože je sastojak poželjan ili nepoželjan, „ingredient“ koji govori o kojem sastojku je riječ te „is_desirable“ što označava da li je sastojka poželjan ili nepoželjan, gdje 0 označava da je nepoželjan, a 1 poželjan. Struktura tablice vidljiva je na slici **4.3**, dok se na slici **4.4** može vidjeti primjer sadržaja tablice.

Name	Type	Collation	Attributes	Null	Default	Comments	Extra
id	int(10)		UNSIGNED	No	*None*		AUTO_INCREMENT
skin_type	varchar(255)	utf8mb4_general_ci		No	*None*		
ingredient	varchar(255)	utf8mb4_general_ci		No	*None*		
is_desirable	tinyint(1)			No	*None*		

Slika 4.3 Struktura tablice "skin_type_ingredients"

id	skin_type	ingredient	is_desirable
1	oily	NIACINAMIDE	1
2	oily	SALICYLID ACID	1
4	oily	BENZOYL PEROXIDE	1
5	oily	HYALURONIC ACID	1
6	oily	GLYCOLIC ACID	1
7	oily	ZINC OXIDE	1
8	oily	MINERAL OIL	0
9	oily	LANOLIN	0
10	oily	PARAFFIN	0
11	oily	ALCOHOL	0

Slika 4.4 Podaci u tablici "skin_type_ingredients"

4.1.3. Tablica „products“

Tablica „products“ sadrži informacije o proizvodima. Sastoji se od stupca „ID“, „name“, „ingredients“ i „image“. Struktura se može vidjeti na slici **4.5**.

Name	Type	Collation	Attributes	Null	Default	Comments	Extra
ID	int(11)			No	*None*		AUTO_INCREMENT
name	varchar(50)	utf8mb4_general_ci		No	*None*		
ingredients	varchar(5000)	utf8mb4_bin		No	*None*		
image	varchar(300)	utf8mb4_general_ci		No	*None*		

Slika 4.5 Struktura tablice "products"

Stupac „ID“ je primarni ključ i označava identifikacijski broj proizvoda, „name“ sadrži ime proizvoda, „ingredients“ sadrži listu sastojaka, a „image“ putanju do slike proizvoda. Na slici **4.6** je prikazan sadržaj tablice.

ID	name	ingredients ▾ 1	image
8	Madagascar Centella Hyalu-Cica Water-Fit Sun Serum	WATER, DIBUTYL ADIPATE, PROPANEDIOL, DIETHYLAMINO ...	../assets/img/skin1004-spf.jpg
1	Balea serum za lice s niacinamidom	AQUA, NIACINAMIDE, GLYCERIN, PENTYLENE GLYCOL, ALO...	../assets/img/balea-niacinamid-serum.jpg
4	Dnevna krema s niacinamidom, SPF 30, 50 ml	AQUA, NIACINAMIDE, DIBUTYL ADIPATE, GLYCERIN, ETHY...	../assets/img/balea-niacinamid-krema.jpg
9	Noćna krema za lice – niacinamid	AQUA, NIACINAMIDE, C12-15 ALKYL BENZOATE, GLYCERIN...	../assets/img/balea-niacinamid-krema-nocna.jpg
2	Balea Ultra Sensitive gel za pranje lica	AQUA, GLYCERIN, PENTYLENE, GLYCOL, COCAMIDOPROPYL ...	../assets/img/balea-gel.jpg

Slika 4.6 Podaci u tablici "products"

4.1.4. Tablica „comments“

Tablica „comments“ sadrži komentare proizvoda. Sastoji se od stupaca „id“, „product_id“, „user_id“, „username“, „comment“ i „created_at“. Struktura tablice se može vidjeti na slici **4.7**.

Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
id	int(11)			No	*None*		AUTO_INCREMENT	Change Drop More
product_id	int(11)			No	*None*			Change Drop More
user_id	int(11)			No	*None*			Change Drop More
username	varchar(50)	utf8mb4_general_ci		No	*None*			Change Drop More
comment	text	utf8mb4_general_ci		No	*None*			Change Drop More
created_at	timestamp			No	current_timestamp()			Change Drop More

Slika 4.7 Struktura tablice "comments“

Stupac „id“ je primarni ključ i označava identifikacijski broj komentara. Stupac „product_id“ je strani ključ koji povezuje tablicu „comments“ s tablicom „products“ i označava identifikacijski broj proizvoda na koji se komentar odnosi, a „user_id“ je strani ključ koji povezuje tablicu „comments“ s tablicom „user“ i označava identifikacijski broj korisnika koji je postavio komentar. Stupac „comment“ sadrži tekst komentara, a „created_at“ vrijeme kreiranja komentara. Primjer sadržaja tablice se može vidjeti na slici **4.8**.

id	product_id	user_id	username	comment	created_at
39	1	9	user1	odličan proizvod	2024-06-26 22:54:14
40	4	9	user1	lice je masno nakon ove kreme	2024-06-26 22:54:43
41	8	10	user2	Koža je mekana	2024-06-26 22:55:22
42	9	11	user3	dobro	2024-06-26 22:55:59

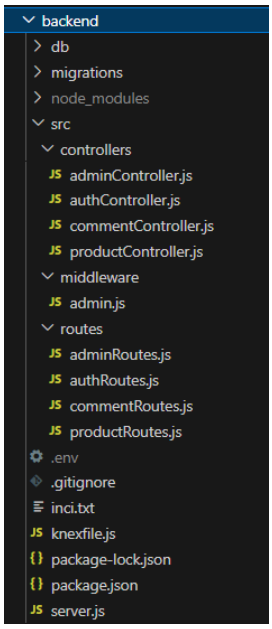
Slika 4.8 Sadržaj tablice "comments"

4.1.5. Povezivanje s bazom podataka

Povezivanje na bazu podataka se postiže korištenjem biblioteke `knex`, koja pruža apstrakciju za izvršavanje SQL upita. Konfiguracija za povezivanje definira detalje kao što su tip baze podataka i informacije za spajanje (host, korisničko ime, lozinka). `Knex` omogućuje stvaranje instance koja se koristi za izvršavanje SQL upita prema konfiguriranoj bazi, što olakšava interakciju s bazom podataka u različitim okruženjima poput razvoja, produkcije ili testiranja.

4.2. Backend dio

Backend dio aplikacije odnosi se na dijelove računalnog *koda* koji omogućuju rad aplikacije, a koji nije vidljiv korisnicima. *Backend* dio ove web aplikacije rađen je u razvojnom okviru Node.js. Struktura projekta vidljiva je na slici **4.9**.



Slika 4.9 *Backend* struktura

Glavne komponente *backend* dijela aplikacije su server.js, kontroleri i rute.

