

Univerzitet u Beogradu
Fakultet organizacionih nauka
Katedra za elektronsko poslovanje

Prodavnica digitalnih proizvoda

Rbr	Ime	Prezime	Broj indeksa
1.	Tamara	Gligorijević	2021/0331
2.	Jana	Gojković	2021/0300
Mentor		Tamara Naumović	
GitHub link		GitHub	

Sadržaj

1. UVOD	3
2. METODE I TEHNOLOGIJE ZA RAZVOJ VEB APLIKACIJE	4
2.1. Uprošćena Larmanova metoda	4
2.2. Tehnologije za razvoj frontend dela aplikacije	5
2.2.1. JavaScript	5
2.2.2. React	5
2.3. Tehnologije za razvoj backend dela aplikacije	6
2.3.1. PHP	6
2.3.2. Laravel	7
3. IMPLEMENTACIJA VEB APLIKACIJE	8
3.1. Faza prikupljanja korisničkih zahteva	8
3.1.1. Verbalni opis	8
3.1.2. Slučajevi korišćenja	9
3.2 Faza analize	22
3.2.1. Sistemski dijagrami sekvence	22
3.2.2. Struktura softverskog sistema	42
3.3. Faza projektovanja	43
3.3.1. Projektovanje korisničkog interfejsa (ekstranskih formi)	44
3.3.2. Projektovanje aplikacione logike	57
3.3.2.1. Laravel Kontroleri (Controller)	57
3.3.2.2. Laravel Autentifikacija (Authentication)	62
3.3.2.3. Laravel Resursi (Resources)	67
3.3.2.5. Laravel Database Seeder	96
3.3.2.6. React Hooks	98
3.3.2.7 React Komponente (Components)	99
3.3.2.8. React Rutiranje (BrowserRouter)	105
3.3.2.9 Laravel i React	109
3.3.3. Projektovanje strukture softverskog sistema	112
3.3.3.1. Migracije	112
3.3.3.2. Modeli	113
3.3.4. Projektovanje skladišta podataka	117
3.4. Faza implementacije	119
3.5. Faza testiranja	124
4. ZAKLJUČAK	126
5. LITERATURA	128

1. UVOD

Tema ovog seminarskog rada je izrada savremene veb aplikacije – digitalne prodavnice umetničkih dela. U pitanju je interaktivna platforma koja korisnicima omogućava pregled i kupovinu digitalne umetnosti, dok umetnicima pruža prostor za predstavljanje i prodaju svojih radova. Aplikacija je osmišljena tako da kombinuje moderan dizajn i intuitivno korisničko iskustvo. Pored tehničke strane, aplikacija je koncipirana kao održiv poslovni model koji funkcioniše po principu digitalne platforme. Zahvaljujući potpunoj digitalizaciji i automatizaciji procesa, sistem ne zahteva fizičku logistiku, što ga čini održivim i prilagođenim tržištu. Admini postavljaju dela i određuju cene, dok korisnici imaju mogućnost da ih pregledaju i kupe. Po završetku transakcije, kupac automatski dobija pristup digitalnom fajlu, što eliminiše potrebu za fizičkom dostavom i omogućava brzu i jednostavnu distribuciju sadržaja. Monetizacija sistema može se ostvariti kroz proviziju od prodaje, naplatu dodatnih funkcionalnosti umetnicima ili plasman promotivnog sadržaja.

Cilj rada je da prikaže kompletan proces razvoja ovakve aplikacije – od početne analize i definisanja zahteva, preko faze projektovanja i implementacije, pa sve do testiranja i evaluacije funkcionalnosti sistema. Poseban akcenat stavljen je na upotrebu savremenih tehnologija koje omogućavaju efikasan razvoj i pouzdan rad aplikacije – Laravel (PHP framework) za serverski deo i React (JavaScript biblioteka) za izgradnju korisničkog interfejsa. U nastavku rada biće detaljno prikazani svi koraci razvoja aplikacije, počevši od analize zahteva, preko dizajna i implementacije, pa sve do testiranja, evaluacije i mogućnosti daljeg unapređenja sistema.

2. METODE I TEHNOLOGIJE ZA RAZVOJ VEB APLIKACIJE

2.1. Uprošćena Larmanova metoda

U realizaciji projekta „*Digital art store*“ primenjena je uprošćena Larmanova metodologija razvoja softverskog sistema. Razvoj aplikacije prošao je kroz sledećih pet faza:

- **Prikupljanje korisničkih zahteva**

U ovoj fazi identifikovane su tri glavne uloge korisnika: neulogovani korisnik, ulogovani korisnik i administrator. Na osnovu tih uloga definisani su funkcionalni zahtevi: pregled galerije, kupovina, istorija kupovine, favoriti, kao i administrativne funkcije kao što su dodavanje i brisanje slika i kategorija.

- **Analiza**

Analizom sistema razrađeni su ključni slučajevi korišćenja za svaku ulogu, kao i relacije među entitetima. Pripremljeni su osnovni modeli podataka i sistemska arhitektura. Takođe je planirana i struktura REST API-ja za komunikaciju između frontenda i backenda.

- **Projektovanje**

Definisani su tehnički elementi sistema: korisnički interfejs (React komponente), REST API rute (Laravel), modeli baze podataka, i administratorski dashboard. Takođe su izrađeni dijagrami sekvene i PMOV dijagrami koji ilustruju interakcije korisnika i sistema.

- **Implementacija**

Frontend je razvijen korišćenjem React-a i JavaScript-a, dok je backend implementiran u Laravel PHP framework-u. Uspostavljena je komunikacija između frontenda i backenda pomoću axios biblioteke i REST API-ja. MySQL je korišćen kao sistem za upravljanje bazom podataka.

- **Testiranje**

Aplikacija je testirana manuelno, uključujući testiranje korisničkih i administratorskih funkcionalnosti. Testirani su svi ključni tokovi: registracija, login, dodavanje slike u korpu, kupovina, validacija formulara i restrikcije nad kategorijama koje mogu biti obrisane. Takođe je proverena validnost API odgovora.

2.2. Tehnologije za razvoj *frontend* dela aplikacije

2.2.1. JavaScript

JavaScript je interpretirani, višenamenski skriptni jezik koji se najčešće koristi za razvoj interaktivnih elemenata na web stranicama. Kreiran je 1995. godine od strane Brendan Eich-a u kompaniji *Netscape*. Danas predstavlja osnovu za klijentski deo svih modernih web aplikacija.

Koristi se za manipulaciju HTML elementima, upravljanje događajima, komunikaciju sa serverom (npr. pomoću `fetch` ili `axios`) i dinamičko prikazivanje sadržaja bez potrebe za reload-om stranice.

Osnovni koncepti i vrednosni tipovi:

- Primitivni tipovi: *string, number, boolean, null, undefined*
- Kompleksni tipovi: *object, array, function*
- Rad sa promenljivama: *let, const, var*
- Funkcije i callback funkcije
- Asinhrono programiranje (*Promises, async/await*)

Prednosti:

- Omogućava bogat i interaktivan korisnički interfejs
- Podržan u svim modernim pretraživačima
- Veliki broj biblioteka i okruženja (*React, Vue, Angular*)

Mane:

- Slaba tipizacija može izazvati greške pri razvoju većih sistema
- Izvršava se u browser-u, što može biti ranjivo ako se loše koristi

2.2.2. React

React je popularna JavaScript biblioteka za izgradnju korisničkih interfejsa, razvijena od strane Facebook-a 2013. godine. Fokusira se na razvoj komponenti koje predstavljaju delove interfejsa, čime se olakšava održavanje i organizacija koda. Koristi *JSX* – proširenje JavaScript-a koje omogućava pisanje HTML elemenata unutar JavaScript koda. Svaka React aplikacija se sastoji od komponenata koje mogu imati svoje stanje (*useState*) i efekte (*useEffect*).

Osnovni koncepti:

- Komponente (funkcionalne i klasne)
- JSX sintaksa
- Props (ulazni parametri komponente)
- State (unutrašnje stanje komponente)
- React Hooks (npr. *useState*, *useEffect*, *useContext*)
- Virtual DOM

Prednosti:

- Efikasno renderovanje i ponovno korišćenje komponenti
- Jednostavno testiranje i održavanje koda
- Bogat ekosistem dodatnih biblioteka (*React Router*, *Redux*...)