4.3. Datoteka „server.js“

Kao ulazna točka u aplikaciju služi datoteka „server.js“. U njoj su postavljene osnovne postavke servera. Razvojno okruženje „Express“ koristi se za rukovanje HTTP zahtjevima, što omogućava jednostavno definiranje ruta *i middleware-a*. Na slici **4.10** je prikazan programski *kod* „server.js“ datoteke. Definira se *middleware* unutar *routera* koji logira metodu i URL svakog dolaznog zahtjeva te se zatim prosljeđuje zahtjev dalje na sljedeći middleware ili rutu. Registriraju se rute za autentifikaciju pod /auth, rute za proizvode i komentare pod /products te rute za administratora pod /admin. Na kraju se pokreće server na definiranom portu.

```
const router = express.Router();
router.use((req, res, next) => {
    console.log('Middleware executed');
    console.log('Request:', req.method, req.originalUrl);
    next();
});

app.use('/auth', authRoutes);
app.use('/products', productRoutes);
app.use(serveStatic("../frontend/dist"));
app.use('/products', commentRoutes);
app.use('/admin', isAdmin, adminRoutes);

const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Slika 4.10 Programski *kod* "server.js"

4.4. Kontroleri

Kontroleri služe kao most između modela (koji upravljaju podacima) i pogleda (koji prikazuju podatke korisnicima). Glavna svrha kontrolera je obrada dolaznih HTTP zahtjeva, primjena poslovne logike i vraćanje odgovarajućih odgovora klijentima. U ovoj web aplikaciji postoje kontroleri za autentifikaciju korisnika, za rad s proizvodima i za rad s komentarima.

4.4.1. Kontroler za autentifikaciju korisnika

Kontroler sadrži funkcije za kreiranje korisnika, prijavu korisnika te pronalaženje korisnika po korisničkom imenu. Primjer programskog *koda* kontrolera prikazan je na slici **4.11** .

Funkcija „createUser“ omogućava registraciju novih korisnika. Prima korisničke podatke iz tijela zahtjeva (korisničko ime, email adresu i lozinku). Provjerava postoji li korisnik s istom email adresom ili s istim korisničkim imenom. Kod provjere se poziva Knex.js upit nad tablicom „user“, gdje „db“ predstavlja objekt Knex instance koji je prethodno konfiguriran za rad s bazom podataka, “ where({ username }“ definira uvjet pretraživanja u bazi podataka, a metoda first() vraća samo prvi rezultat koji zadovoljava zadani uvjet. „Await“ se koristi kako bi se izvršio asinkroni upit. Ista logika je upotrijebljena za provjeru postajanje email adrese. Zatim se *hashira* lozinka pomoću „bcrypt“. Biblioteka „bcrypt“ je biblioteka za *hashiranje* lozinki koja je optimizirana za sigurno i

efikasno *hashiranje*. Broj "soljenja" (engl. *salt rounds*) je postavljen na 10. Nakon svega se sprema novi korisnik u bazu podataka.

Funkcija „loginUser“ omogućava korisnicima prijavu u sustav. Prvo se dohvaćaju korisničko ime i lozinka iz zahtjeva. Provjerava se postoji li korisnik s danim korisničkim imenom, ako postoji uspoređuje se odgovara li unesena lozinka hashiranoj lozinki u bazi podataka pomoću „bcrypt.compare.“ te generira JWT token pomoću „jwt.sign“ koji se vraća klijentu (ili odgovarajuću poruku o grešci).

Funckija „getUserByUsername“ dohvaća korisničke podatke na temelju korisničkog imena. Preuzima korisničko ime iz URL parametara („req.params“) te traži korisnika s danim korisničkim imenom u bazi podataka. Na kraju vraća podatke o korisniku ili odgovarajuću poruku o grešci.

```
const createUser = async (req, res, db) => {
  try {
    const { username, email, password } = req.body;
    const existingUsername = await db('user').where({ username }).first();
    if (existingUsername) {
      return res.status(400).json({ message: 'Username is already taken' });
    }
    const existingEmail = await db('user').where({ email }).first();
    if (existingEmail) {
      return res.status(400).json({ message: 'Email is already registered' });
    }
    const hashedPassword = await bcrypt.hash(password, 10);
    await db('user').insert({
      username,
      email,
      password: hashedPassword,
      role: "user"
    });
    res.status(201).json({ message: 'User created successfully' });
  } catch (error) {
    console.error('Error registering user:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
};
```

Slika 4.11 Programski *kod* funkcije createUser

4.4.2. Kontroler za rad s proizvodima

Kontroler „productControlller.js“ sadrži tri funckije : „getItems“, „getProductDetails“ i „getFilteredItems“. Programski *kod* funckije „getItems“ nalazi se na slici **4.12** , „getProductDetails“ na slici **4.13**, a „getFilteredItems“ na slici **4.14**.

Funkcija „getItems“ dohvaća proizvode iz baze podataka i vraća ih u JSON formatu. Za izvršavanje upita se koristi Knex. Svi proizvodi iz tablice se dohvaćaju pomoću „.select('*')“.

Funkcija „getProductDetails“ dohvaća pojedinosti o proizvodu prema njegovom ID-u. ID se uzima iz parametara zahtjeva. Za dohvaćanje odgovarajućeg proizvoda iz baze podataka se koristi Knex. Ako proizvod nije pronađen, vraća se HTTP status 404, a slučaju greške, vraća se HTTP status 500.

Funkcija „getFilteredItems“ omogućava filtriranje proizvoda prema tipu kože. Tip kože se uzima iz *query* parametara zahtjeva te se zatim se dohvaćaju poželjni i neželjen sastojci za taj tip kože iz tablice „skin_type_ingredients“. Dohvaćaju se svi proizvodi iz tablice „products“ te ih se filtrira prema sastojcima. Proizvod je prihvatljiv ako sadrži barem jedan poželjni sastojak i ne sadrži nijedan neželjeni sastojak. Kao odgovor se vraćaju filtrirani proizvodi.

```
export const getItems = async (req, res) => {
  try {
    const products= await knex('products').select('*')
    console.log(products);
    res.status(200).json(products);
  } catch (error) {
    console.error('Error fetching items:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
};
```

Slika 4.12 Programski *kod* funkcije "getItems"

```
export const getProductDetails = async (req, res) => {
  try {
    const productId = req.params.id;
    const product = await knex('products').where('id', productId).first();
    if (!product) {
      return res.status(404).json({ message: 'Proizvod nije pronađen.' });
    }
    res.status(200).json(product);
  } catch (error) {
    console.error('Greška prilikom dohvaćanja pojedinosti o proizvodu:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
};
```

Slika 4.13 Programski *kod* funkcije getProductDetails

```
export const getFilteredItems = async (req, res) => {
  try {
    const skinType = req.query.skinType;
    const desirableIngredients = await knex('skin_type_ingredients')
      .where({ skin_type: skinType, is_desirable: 1 })
      .pluck('ingredient');
    const undesirableIngredients = await knex('skin_type_ingredients')
      .where({ skin_type: skinType, is_desirable: 0 })
      .pluck('ingredient');
    const allProducts = await knex('products').select('*');
    const filteredProducts = allProducts.filter(product => {
      const productIngredients = product.ingredients.split(',').map(ingredient => ingredient.trim());
      const hasDesirableIngredients = desirableIngredients.some(ingredient => productIngredients.includes(ingredient));
      const hasUndesirableIngredients = undesirableIngredients.some(ingredient => productIngredients.includes(ingredient));
      return hasDesirableIngredients && !hasUndesirableIngredients;
    });
    res.status(200).json(filteredProducts);
  } catch (error) {
    console.error('Error fetching filtered items:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
};
```

Slika 4.14 Programski *kod* funkcije "getFilteredItems"

4.4.3. Kontroler za rad s komentarima

Komentari omogućuju korisnicima da podijele svoja iskustva i mišljenja o proizvodima i time olakšaju izbor ostalim korisnicima. Rad s komentarima omogućen je „commentController.js“ kontrolerom. Programski *kod* kontrolera nalazi se na slici **4.15**.

```
export const getComments = async (req, res) => {
  try {
    const productId = req.params.productId;
    const comments = await knex('comments').where({ product_id: productId });
    res.status(200).json(comments);
  } catch (error) {
    console.error('Error fetching comments:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
};
export const postComment = async (req, res) => {
  try {
    console.log(req.body);
    const { comment, userId, username } = req.body;

    const productId = req.params.productId;
    const newComment = { product_id: productId, user_id: userId, username, comment };
    const [id] = await knex('comments').insert(newComment);
    res.status(201).json({ id, ...newComment });
  } catch (error) {
    console.error('Error posting comment:', error);
    res.status(500).json({ message: 'Internal server error' });
  }
};
```

Slika 4.15 Programski *kod* "commentController.js"

Kontroler se sastoji od funckija „getComments“ i „postComment“.

Funkcija „getComments“ omogućuje dohvaćanje komentara. Ova funkcija dohvaća sve komentare za određeni proizvod iz tablice „comments“. Uzima se „productId“ iz parametra zahtjeva, a zatim se uz korištenje Knexa izvršava SQL upit koji dohvaća sve komentare. Rezultati se vraćaju u JSON formatu ili se u slučaju greške vraća HTTP status 500 s porukom.

4.5. Kontroler za administratora

Administrator ima mogućnost pregleda i brisanja korisnika, proizvoda i komentara. Na slikama **4.16** i **4.17** se nalaze programski *kodovi* funkcija za dohvaćanje svih korisnika i brisanje korisnika. Funkcija „getAllUsers“ dohvaća sve korisnike iz tablice „user“. Za slanje SQL upita i dohvaćanje ID-a, korisničkog imena i email adrese svih korisnika koristi se „knex“. Ako upit uspije, vraća se JSON s podacima o korisnicima i HTTP status 200 (OK). U slučaju greške, vraća se status 500 (Internal Server Error) i poruka o grešci. Funkcija „deleteUser„ briše korisnika iz tablice „user“

prema ID-u. Za brisanje zapisa iz baze podataka se koristi „knex“. Ako je brisanje uspješno, vraća se poruka o uspjehu i status 200. U slučaju greške, vraća se status 500 s porukom o grešci.

```
export const getAllUsers = async (req, res) => {
  try {
    const users = await knex('user').select('id', 'username', 'email');
    res.status(200).json(users);
  } catch (error) {
    console.error('Error fetching users:', error);
    res.status(500).json({ message: 'Error fetching users' });
  }
};
```

Slika 4.16 Programski *kod* "getAllUsers"

```
export const deleteUser = async (req, res) => {
  try {
    const { id } = req.params;
    await knex('user').where('id', id).del();
    res.status(200).json({ message: 'User deleted' });
  } catch (error) {
    console.error('Error deleting user:', error);
    res.status(500).json({ message: 'Error deleting user' });
  }
};
```

Slika 4.17 Programski *kod* "deleteUser"

Funkcije za brisanje proizvoda briše proizvod iz tablice „products“ na temelju ID-a. Ako je brisanje uspješno, vraća se poruka o uspjehu i status 200. U slučaju greške, vraća se status 500 s porukom o grešci.

Funkcija „getAllComments„ dohvaća sve komentare iz tablice „comments“ zajedno s podacima o proizvodima iz tablice „products“. Koristi *join* operaciju kako bi se povezale dvije tablice i dobili podaci o komentaru, korisničkom imenu te imenu proizvoda i njegovom ID-u. Ako je upit uspješan, vraća se JSON s podacima i status 200. U slučaju greške, vraća se status 500 s porukom

o grešci. Funkcija „deleteComment„ briše komentar iz tablice „comments“ na temelju ID-a. Ako je brisanje uspješno, vraća se poruka i status 200. U slučaju greške, vraća se status 500 s porukom o grešci.

4.6. Upravljanje rutama

Rute su temeljni koncept u web razvoju. Koristeći rute, API se može razviti na način koji je jednostavan za održavanje i proširenje. Korištenjem različitih HTTP metoda (GET, POST, PUT, DELETE) rute definiraju kako aplikacija odgovara na dolazne zahtjeve i koje akcije se poduzimaju u odgovoru na te zahtjeve. U okviru Node.js aplikacija, rute se često definiraju koristeći Express.js. U ovoj web aplikaciji postoje četiri modula ruta: „authRoutes“, „commentRoutes“, „productRoutes“ i „adminRoutes“. Na slici **4.18** prikazan je programski *kod* za modul koji definira rute za autentifikaciju korisnika („authRoutes.js“). Ovaj modul sadrži rute za registraciju, prijavu i dohvaćanje korisničkih podataka.

```
const router = express.Router();
router.use((req, res, next) => {
    req.db = knex;
    next();
  });

router.post('/register', async (req, res) => {
  await createUser(req, res, req.db);
});
router.post('/login', async (req, res) => {
  await loginUser(req, res, req.db);
});
router.get('/user/:username', async (req, res) => {
  await getUserByUsername(req, res, req.db);
});
export default router;
```

Slika 4.18 Programski *kod* "authRoutes.js"

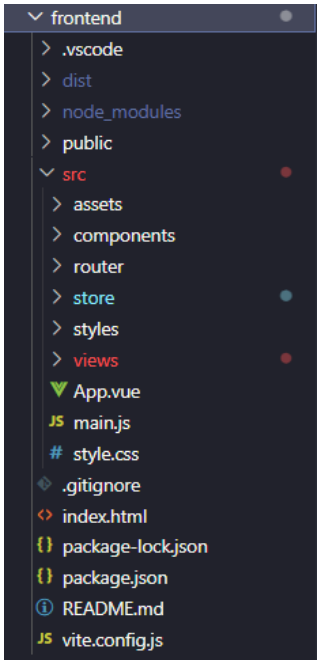
Na slici **4.19** se nalaze rute za administratora. Ove rute omogućuju rad s korisnicima, proizvodima i komentarima, a pristup im je ograničen na administratore zahvaljujući *middleware* funkciji „isAdmin“.