Mane:

- Veća složenost kod početnog učenja
- Brz razvoj i promena verzija može dovesti do potrebe za stalnim ažuriranjem znanja

2.3. Tehnologije za razvoj *backend* dela aplikacije

2.3.1. PHP

PHP (*Hypertext Preprocessor*) je skriptni jezik koji se izvršava na serveru i koristi se za dinamičko generisanje web stranica. Kreiran je 1994. godine, a danas je jedna od najpopularnijih tehnologija za backend razvoj.

Način korišćenja:

PHP kod se izvršava na serveru i omogućava rad sa bazama podataka, sesijama, fajlovima, kao i obradu korisničkih zahteva. Kroz Laravel framework, PHP omogućava organizovaniji i moderniji pristup programiranju.

Osnovni koncepti i tipovi:

- Tipovi: *int*, *float*, *string*, *bool*, *array*, *object*
- Klase i objekti (OOP podrška)
- Superglobalne promenljive: *\$_POST*, *\$_GET*, *\$_SESSION*, *\$_SERVER*

Prednosti:

- Brz razvoj i jednostavna integracija sa *MySQL* bazama
- Velika količina dokumentacije i dostupnih biblioteka
- Dobro podržan od strane hosting provajdera

Mane:

- Bez framework-a, kod može postati neorganizovan
- Mogućnost bezbednosnih propusta ako se loše koristi (npr. *SQL injection*)

2.3.2. Laravel

Laravel je open-source PHP framework koji omogućava brz i jednostavan razvoj web aplikacija. Baziran je na MVC (*Model-View-Controller*) arhitekturi i prvi put je objavljen 2011. godine.

Način korišćenja:

Laravel koristi jasnu sintaksu za definisanje ruta, modela, kontrolera, middleware-a i autentifikacije. Omogućava kreiranje REST API-ja, rad sa bazama putem ORM sistema (*Eloquent*), kao i generisanje koda putem *Artisan CLI* alata.

Osnovni koncepti:

- **Routing** (`Route::get()`, `Route::post()`)
- **Kontroleri i modeli** (`php artisan make:controller`, `make:model`)
- **Eloquent ORM** za komunikaciju sa bazom
- **Migracije** za verzionisanje baze (`php artisan migrate`)
- **Sanctum** za autentifikaciju korisnika putem tokena

Prednosti:

- Jasna struktura i organizacija koda
- Ugrađeni sistemi za validaciju, autentifikaciju, migraciju baze i još mnogo toga
- Aktivna zajednica i redovna ažuriranja

Mane:

- Veća početna složenost za studente koji tek ulaze u backend razvoj
- Za rad zahteva PHP znanje i razumevanje MVC koncepta

3. IMPLEMENTACIJA VEB APLIKACIJE

3.1. Faza prikupljanja korisničkih zahteva

3.1.1. Verbalni opis

U okviru ovog projekta razvijena je veb aplikacija za prodaju digitalnih umetničkih proizvoda. Aplikacija je namenjena korisnicima koji žele da pregledaju i kupuju digitalne proizvode, kao i administratorima koji upravljaju sadržajem sajta i porudžbinama.

U sistemu postoje tri vrste korisnika:

- **Neulogovani korisnici** – mogu da pregledaju ponudu proizvoda, pretražuju galeriju i vide osnovne informacije o svakom delu. Za kupovinu i preuzimanje fajlova neophodna je registracija.
- **Registrovani (ulogovani) korisnici** – imaju mogućnost da dodaju proizvode u korpu, izvrše kupovinu i nakon update pristupe digitalnim fajlovima za preuzimanje. Takođe imaju pristup svom korisničkom profilu, gde mogu pratiti istoriju porudžbina.
- **Admin** – ima pristup posebnom administrativnom panelu gde može dodavati, menjati i brisati proizvode, kao i upravljati kategorijama. Takođe ima uvid u sve porudžbine i osnovnu analitiku sajta, uključujući pregled najaktivnijih kupaca. Ova uloga je ograničena i dostupna isključivo ovlašćenim korisnicima.

Aplikacija je razvijena korišćenjem Laravel framework-a za serversku stranu, dok je korisnički interfejs realizovan pomoću React biblioteke. Komunikacija između klijentske i serverske strane odvija se putem REST API-ja, čime je omogućena jasna i modularna razmena podataka. Sistem uključuje korisničku autentikaciju, validaciju ulaznih podataka, upravljanje sesijama i zaštitu pristupa funkcionalnostima u skladu sa korisničkom ulogom.

3.1.2. Slučajevi korišćenja

Primeri slučajeva korišćenja:

SK1 – Kreiranje slika

SK2 – Izmena slika

SK3 – Brisanje slika

SK4 – Pretraga slika

SK5 – Filtriranje slika

SK6 – Kreiranje kategorije

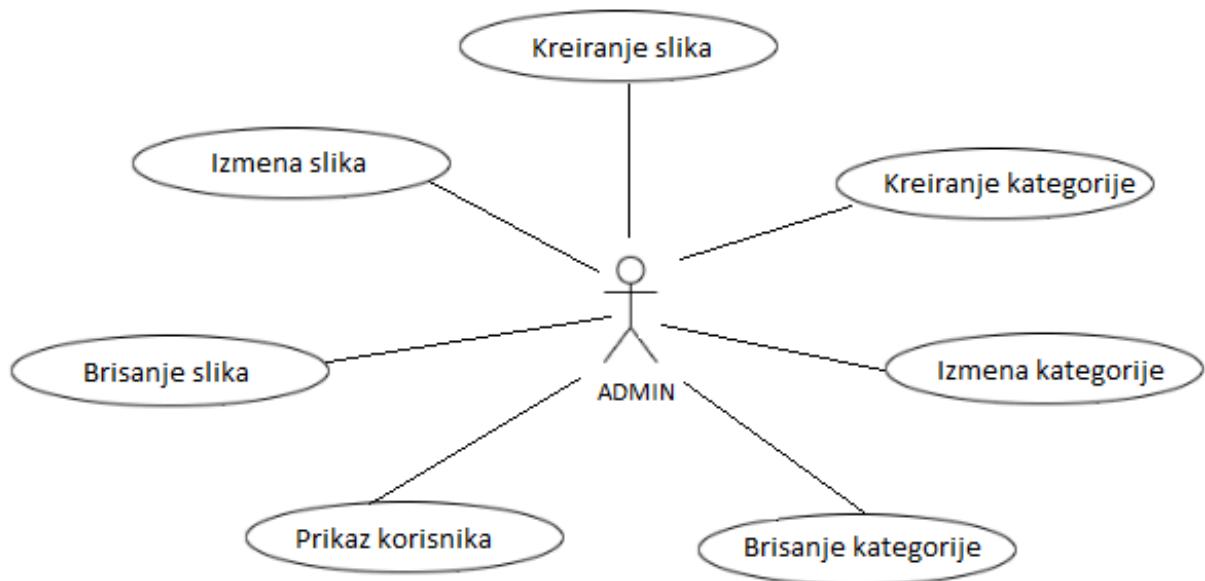
SK7 – Izmena kategorije

SK8 – Brisanje kategorije

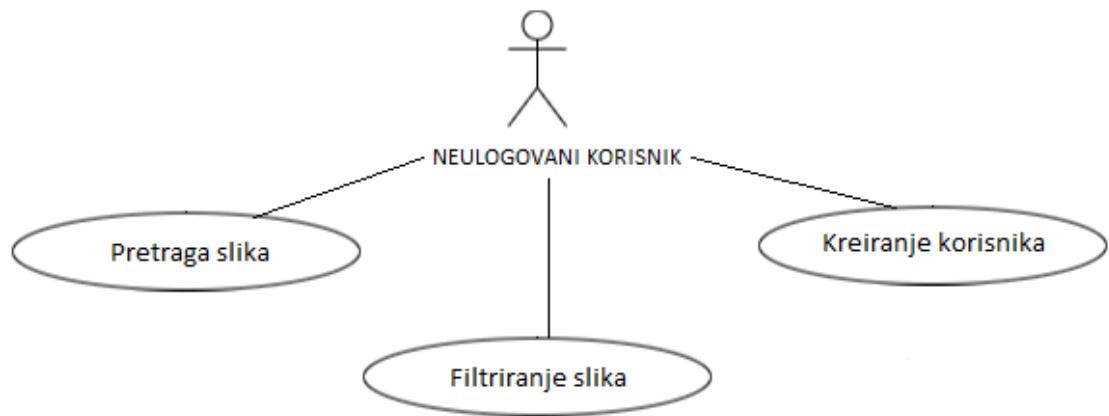
SK9 – Kreiranje korisnika

SK10 – Prikaz korisnika

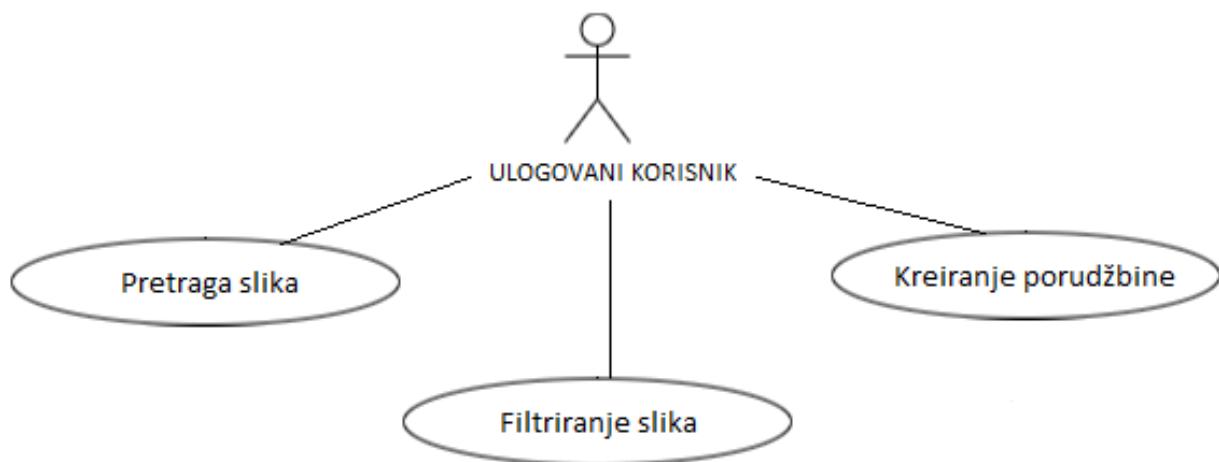
SK11 – Kreiranje porudžbine



Slika 1 - Slučajevi korišćenja admina



Slika 2 - Slučajevi korišćenja neulogovanog korisnika



Slika 3 - Slučajevi korišćenja ulogovanog korisnika

SK1: Slučaj korišćenja – Kreiranje slika

Naziv SK

Kreiranje slika

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov: Sistem je uključen i administrator je ulogovan sa svojim nalogom. Otvoren je administratorski panel i prikazana je forma za dodavanje nove slike.

Osnovni scenario SK:

1. Administrator **unosi** podatke o novoj slici (naziv, opis, kategorija, cena). (*APUSO*)
2. Administrator **proverava** da li su podaci ispravno uneti. (*ANSO*)
3. Administrator **Šalje** zahtev sistemu da sačuva novu sliku. (*APSO*)
4. Sistem **obrađuje** zahtev i pamti novu sliku u bazi podataka. (*SO*)
5. Sistem **prikazuje** administratoru poruku: „Slika je uspešno kreirana!“ (*IA*)

Alternativni scenario:

3.1 Ako administrator nije popunio sva obavezna polja sistem prikazuje poruku: „Popunite sva obavezna polja.“ (*IA*)

SK2: Slučaj korišćenja – Izmena slika

Naziv SK

Izmena slika

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov: Sistem je uključen, administrator je ulogovan, i u administrativnom panelu je prikazana lista postojećih slika.

Osnovni scenario SK:

1. Administrator **bira** sliku za izmenu. (*APUSO*)
2. Administrator **poziva** sistem da učita podatke o odabranoj slici. (*APSO*)
3. Sistem **učitava** podatke o slici. (*SO*)
4. Sistem **prikazuje** administratoru podatke o slici. (*IA*)
5. Administrator **menja** podatke o slici. (*APUSO*)
6. Administrator **kontroliše** da li je korektno uneo (izmenio) podatke. (*ANSO*)
7. Administrator **šalje** izmene sistemu. (*APSO*)
8. Sistem **pamti** izmene u bazi podataka. (*SO*)
9. Sistem **prikazuje** poruku o uspešnom čuvanju. (*IA*)

Alternativni scenariji:

6.1 Ako administrator nije popunio sva obavezna polja sistem prikazuje poruku: „Popunite sva obavezna polja.” i vraća administratora na formu za izmenu. (*IA*)

7.1 Ako sistem ne može da sačuva izmene (zbog tehničkih problema), prikazuje se poruka o grešci i administrator može da pokuša ponovo. (*IA*)

SK3: Slučaj korišćenja – Brisanje slika

Naziv SK

Brisanje slika

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov: Sistem je uključen, administrator je ulogovan, i u administrativnom panelu je prikazana lista postojećih slika.

Osnovni scenario SK:

1. Administrator **bira** sliku koju želi da obriše. (*APUSO*)
2. Sistem **učitava** podatke o odabranoj slici. (*SO*)
3. Sistem **prikazuje** administratoru podatke o slici. (*IA*)
4. Administrator **potvrđuje** brisanje slike. (*ANSO*)
5. Administrator **šalje** zahtev za brisanje slike. (*APSO*)
6. Sistem **briše** sliku iz baze podataka. (*SO*)
7. Sistem **prikazuje** administratoru poruku: „Slika je uspešno obrisana.“ (*IA*)

Alternativni scenariji:

4.1 Ako administrator odustane od brisanja slike, sistem prekida izvršenje scenarija bez promene podataka. (*IA*)

6.1 Ako dođe do greške pri brisanju (npr. baza nije dostupna), sistem prikazuje poruku: „Greška, sistem ne može da obriše sliku.“ Izvršavanje se prekida. (*IA*)

SK4: Slučaj korišćenja – Pretraga slika

Naziv SK

Pretraga slika

Aktori SK

Neulogovani i ulogovani korisnici

Učesnici SK

Korisnik i sistem

Preduslov: Sistem je uključen, korisnik pristupa stranici sa prikazom ponude slika.

Osnovni scenario SK:

1. Korisnik **unosi** pojam za pretragu u pretraživač. (*APUSO*)
2. Korisnik **pokreće** pretragu. (*APSO*)
3. Sistem **prima** zahtev i pretražuje bazu podataka. (*SO*)
4. Sistem **prikazuje** korisniku rezultate koji odgovaraju unetim kriterijumima. (*IA*)
5. Korisnik **pregleda** prikazane rezultate. (*APUSO*)

Alternativni scenario:

3.1 Ako ne postoji nijedna slika koja odgovara unetim kriterijumima, sistem prikazuje poruku: „Nema rezultata za unetu pretragu.“ (*IA*)

SK5: Slučaj korišćenja – Filtriranje slika

Naziv SK

Filtriranje slika

Aktori SK

Neulogovani i ulogovani korisnici

Učesnici SK

Korisnik i sistem

Preduslov: Sistem je uključen. Korisnik se nalazi na stranici sa prikazom svih dostupnih slika i vidi dropdown meni za filtriranje po kategorijama.

Osnovni scenario SK:

1. Korisnik **otvara** dropdown meni sa listom kategorija. (*APUSO*)
2. Korisnik **bira** željenu kategoriju klikom na nju. (*APUSO*)
3. Sistem automatski **izvršava** filtriranje slika na osnovu odabrane kategorije. (*SO*)
4. Sistem **prikazuje** korisniku slike koje odgovaraju izabranoj kategoriji. (*IA*)
5. Korisnik **pregleda** prikazane rezultate. (*APUSO*)

Alternativni scenario:

3.1 Ako ne postoji nijedna slika koja odgovara izabranim kriterijumima, sistem prikazuje poruku: „Ne postoji slika koja odgovara izabranom kriterijumu.“ (*IA*)

SK6: Slučaj korišćenja – Kreiranje kategorije

Naziv SK

Kreiranje kategorije

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov: Sistem je uključen. Administrator je ulogovan i nalazi se u administrativnom panelu gde ima pristup sekciji za dodavanje kategorija.

Osnovni scenario SK:

1. Administrator **unosi** naziv i opis nove kategorije. (*APUSO*)
2. Administrator **proverava** da li su podaci ispravno uneti. (*ANSO*)
3. Administrator **klikne** na dugme *Create category* kako bi kreirao novu kategoriju. (*APSO*)
4. Sistem validira unos i **čuva** novu kategoriju u bazi podataka. (*SO*)
5. Sistem **prikazuje** poruku: „Kategorija je uspešno dodata.“ (*IA*)

Alternativni scenariji:

3.1 Ako administrator ne popuni obavezno polje, sistem prikazuje poruku: „Popunite ovo polje.“ i ne izvršava kreiranje. (*IA*)

4.1 Ako kategorija sa istim nazivom već postoji, sistem izbacuje grešku i ne kreira kategoriju. (*IA*)

SK7: Slučaj korišćenja – Izmena kategorije

Naziv SK

Izmena kategorije

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov:

Sistem je uključen, administrator je ulogovan i nalazi se u administrativnom panelu sa prikazom liste postojećih kategorija.