```
router.use(isAdmin);
router.get('/users', getAllUsers);
router.delete('/users/:id', deleteUser);
router.delete('/products/:id', deleteProduct);
router.get('/comments', getAllComments);
router.delete('/comments/:id', deleteComment);
router.get('/isAdmin', isAdmin);
export default router;
```

Slika 4.19 Programski *kod* "adminRoutes.js"

4.7. Frontend dio

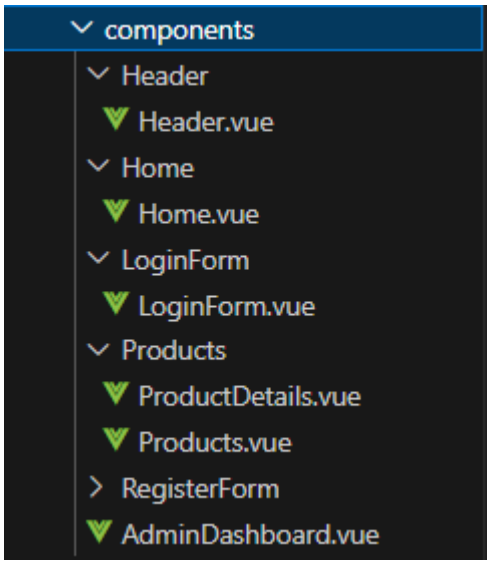
Frontend dio aplikacije je odgovoran za interakciju s korisnicima, vizualni prikaz podataka te komunikaciju sa serverom. *Frontend* dio ove web aplikacije rađen je u Vue.js. Struktura projekta vidljiva je na slici **4.20**. Projekt je sa src direktorijem koji sadrži sve važne dijelove aplikacije.



Slika 4.20 Struktura *frontend* dijela

4.8. Komponente

Komponente se u Vue.js koriste za lakšu ponovnu upotrebu *koda*. Omogućuju organiziranje aplikacije na način koji je lakši za održavanje i razumijevanje. Na slici **4.21** nalazi se popis komponenti korištenih u ovoj web aplikaciji.



Slika 4.21 Komponente

4.8.1. Komponenta zaglavlja

Komponenta „Header.vue“ implementira zaglavlje. Programski *kod* nalazi se na slici **3.3** . Funkcije „useUser“ i „clearUser“ se učitavaju iz „../../store/auth.js“ te upravljaju stanjem korisnika (prijava/odjava). Iz „vue-router“ se učitava „useRouter“ kako bi se omogućila navigacija unutar komponente. Za navigaciju se koristi instanca Vue Router-a, „router“. Iz „useUser“ se dobiva „user“ koji predstavlja trenutno prijavljenog korisnika. Funkcija „logout“ poziva „clearUser“ kako bi se očistilo stanje korisnika. Nakon odjave preusmjerava se na stranicu za prijavu. Dio predloška označava strukturu komponente. Koristi se uvjetno prikazivanje (v-if i v-else) kako bi se prikazale različite opcija ovisno o tome je li korisnik prijavljen ili ne. Ako je korisnik prijavljen, prikazuje se poruka s korisničkim imenom i gumb za odjavu. Ako nije, prikazat će poveznice za prijavu i registraciju. Za omogućavanje navigacije bez ponovnog učitavanja stranice koristi se komponenta RouterLink.

4.8.2. Komponenta za prijavu

Komponenta „LoginForm.vue“ implementira funkcionalnost prijave korisnika. Omogućuje unos korisničkog imena i lozinke te prijavu. Programski *kod* nalazi se na slikama **4.22** i **4.23** . Funckija „login“ šalje zahtjev za prijavu korisnika s korisničkim imenom i lozinkom. Dobiveni token se sprema u „localStorage“, dohvaćaju se korisnički podaci te se spremaju pomoću „setUser“. Nakon uspješne prijave korisnika se preusmjerava na početnu stranicu. U dijelu predloška (engl. *template*) (slika **4.23**) se nalazi forma za unos korisničkog imena i lozinke te dugme za prijavu. Oznaka „<form @submit.prevent="login" class="auth-form">“ definira obrazac za prijavu sa „submit“ događajem koji poziva funkciju „login“, „@submit.prevent="login"“ sprječava osvježavanje stranice i poziva funkciju „login“. Polja za unos koriste „v-model“ za spajanje vrijednosti s referencom.

```
const response = await axios.post("http://localhost:3000/auth/login", {
    username: username.value,
    password: password.value,
});
if (response.status !== 200) {
    throw new Error("Login failed: Invalid response");
}
const token = response.data.token;
localStorage.setItem("token", token);
const userInfoResponse = await axios.get(
    `http://localhost:3000/auth/user/${username.value}`,
    {
        headers: {
            Authorization: `Bearer ${token}`,
        },
    }
);
if (userInfoResponse.status !== 200) {
    throw new Error("Login failed: Invalid user info response");
}
const userInfo = userInfoResponse.data;
setUser(userInfo);

console.log("Logged in successfully!");
router.push("/");
```

Slika 4.22 "LoginForm.vue" skripta

```
<template>
    <div class="auth-container">
        <form @submit.prevent="login" class="auth-form">
                <input
                    type="username"
                    id="username"
                    v-model="username"
                    class="auth-input"
                    placeholder="Username"
                    required
                />
                <input
                    type="password"
                    id="password"
                    v-model="password"
                    class="auth-input"
                    placeholder="Password"
                    required
                />

            <button type="submit" class="content-action-btn" id="auth-btn">LOGIN</button>
        </form>
    </div>
</template>
```

Slika 4.23 "LoginForm.vue" predložak

4.8.3. Komponenta za registraciju

Za registraciju se koristi komponenta „RegisterForm.vue“. Komponenta omogućuje slanje podatka na poslužitelj te se prikazuju odgovarajuće poruke o uspjehu ili neuspjehu registracije. Predložak (engl. *Template*) sadrži obrazac za unos korisničkih podataka (korisničko ime, email, lozinka) i dugme za podnošenje zahtjeva. Oznaka „<form @submit.prevent="register" autocomplete="off" class="auth-form">“ Definira obrazac za registraciju sa „submit“ događajem koji poziva funkciju „register“, „@submit.prevent="register"“ sprječava osvježavanje stranice i poziva funkciju „register“. Polja za unos korisničkog imena, lozinke i email adrese koje koriste „v-model“ za spajanje vrijednosti s referencom. Dugme za registraciju podnosi obrazac i pokreće funkciju „register“. U skripti (slika **4.24**) se nalazi funkcija za registraciju. Funkcija „register“ je asinkrona funkcija koja šalje POST zahtjev za registraciju s podacima iz referenci.

```
const username = ref("");
const email = ref("");
const password = ref("");
const confirmPassword = ref("");
const errorMessage = ref("");
const router = useRouter();
const register = async () => {
    if (password.value !== confirmPassword.value) {
        alert("Passwords do not match!");
        return;
    }

    try {
        await axios.post("/auth/register", {
            username: username.value,
            email: email.value,
            password: password.value,
            role: "user",
        });
        try {
            await loginUser(username.value, password.value, router);
        } catch (error) {
            errorMessage.value = "Prijava nije uspjela";
            console.error("Login failed:", error.message);
        }

    } catch (error) {
        errorMessage.value = "Registracija nije uspjela";
        console.error("Registration failed:", error);
    }
};
```

Slika 4.24 "RegisterForm.vue" skripta

4.8.4. Komponente za proizvode

Za proizvode su korištene dvije komponente, a to su „Products.vue“ i „ProductDetails.vue“.

Komponenta „Product.vue“ implementira funkcionalnosti za prikaz proizvoda iz baze podataka tako da su filtrirani po tipu kože. Komponenta koristi „axios“ za komunikaciju s *backendom*, dobivajući podatke o proizvodima iz API-ja. Predložak omogućuje korisnicima odabir tipa kože pomoću padajućeg izbornika, zatim se prikazuju filtrirani proizvodi. Na promjenu tipa kože pokreće se metoda „fetchFilteredProducts“, koja šalje HTTP GET zahtjev na *backend* koristeći „axios“ kako bi se dohvatili filtrirani proizvodi na temelju odabranog tipa kože. Pomoću „v-for“ petlje prolazi se kroz sve proizvode, postavlja se jedinstveni ključ za svaki proizvod koristeći njegov ID te se postavljaju URL rute za svaki proizvod kako bi korisnik mogao kliknuti na proizvod i vidjeti njegove detalje. U dijelu skripte nalazi se asinkrona funkcija „fetchFilteredProducts“ koja dohvaća proizvode filtrirane prema tipu kože. Šalje se GET zahtjev na API *endpoint* „http://localhost:3000/products/filter“ s trenutnim „skinType“ kao parametrom. Dohvaćeni proizvodi se spremaju u polje „products“.

Komponenta „ProductDetails.vue“ prikazuje detalje o određenom proizvodu, uključujući ime, sliku i sastojke proizvoda te omogućuje pregled i dodavanje komentara. Predložak sadrži kontejnere za prikaz podataka o proizvodu i komentare te polje za unos komentara. Pomoću „v-if="user"“ provjerava se da li je korisnik prijavljen, ako je moguće je komentirati, a ako nije prikazuje se poruka i link za prijavu. U dijelu skripte nalaze se funkcije „getProduct“ (slika **4.28**), „getComments“ (slika **4.26**), „getUser“ (slika **4.25**), „postComment“ (slika **4.27**) i „formatDateTime“ (slika **Pogreška! Izvor reference nije pronađen.**). Funkcija „getProduct“ je a sinkrona funkcija koja dohvaća informacije o proizvodu prema ID-u iz URL-a. Naredba „axios.get('http://localhost:3000/products/${id}')“ šalje GET zahtjev na API *endpoin*t za dohvaćanje informacija o proizvodu, a „this.product = response.data“ sprema dohvaćene informacije o proizvodu u objekt „product“. Funkcija „getComments“ je također asinkrona funkcija koja dohvaća komentare. Šalje se GET zahtjev na API *endpoint* za dohvaćanje komentara te se dohvaćeni komentari spremaju u polje „comments“. Funkcija „getUser“ dohvaća informacije o korisniku iz „localStorage“ i sprema ih u objekt „user“. Funkcija „postComment“ je asinkrona funkcija koja omogućava korisnicima da dodaju novi komentar za proizvod. Šalje se POST zahtjev na API *endpoint* za dodavanje komentara. Ako je zahtjev uspješan dodaje se novi komentar u polje „comments“ te se prazni polje za unos novog komentara. Funkcija „formatDateTime“ pretvara datum i vrijeme iz stringa u objekt „Date“.

```
getUser() {
  const user = JSON.parse(localStorage.getItem('user'));
  if (user) {
    this.user = user;
  }
},
```

Slika 4.25 Programski *kod* "getUser"

```
async getComments() {
  const id = this.$route.params.id;
  try {
    const response = await axios.get(`http://localhost:3000/products/${id}/comments`);
    this.comments = response.data.map(comment => ({
      ...comment,
      created_at: new Date(comment.created_at)
    }));
  } catch (error) {
    console.error('Error fetching comments:', error);
  }
},
```

Slika 4.26 Programski *kod* "getComments"

```
async postComment() {
  const id = this.$route.params.id;
  if (!this.newComment.trim()) return;

  try {
    const response = await axios.post(`http://localhost:3000/products/${id}/comments`, {
      comment: this.newComment,
      userId: this.user.ID,
      username: this.user.username
    });
    this.comments.push({
      ...response.data,
      created_at: new Date(response.data.created_at)
    });
    this.newComment = '';
  } catch (error) {
    console.error('Error posting comment:', error);
  }
},
```

Slika 4.27 Programski *kod* "postComment"

```
async getProduct() {
  const id = this.$route.params.id;
  try {
    const response = await axios.get(`http://localhost:3000/products/${id}`);
    this.product = response.data;
  } catch (error) {
    console.error('Error fetching product:', error);
  }
},
```

Slika 4.28 Programski *kod* "getProduct"

4.8.5. Komponenta početne stranice

Za početnu stranicu korištena je komponenta „Home.vue“. Ova komponenta prikazuje početnu stranicu web aplikacije koja se prilagođava *skrolanju* korisnika. Kada se korisnik pomiče prema dolje, pozicija pozadine zaglavlja i položaj podnožja prilagođavaju se kako bi stvorili efekt paralakse. Ako je korisnik prijavljen, prikazuje se personalizirana poruka i poveznica na proizvode, dok neprijavljeni korisnici vide opcije za prijavu i registraciju.

4.9. Komponenta administratora

Ova komponenta predstavlja administrativnu kontrolnu ploču koja omogućuje administratoru upravljanje korisnicima, proizvodima i komentarima unutar aplikacije. Kada se komponenta učita, provjerava se da li je korisnik administrator koristeći „useUser“. Ako je korisnik administrator, „isAdmin“ se postavlja na *true*, a zatim se dohvaćaju podaci o korisnicima, proizvodima i komentarima koristeći „axios“ za slanje HTTP zahtjeva prema API-ju.

Komponenta prikazuje tri glavna odjeljka: korisnike, proizvode i komentare. Svaki odjeljak sadrži popis s odgovarajućim informacijama. Pored svakog elementa nalazi se gumb za brisanje, koji omogućuje administratoru da ukloni odabrani element. Nakon brisanja, popis se automatski osvježava kako bi prikazao ažurirane podatke.

Ako korisnik nije administrator, umjesto upravljačkih elemenata, prikazuje se poruka o nemogućnosti pristupa stranici. Ovaj pristup osigurava daaliovlašteni korisnici mogu pristupiti podacima i funkcionalnostima, čime se povećava sigurnost aplikacije.

4.10. Pogledi

Pogledi (engl. *Views*) u Vue.js se odnose na komponente koje predstavljaju konkretne stranice unutar aplikacije. Unutar pogleda se uključuju komponente. Ova web aplikacija sadrži poglede „AdminDasboardView.vue“ (slika **4.29**), „HomeView.vue“, „LoginView.vue“, „ProductDetailsView.vue“ , „ProductView.vue“ i „RegisterView.vue“ .