Osnovni scenario SK:

1. Administrator bira kategoriju za izmenu. (APUSO)
2. Sistem učitava podatke o odabranoj kategoriji. (SO)
3. Sistem prikazuje administratoru podatke o kategoriji. (IA)
4. Administrator menja podatke o kategoriji. (APUSO)
5. Administrator proverava izmene. (ANSO)
6. Administrator potvrđuje i šalje izmene sistemu. (APSO)
7. Sistem čuva izmene u bazi. (SO)
8. Sistem prikazuje poruku: „Kategorija je uspešno izmenjena.“ (IA)

Alternativni scenario:

6.1 Ako se desi greška prilikom čuvanja, sistem prikazuje poruku o grešci i ne menja podatke. (IA)

SK8: Slučaj korišćenja – Brisanje kategorije

Naziv SK

Brisanje kategorije

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov:

Sistem je uključen, administrator je ulogovan i nalazi se na stranici sa listom kategorija.

Osnovni scenario SK:

1. Administrator bira kategoriju koju želi da obriše. (APUSO)
2. Sistem prikazuje informacije o izabranoj kategoriji. (IA)
3. Administrator potvrđuje brisanje. (ANSO)
4. Administrator šalje zahtev za brisanje. (APSO)
5. Sistem briše kategoriju iz baze podataka. (SO)
6. Sistem prikazuje poruku: „Kategorija je uspešno obrisana.“ (IA)

Alternativni scenariji:

3.1 Ako administrator odustane od brisanja, sistem prekida akciju bez promene podataka. (IA)

5.1 Ako dođe do greške, sistem prikazuje poruku: „Greška prilikom brisanja kategorije.“ (IA)

SK9: Slučaj korišćenja – Kreiranje korisnika

Naziv SK

Kreiranje korisnika

Aktori SK

Neulogovani korisnik

Učesnici SK

Korisnik i sistem

Preduslov:

Korisnik pristupa stranici za registraciju. Sistem je dostupan.

Osnovni scenario SK:

1. Korisnik otvara stranicu za registraciju. (APUSO)
2. Korisnik unosi podatke potrebne za registraciju (ime, email, lozinka). (APUSO)
3. Korisnik proverava da li je sve ispravno uneo. (ANSO)
4. Korisnik šalje zahtev za registraciju. (APSO)
5. Sistem proverava da li su svi podaci validni i da li email već postoji. (SO)
6. Sistem čuva podatke u bazi i kreira novog korisnika. (SO)
7. Sistem prikazuje poruku: „Uspešno ste se registrovali.“ (IA)

Alternativni scenariji:

- 5.1 Ako je unet email već registrovan, sistem prikazuje poruku: „Korisnik sa ovim emailom već postoji.“ (IA)
- 5.2 Ako neko od obaveznih polja nije popunjeno, sistem prikazuje poruku: „Popunite sva obavezna polja.“ (IA)
- 5.3 Ako lozinka ne zadovoljava bezbednosne kriterijume, sistem prikazuje poruku: „Lozinka mora imati najmanje 8 karaktera i sadržati bar jedno veliko slovo i broj.“ (IA)

SK10: Slučaj korišćenja – Prikaz korisnika

Naziv SK

Prikaz korisnika

Aktori SK

Administrator

Učesnici SK

Administrator i sistem

Preduslov:

Administrator je ulogovan i ima pristup stranici sa listom korisnika.

Osnovni scenario SK:

1. Administrator otvara stranicu za pregled korisnika. (APUSO)
2. Sistem učitava sve korisnike iz baze. (SO)
3. Sistem prikazuje listu korisnika. (IA)
4. Administrator pregleda detalje korisnika. (APUSO)

Alternativni scenario:

- 2.1 Ako ne postoji nijedan korisnik u sistemu, prikazuje se poruka: „Nema korisnika za prikaz.“ (IA)

SK11: Slučaj korišćenja – Kreiranje porudžbine

Naziv SK

Kreiranje porudžbine

Aktori SK

Registrovani korisnik

Učesnici SK

Korisnik i sistem

Preduslov:

Korisnik je ulogovan i ima jednu ili više slika u korpi.

Osnovni scenario SK:

1. Korisnik otvara korpu i pregleda slike za kupovinu. (APUSO)
2. Korisnik unosi potrebne informacije za plaćanje (npr. ime i prezime, broj kartice, datum isteka, CVV kod). (APUSO)
3. Korisnik proverava unete podatke o plaćanju i potvrđuje porudžbinu. (ANSO)
4. Sistem validira podatke i kreira porudžbinu u bazi. (SO)
5. Sistem prikazuje poruku: „Porudžbina je uspešno kreirana.“ (IA)

Alternativni scenariji:

2.1 Ako je korpa prazna, sistem prikazuje poruku: „Korpa je prazna, dodajte slike pre porudžbine.“ (IA)

2.2 Ako neki od podataka za plaćanje nije unet ili nije validan, sistem prikazuje poruku: „Unesite validne podatke za plaćanje.“ (IA)

4.1 Ako dođe do greške prilikom kreiranja porudžbine, sistem prikazuje poruku: „Greška prilikom kreiranja porudžbine.“ (IA)

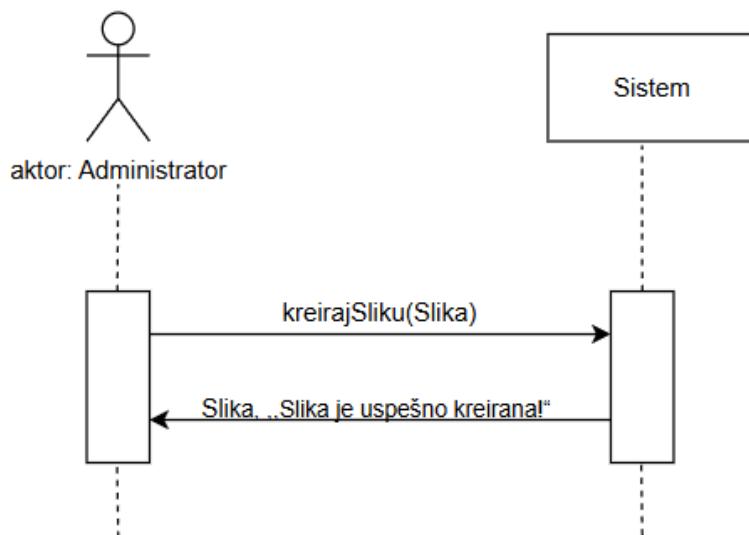
3.2 Faza analize

3.2.1. Sistemski dijagrami sekvence

DS1: Dijagram sekvence slučaja korišćenja - Kreiranje slika

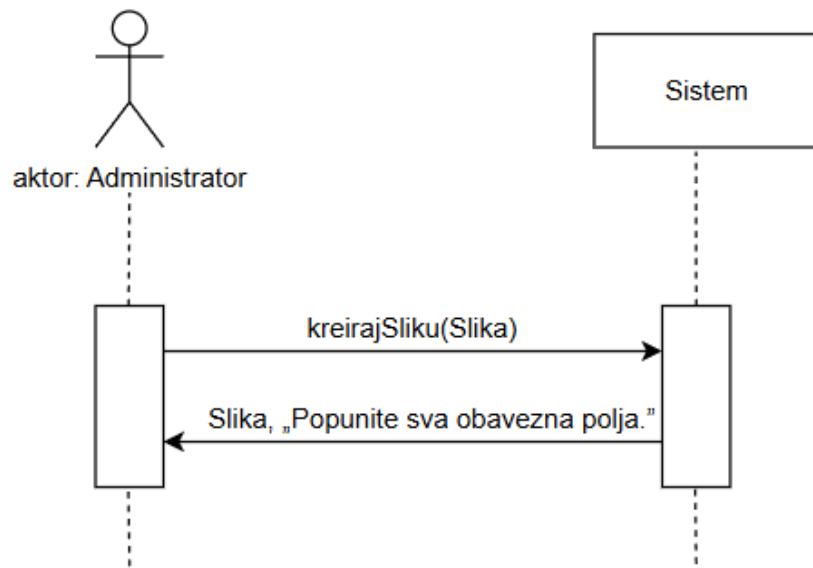
Osnovni scenario SK:

1. Administrator **unosi** podatke o novoj slici (naziv, opis, kategorija, cena). (*APUSO*)
2. Administrator **proverava** da li su podaci ispravno uneti. (*ANSO*)
3. Administrator **šalje** zahtev sistemu da sačuva novu sliku. (*APSO*)
4. Sistem **obrađuje** zahtev i pamti novu sliku u bazi podataka. (*SO*)
5. Sistem **prikazuje** administratoru poruku: „Slika je uspešno kreirana!“ (*IA*)



Alternativni scenario:

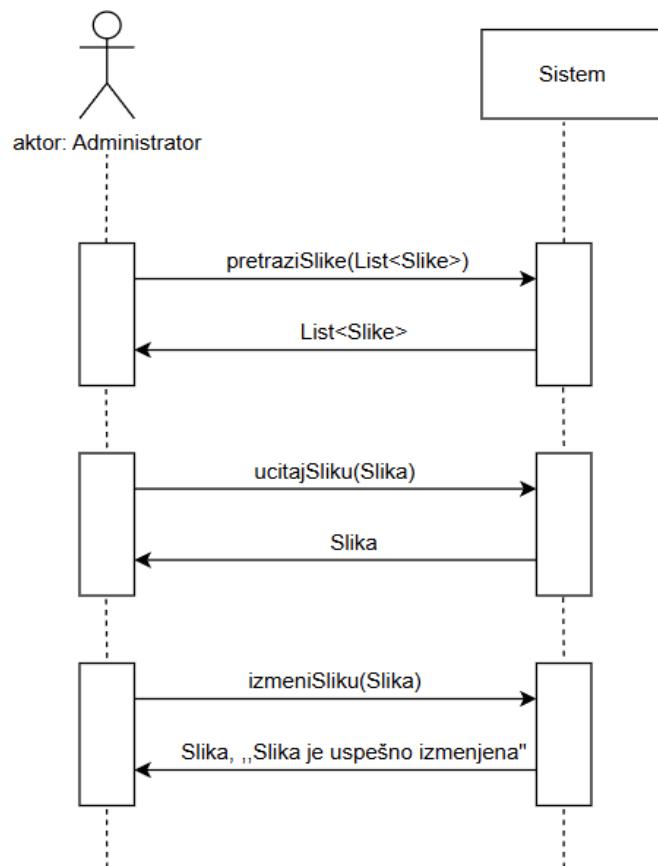
3.1 Ako administrator nije popunio sva obavezna polja sistem prikazuje poruku: „Popunite sva obavezna polja.” (IA)



DS2: Dijagram sekvence slučaja korišćenja – Izmena slika

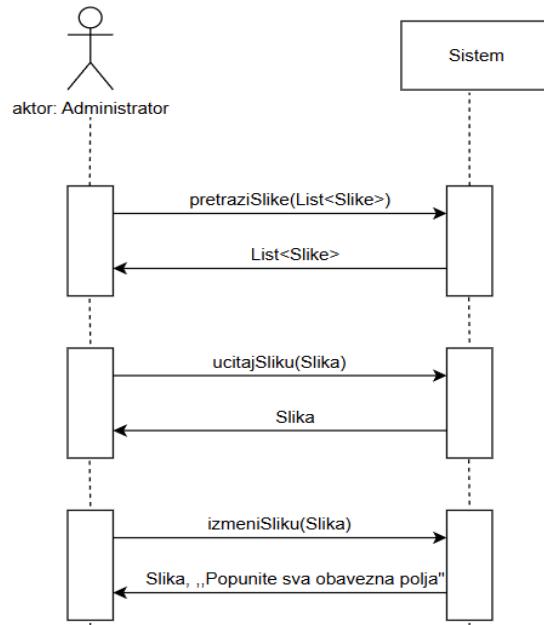
Osnovni scenario SK:

1. Administrator **bira** sliku za izmenu. (APUSO)
2. Administrator **poziva** sistem da učita podatke o odabranoj slici. (APSO)
3. Sistem **učitava** podatke o slici. (SO)
4. Sistem **prikazuje** administratoru podatke o slici. (IA)
5. Administrator **menja** podatke o slici. (APUSO)
6. Administrator **kontroliše** da li je korektno uneo (izmenio) podatke. (ANSO)
7. Administrator **šalje** izmene sistemu. (APSO)
8. Sistem **pamti** izmene u bazi podataka. (SO)
9. Sistem **prikazuje** poruku o uspešnom čuvanju. (IA)

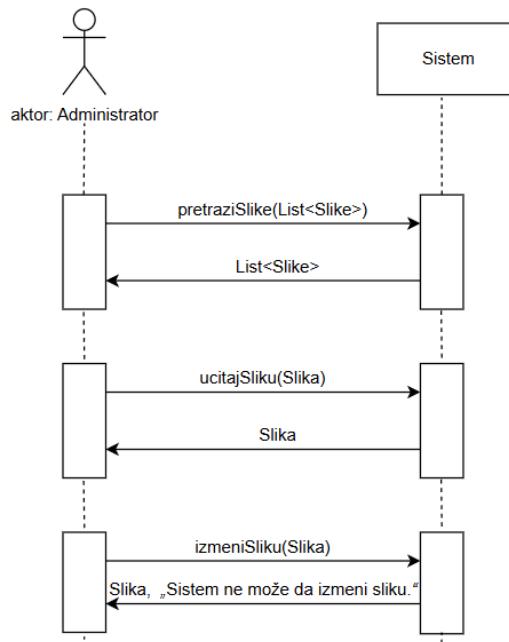


Alternativni scenariji:

6.1 Ako administrator nije popunio sva obvezna polja sistem prikazuje poruku: „Popunite sva obvezna polja.” i vraća administratora na formu za izmenu. (IA)



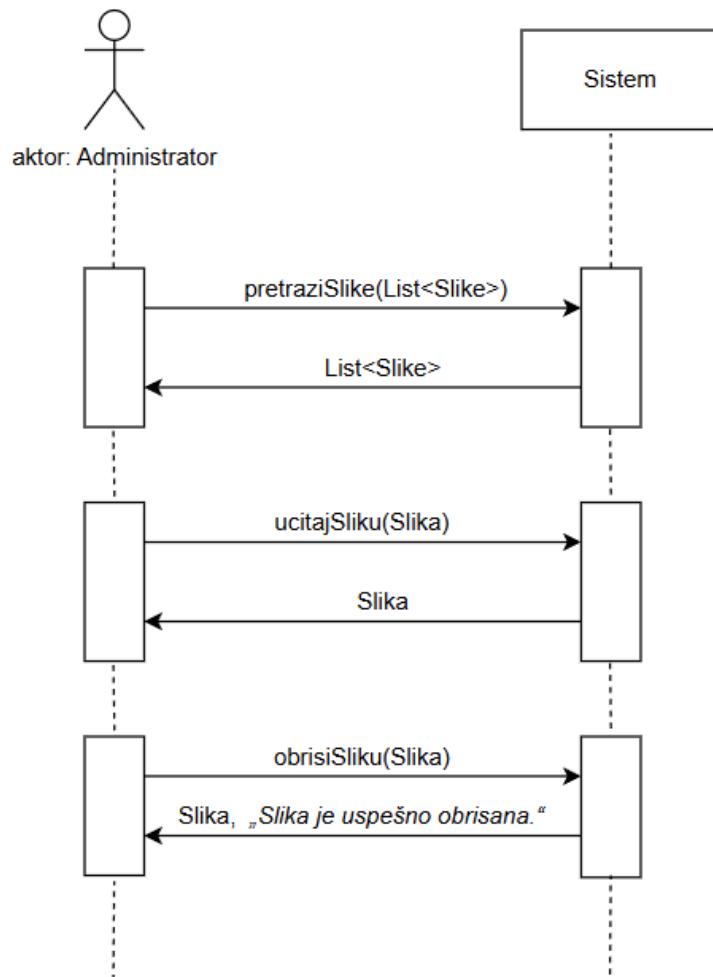
7.1 Ako sistem ne može da sačuva izmene (zbog tehničkih problema), prikazuje se poruka o grešci i administrator može da pokuša ponovo. (IA)