```
<script setup>
import Header from '../components/Header/Header.vue';
import AdminDashboard from '../components/AdminDashboard.vue';
</script>
<template>
    <Header></Header>
    <AdminDashboard></AdminDashboard>
</template>
```

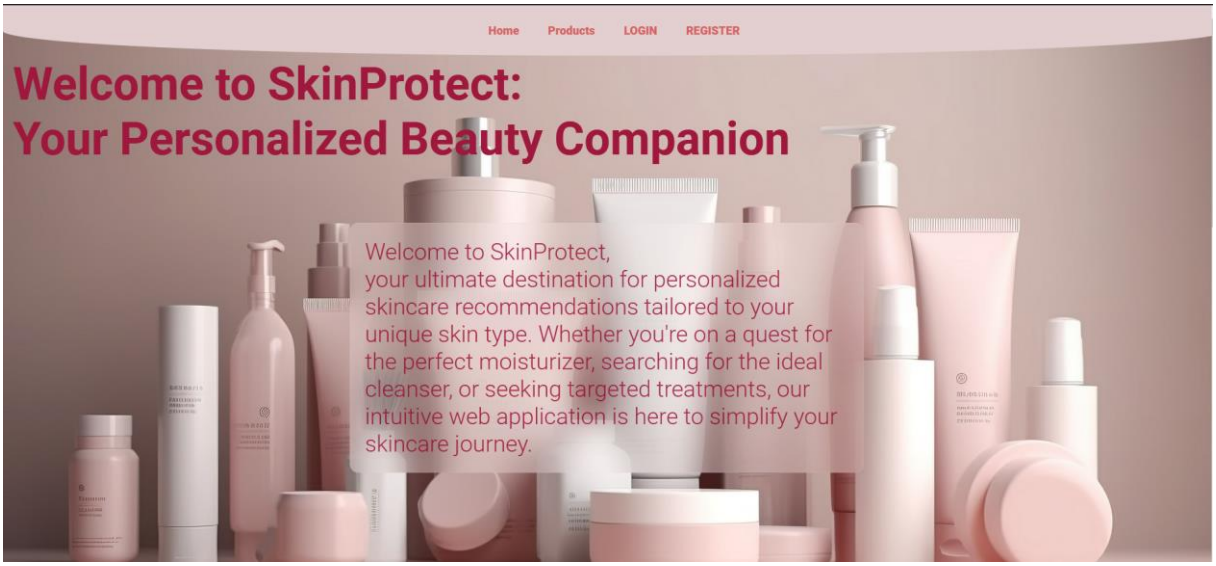
Slika 4.29 „AdminDashboardView.vue“

4.11. Rute

Upravljanje rutama omogućava „Router“. „Router“ je biblioteka koja omogućava upravljanje rutama u aplikaciji i dinamičko prikazivanje različitih pogleda ovisno o URL-u korisnika.

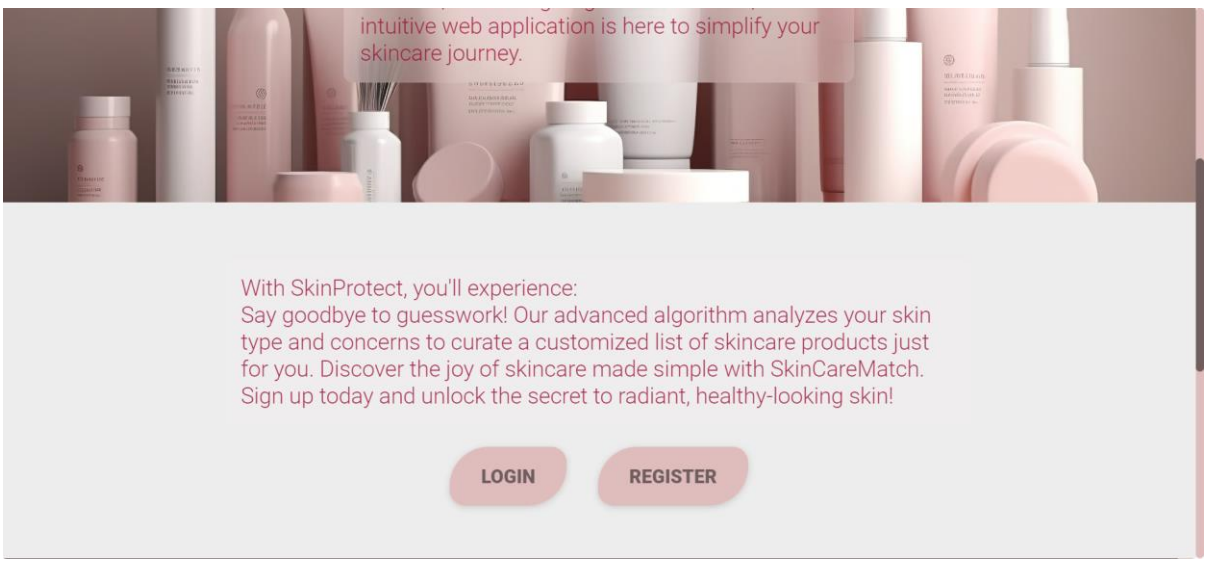
5. Izgled aplikacije

Naslovna stranica nalazi se na slici **5.1**. Pokazuje poruku dobrodošlice i uvodni tekst. U zaglavlju se nalaze linkovi na naslovnu stranicu, stranicu s proizvodima i stranicama za prijavu i registraciju.



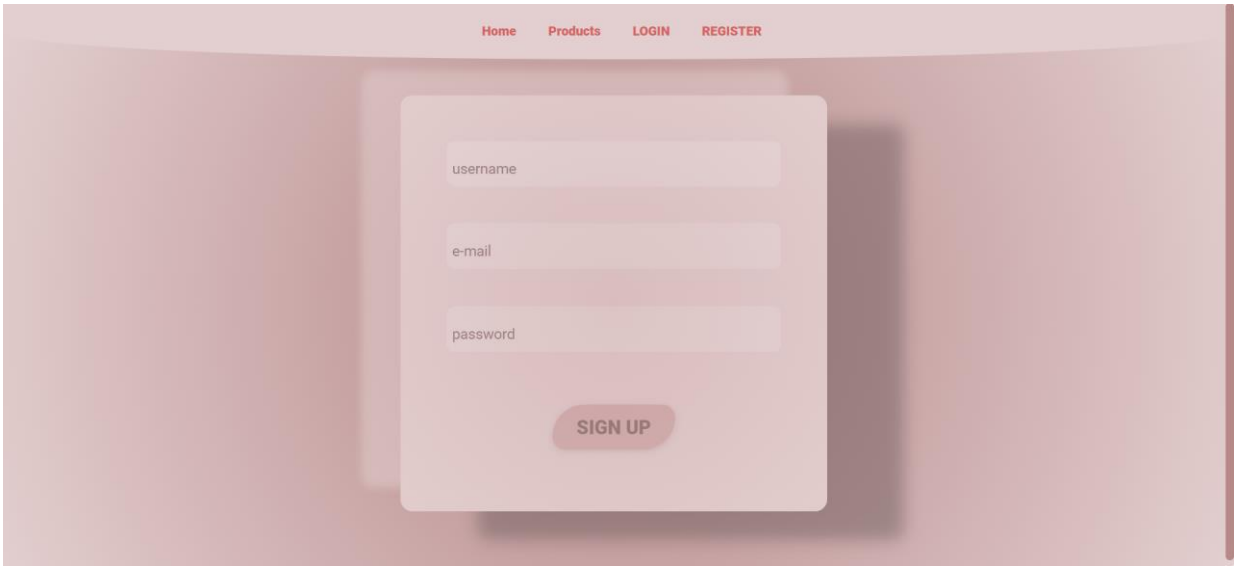
Slika 5.1 Naslovna stranica

Na *skrol* se dolazi do druge poruke i ako korisnik nije prijavljen vidljivi su linkovi za prijavu i registraciju (slika **5.2**)



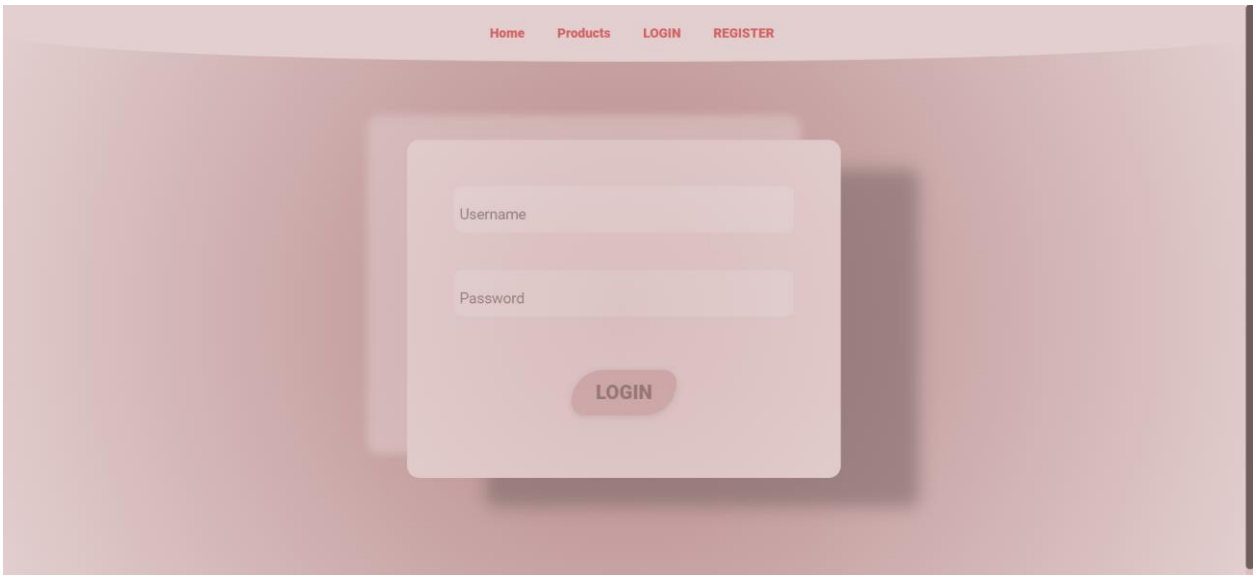
Slika 5.2 Naslovna stranica 2.dio

Klikom na „register“ se dolazi do stranice za registraciju, što se može vidjeti na slici **5.3**.



Slika 5.3 Stranica za registraciju

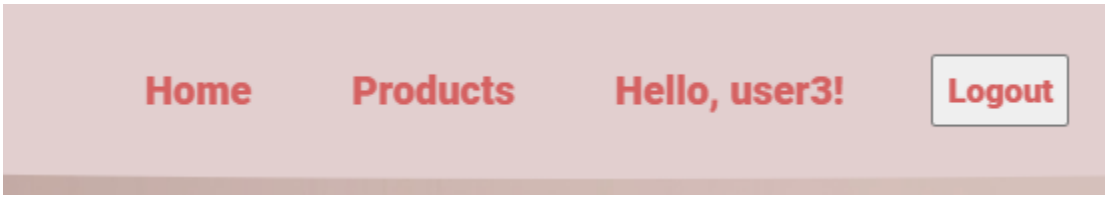
Za registraciju je potrebno unijeti korisničko ime, email adresu i lozinku te kliknuti na „sign up“.



Slika 5.4 Stranica za prijavu

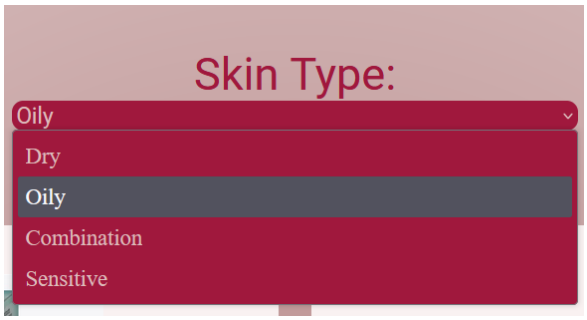
Klikom na „login“ se dolazi do stranice za prijavu (slika **5.4**). Za prijavu je potrebno unijeti korisničko ime i lozinku te kliknuti za „login“.

Kada je korisnik prijavljen u zaglavlju se prikazuje poruka s njegovim korisničkim imenom i link za odjavu (slika **5.5**).

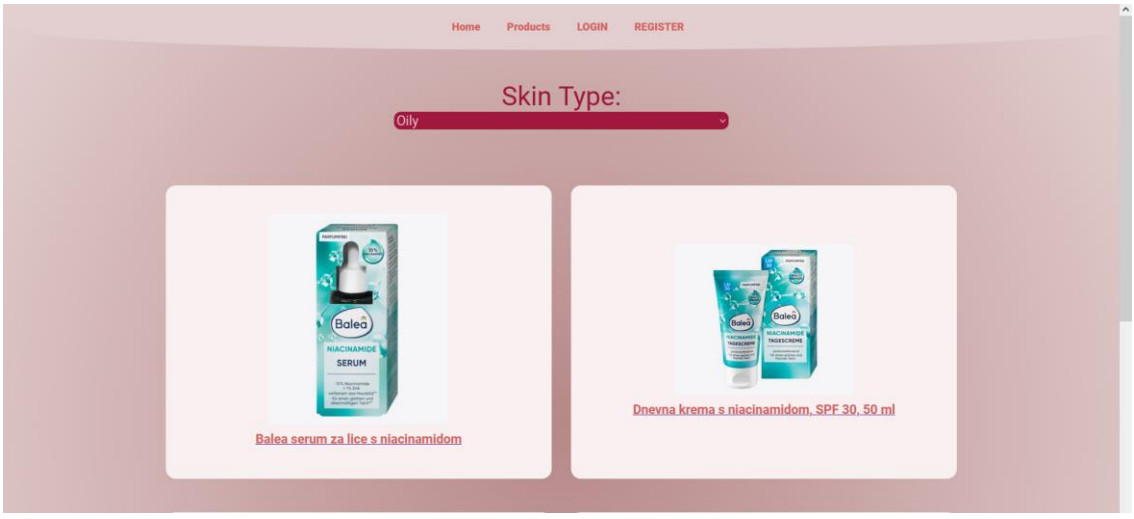


Slika 5.5 Zaglavlje kada je korisnik prijavljen

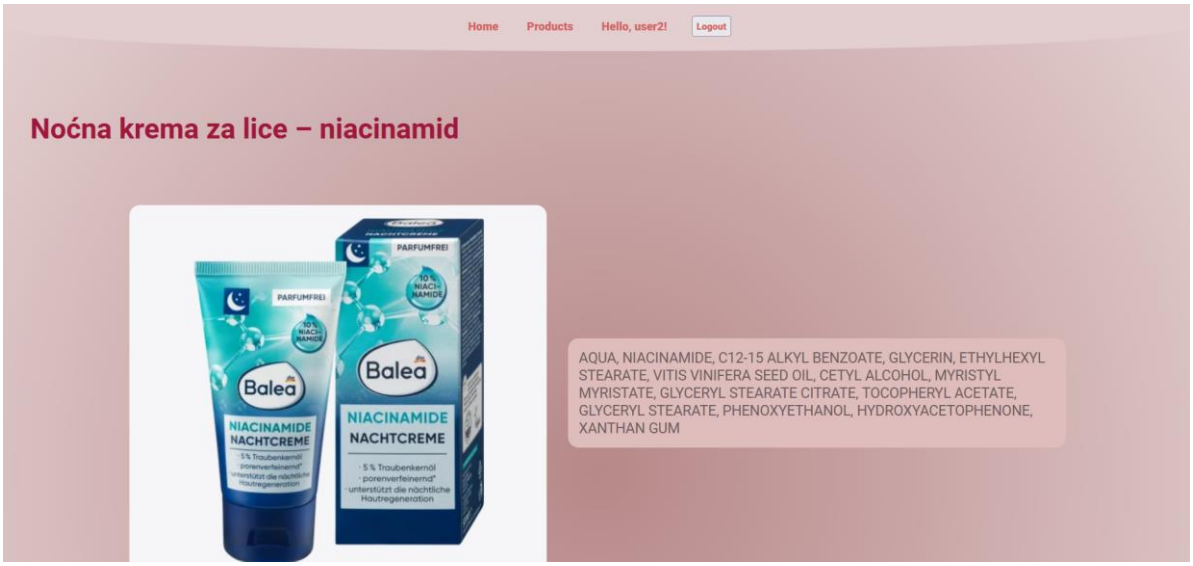
Klikom na „Products“ dolazi se do stranice s proizvodima. Pomoću padajućeg izbornika odabire se tip kože (slika **5.6**) te se prema odabranom tipu kože prikazuju proizvodi (slika **5.7**).



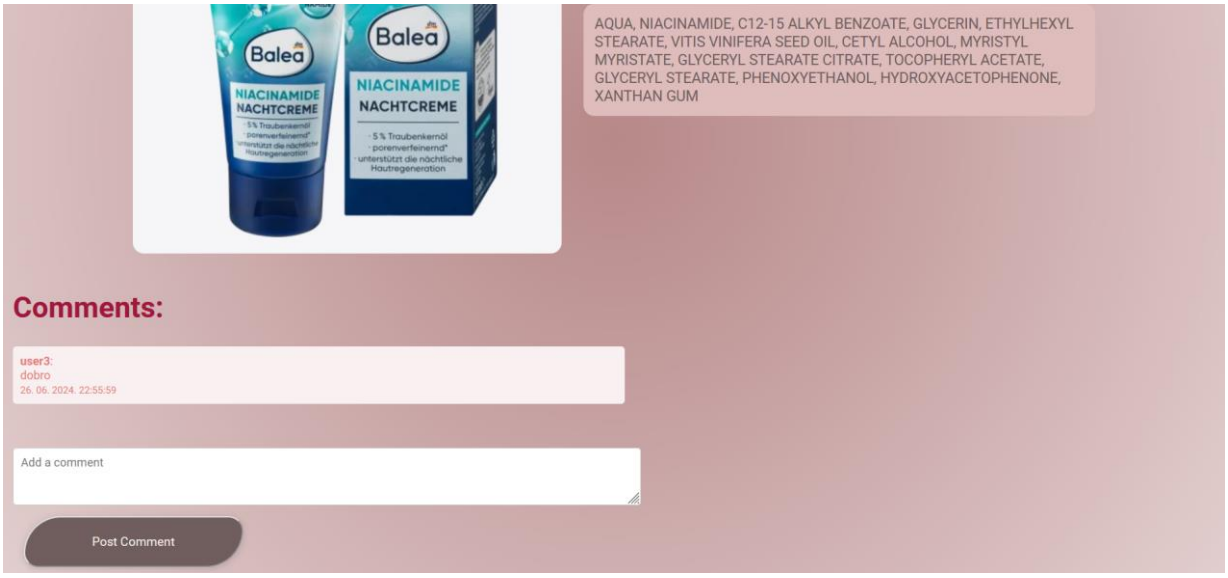
Slika 5.6 Odabir tipa kože



Klikom na proizvod dolazi se do stranice s detaljima o proizvodu (slika **5.8**) .



Slika 5.8 Detalji o proizvodu



Slika 5.9 Komentari za prijavljene korisnike

Prijavljeni korisnici imaju mogućnost komentiranja (slika **5.9**), dok neprijavljeni vide poruku i link za prijavu (slika **5.10**) .

Slika 5.10 Komentari za neprijavljene korisnike

Administrator ima mogućnost brisanja korisnika, proizvoda i komentara. Izlistani su mu korisnici, proizvodi i komentari te pored svakog ima tipku „delete“ pomoću koje može obrisati željeno (slika **5.11**).

Slika 5.11 Ploča administratora

6. Zaključak

Ovaj završni rad je imao zadatak objasniti proces razvoja web aplikacije koja pruža korisnicima personalizirano iskustvo u svijetu njege kože, omogućujući im da unesu svoj tip kože i dobiju listu kozmetičkih proizvoda koji su najprikladniji za njih. Kroz intuitivno sučelje, korisnici mogu jednostavno filtrirati proizvode prema tipu kože, pregledavati detalje o proizvodima i čitati komentare drugih korisnika. Implementacija Vue.js omogućava dinamično i responzivno korisničko sučelje, dok korištenje „Axios-a“ za komunikaciju s *backend* serverom (Node.js) osigurava brzu i pouzdanu razmjenu podataka. S ovom aplikacijom, korisnici mogu sa sigurnošću odabrati proizvode koji odgovaraju njihovim specifičnim potrebama, čime se poboljšava njihova rutina njege kože i postiže zdraviji i ljepši izgled kože.

Literatura

[1] SkinCarisma, [Mrežno]. Dostupno na: https://www.skincarisma.com/. [Pokušaj pristupa 24. 06. 2024.].

[2] INCIdecoder,[Mrežno]. Dostupno na: https://incidecoder.com/. [Pokušaj pristupa 24. 06. 2024.].

[3] SkinSAFE, [Mrežno]. Dostupno na: https://www.skinsafeproducts.com/. [Pokušaj pristupa 24. 06. 2024.].

[4] W3Schools, HTML Introduction, [Mrežno]. Dostupno na: https://www.w3schools.com/html/html_intro.asp. [Pokušaj pristupa 24. 6. 2024.].

[5] M. D. Network, »HTML općenito,« [Mrežno]. Dostupno na: https://developer.mozilla.org/en-US/docs/Web/HTML. [Pokušaj pristupa 25. 06. 2024.].

[6] DEV, Getting started with SCSS - The CSS Preprocessor with Superpowers, [Mrežno]. Dostupno na: https://dev.to/classicthedemigod/getting-started-with-scss-the-css-preprocessor-with-superpowers-34ed. [Pokušaj pristupa 25. 06. 2024.].

[7] GeeksforGeeks, Difference Between CSS and SCSS, [Mrežno]. Dostupno na: https://www.geeksforgeeks.org/what-is-the-difference-between-css-and-scss/. [Pokušaj pristupa 25. 06. 2024.].

[8] Vue.js, Introduction, [Mrežno]. Dostupno na: https://vuejs.org/guide/introduction.html. [Pokušaj pristupa 25. 06. 2024.].