DS3: Dijagram sekvence slučaja korišćenja – Brisanje slike

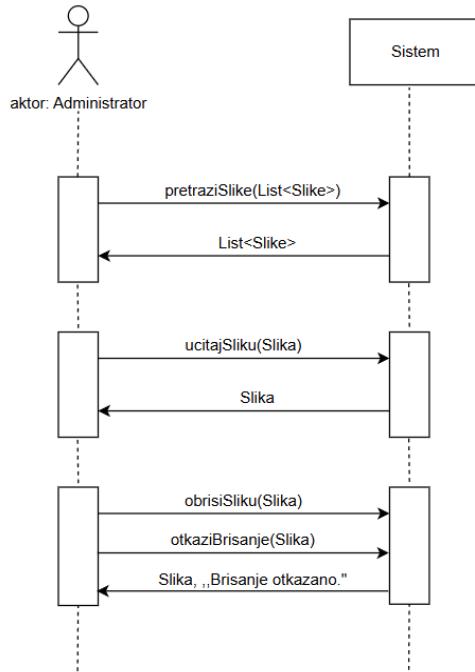
Osnovni scenario SK:

1. Administrator **bira** sliku koju želi da obriše. (APUSO)
2. Sistem **učitava** podatke o odabranoj slici. (SO)
3. Sistem **prikazuje** administratoru podatke o slici. (IA)
4. Administrator **potvrđuje** brisanje slike. (ANSO)
5. Administrator **šalje** zahtev za brisanje slike. (APSO)
6. Sistem **briše** sliku iz baze podataka. (SO)
7. Sistem **prikazuje** administratoru poruku: „Slika je uspešno obrisana.“ (IA)



Alternativni scenario:

4.1 Ako administrator odustane od brisanja slike, sistem prekida izvršenje scenarija bez promene podataka. (IA)

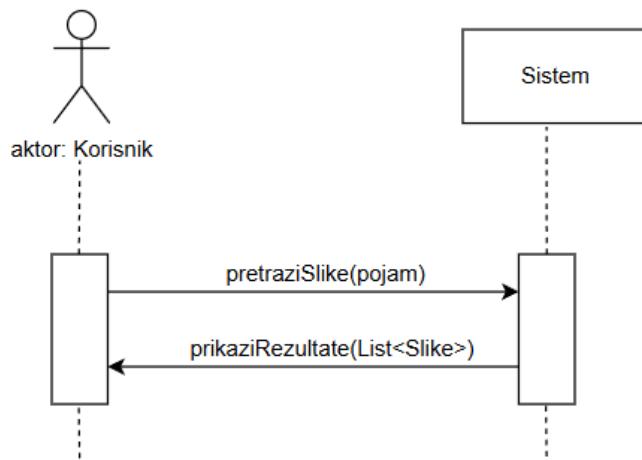


6.1 Ako dođe do greške pri brisanju (npr. baza nije dostupna), sistem prikazuje poruku: „Greška, sistem ne može da obriše sliku.“ Izvršavanje se prekida. (IA)

DS4: Dijagram sekvence slučaja korišćenja- Pretraga slika

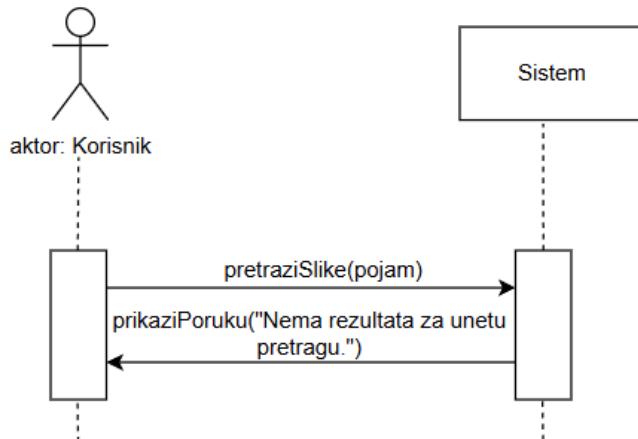
Osnovni scenario SK:

1. Korisnik **unosi** pojam za pretragu u pretraživač. (*APUSO*)
2. Korisnik **pokreće** pretragu. (*APSO*)
3. Sistem **prima** zahtev i pretražuje bazu podataka. (*SO*)
4. Sistem **prikazuje** korisniku rezultate koji odgovaraju unetim kriterijumima. (*IA*)
5. Korisnik **pregleda** prikazane rezultate. (*APUSO*)



Alternativni scenario:

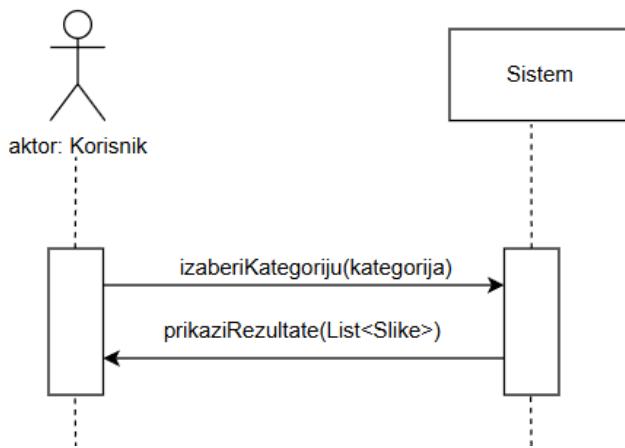
- 3.1 Ako ne postoji nijedna slika koja odgovara unetim kriterijumima, sistem prikazuje poruku: „Nema rezultata za unetu pretragu.“ (*IA*)



DS5: Dijagram sekvence slučaja korišćenja – Filtriranje slika

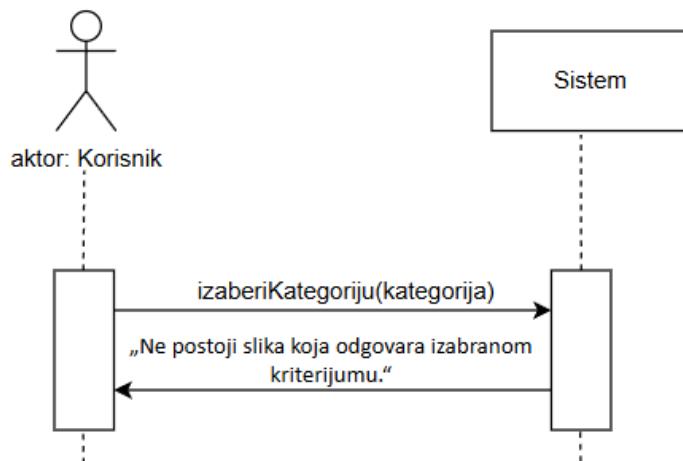
Osnovni scenario SK:

1. Korisnik **otvara** dropdown meni sa listom kategorija. (*APUSO*)
2. Korisnik **bira** željenu kategoriju klikom na nju. (*APUSO*)
3. Sistem automatski **izvršava** filtriranje slika na osnovu odabrane kategorije. (*SO*)
4. Sistem **prikazuje** korisniku slike koje odgovaraju izabranoj kategoriji. (*IA*)
5. Korisnik **pregleda** prikazane rezultate. (*APUSO*)



Alternativni scenario:

3.1 Ako ne postoji nijedna slika koja odgovara izabranim kriterijumima, sistem prikazuje poruku: „Ne postoji slika koja odgovara izabranom kriterijumu.“ (*IA*)



DS6: Dijagram sekvence slučaja korišćenja - Kreiranje kategorije

Osnovni scenario SK:

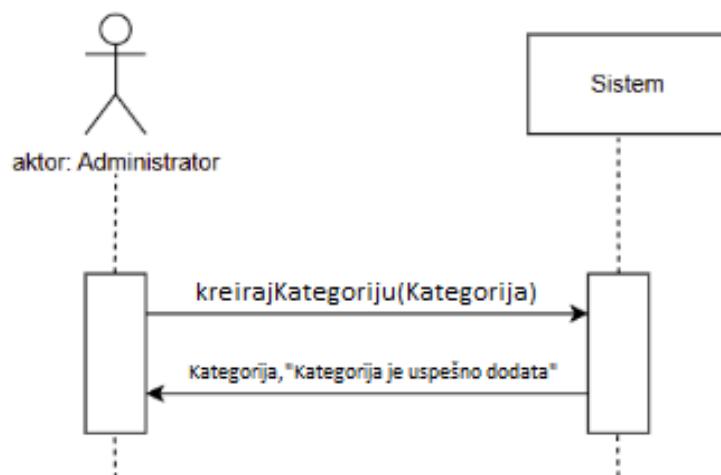
Administrator **unosi** naziv i opis nove kategorije. (APUSO)