[9] »Node.js,« [Mrežno]. Available: https://nodejs.org/en. [Pokušaj pristupa 25. 06. 2024.].

[10] w3schools, Node.js Introduction, [Mrežno]. Dostupno na: https://www.w3schools.com/nodejs/nodejs_intro.asp. [Pokušaj pristupa 25. 06. 2024.].

[11] Express.js, Express,[Mrežno]. Available: https://expressjs.com/. [Pokušaj pristupa 25. 06. 2024.].

[12] w3schools, What is MySQL?, [Mrežno]. Dostupno na: https://www.w3schools.com/MySQL/mysql_intro.asp. [Pokušaj pristupa 25. 06. 2024.].

[13] Oracle, What is MySQL?, [Mrežno]. Dostupno na: https://www.oracle.com/mysql/what-is-mysql/. [Pokušaj pristupa 25. 06. 2024.].

[14] V. S. Code, Getting Started, [Mrežno]. Dostupno na: https://code.visualstudio.com/docs. [Pokušaj pristupa 25. 06. 2024.].

Sažetak

U završnom radu obrađena je izrada web aplikacije za odabir kozmetičkih proizvoda prema tipu kože. Cilj aplikacije je omogućiti korisnicima unos svog tipa kože te im preporučiti proizvode čiji sastav najbolje odgovara njihovim potrebama. Kroz aplikaciju, korisnici mogu pregledavati proizvode, čitati komentare drugih korisnika i filtrirati rezultate prema svom tipu kože. Implementirane tehnologije, poput Vue.js za *frontend* i Node.js za *backend*, omogućuju dinamično sučelje i brzu razmjenu podataka. Aplikacija omogućava sigurniji i personaliziraniji izbor kozmetičkih proizvoda, poboljšavajući korisničku rutinu njege kože

Ključne riječi: Baza podataka, filtriranje, Node.js, proizvod, Vue.js, web aplikacija

Abstract

In the final paper, the development of a web application for selecting cosmetic products based on skin type is presented. The goal of the application is to allow users to input their skin type and recommend products whose ingredients best match their needs. Through the application, users can browse products, read comments from other users, and filter results according to their skin type. Implemented technologies, such as Vue.js for the frontend and Node.js for the backend, enable a dynamic interface and fast data exchange. The application provides a safer and more personalized selection of cosmetic products, improving the user's skincare routine.

Keywords: Database, filtering, Node.js, product, Vue.js, web application

Prilozi

Link na Github repozitorij s programskim *kodom* aplikacije: https://github.com/leagreguric/skin-protect-node.js-vue.js