Administrator **proverava** da li su podaci ispravno uneti. (ANSO)

Administrator **klikne** na dugme *Create category* kako bi kreirao novu kategoriju. (APSO)

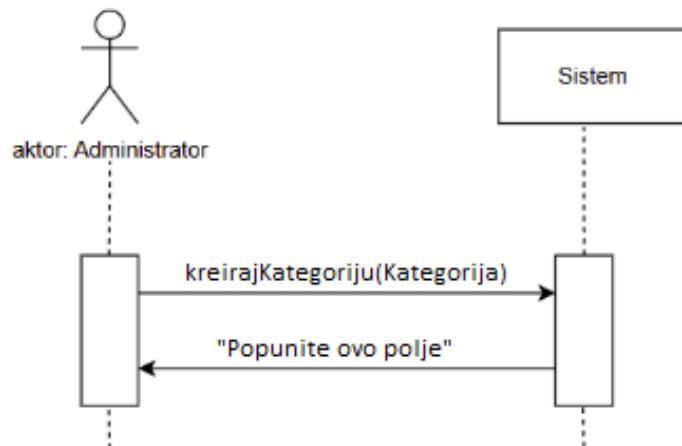
Sistem validira unos i **čuva** novu kategoriju u bazi podataka. (SO)

Sistem **prikazuje** poruku: „Kategorija je uspešno dodata.“ (IA)

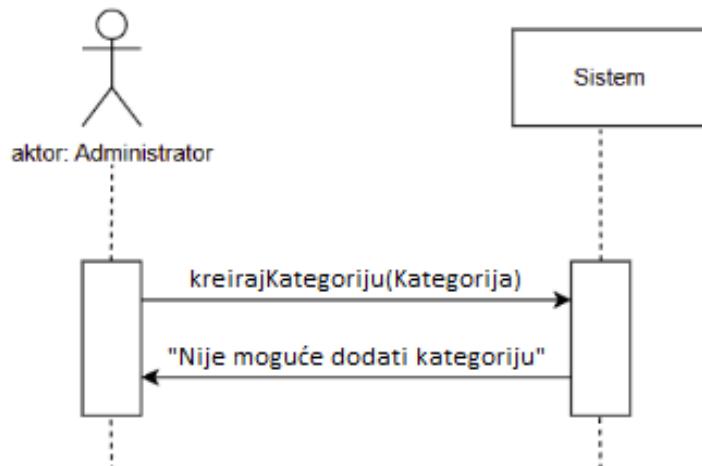


Alternativni scenariji:

3.1 Ako administrator ne popuni obavezno polje, sistem prikazuje poruku: „Popunite ovo polje.“ i ne izvršava kreiranje. (IA)



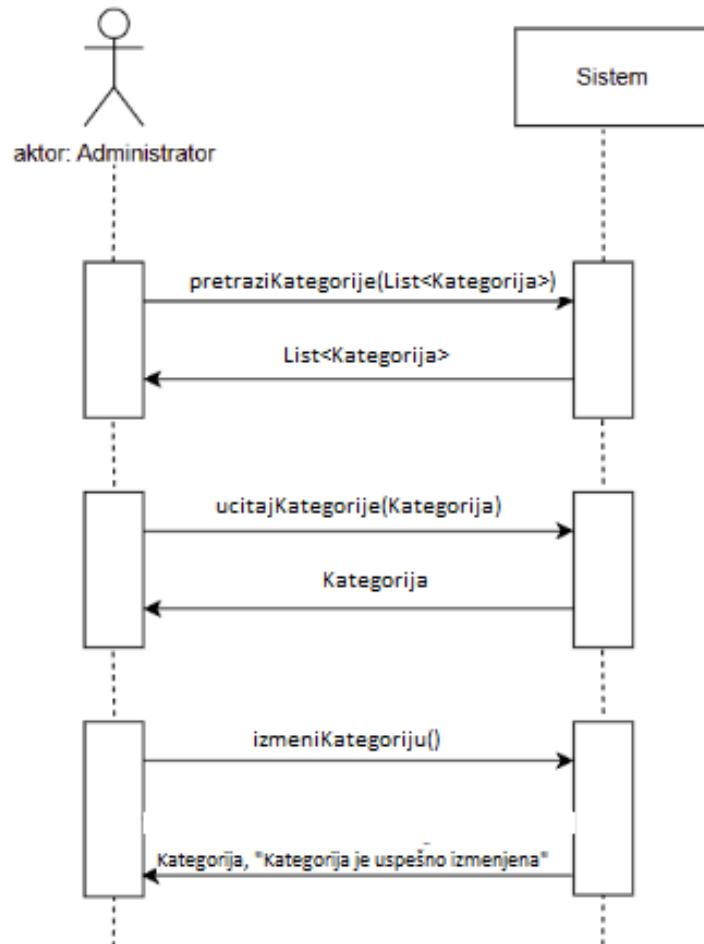
4.1 Ako kategorija sa istim nazivom već postoji, sistem izbacuje grešku i ne kreira kategoriju. (IA)



DS7: Dijagram sekvence slučaja korišćenja - Izmena kategorije

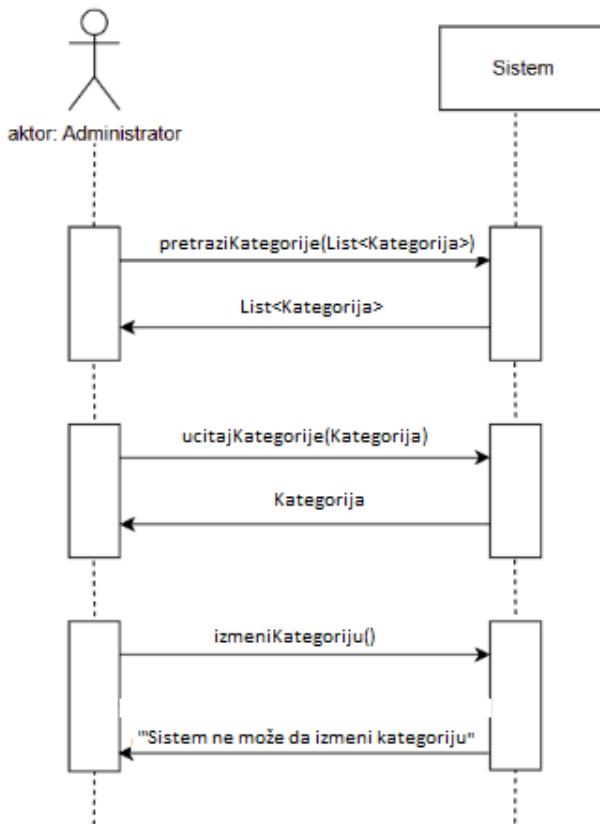
Osnovni scenario SK:

1. Administrator **bira** kategoriju za izmenu. (APUSO)
2. Sistem **učitava** podatke o odabranoj kategoriji. (SO)
3. Sistem **prikazuje** administratoru podatke o kategoriji. (IA)
4. Administrator **menja** podatke o kategoriji. (APUSO)
5. Administrator **proverava** izmene. (ANSO)
6. Administrator **potvrđuje** i šalje izmene sistemu. (APSO)
7. Sistem **čuva** izmene u bazi. (SO)
8. Sistem prikazuje poruku: „Kategorija je uspešno izmenjena.“ (IA)



Alternativni scenariji:

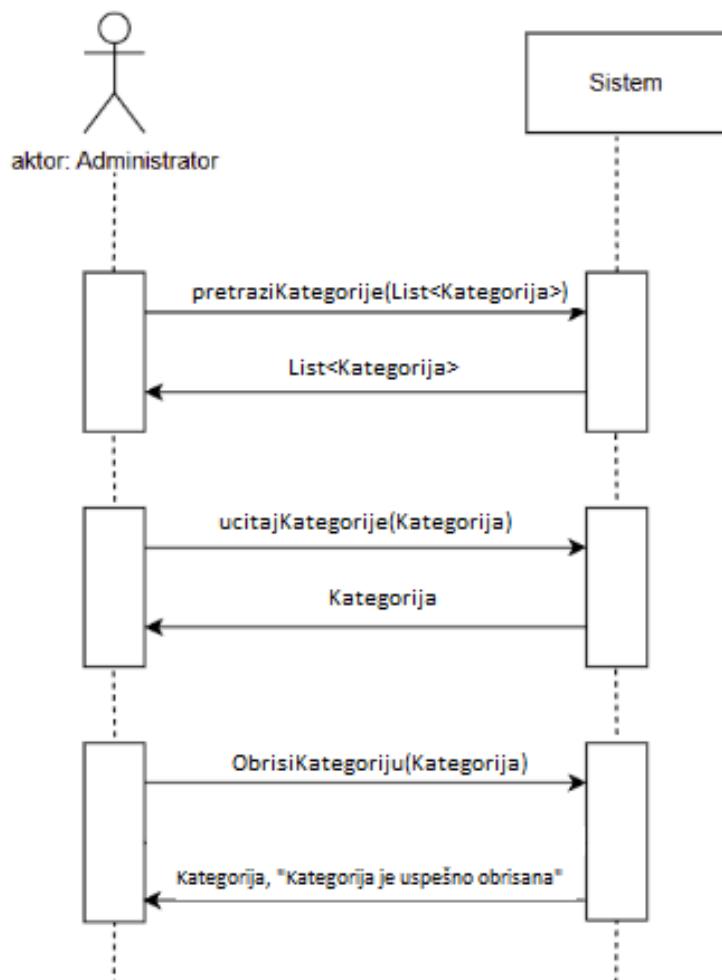
6.1 Ako se desi greška prilikom čuvanja, sistem prikazuje poruku o grešci i ne menja podatke.
(IA)



DS8: Dijagram sekvence slučaja korišćenja - Brisanje kategorije

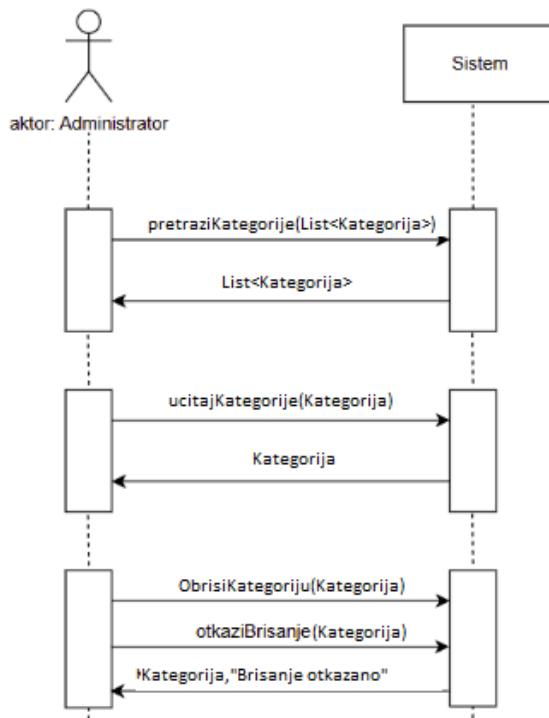
Osnovni scenario SK:

1. Administrator **bira** kategoriju koju želi da obriše. (APUSO)
2. Sistem **prikazuje** informacije o izabranoj kategoriji. (IA)
3. Administrator **potvrđuje** brisanje. (ANSO)
4. Administrator **šalje** zahtev za brisanje. (APSO)
5. Sistem **briše** kategoriju iz baze podataka. (SO)
6. Sistem **prikazuje** poruku: „Kategorija je uspešno obrisana.“ (IA)

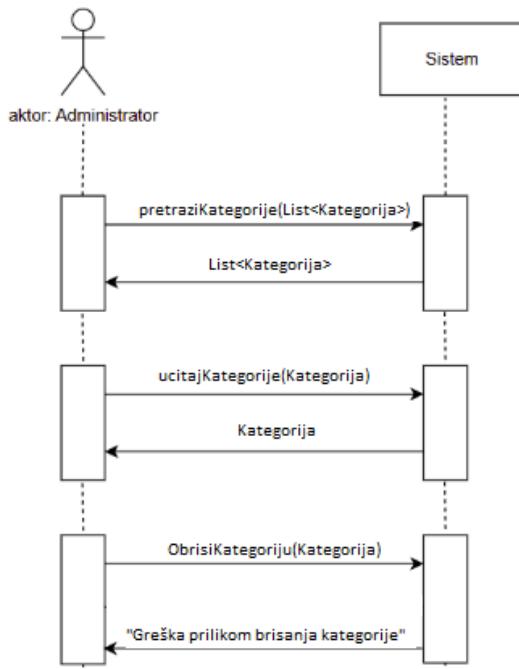


Alternativni scenariji:

3.1 Ako administrator odustane od brisanja, sistem prekida akciju bez promene podataka. (IA)



5.1 Ako dođe do greške, sistem prikazuje poruku: „Greška prilikom brisanja kategorije.“ (IA)



DS9: Dijagram sekvence slučaja korišćenja - Kreiranje korisnika

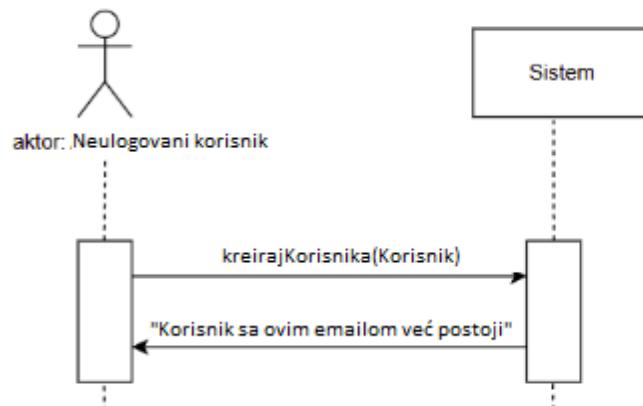
Osnovni scenario SK:

1. Korisnik **otvara** stranicu za registraciju. (APUSO)
2. Korisnik **unosi** podatke potrebne za registraciju (ime, email, lozinka). (APUSO)
3. Korisnik **proverava** da li je sve ispravno uneo. (ANSO)
4. Korisnik **šalje** zahtev za registraciju. (APSO)
5. Sistem **proverava** da li su svi podaci validni i da li email već postoji. (SO)
6. Sistem **čuva** podatke u bazi i **kreira** novog korisnika. (SO)
7. Sistem **prikazuje** poruku: „Uspešno ste se registrovali.“ (IA)

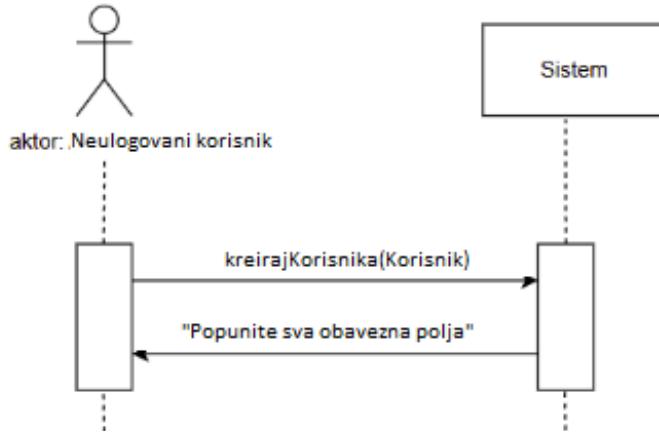


Alternativni scenariji:

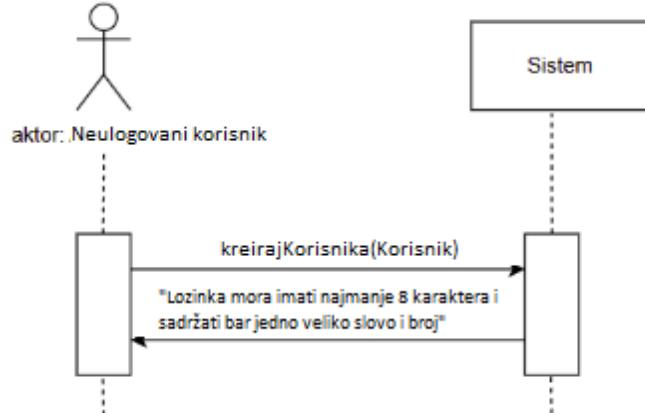
- 5.1 Ako je unet email već registrovan, sistem prikazuje poruku: „Korisnik sa ovim emailom već postoji.“ (IA)



5.2 Ako neko od obveznih polja nije popunjeno, sistem prikazuje poruku: „Popunite sva obavezna polja.“ (IA)



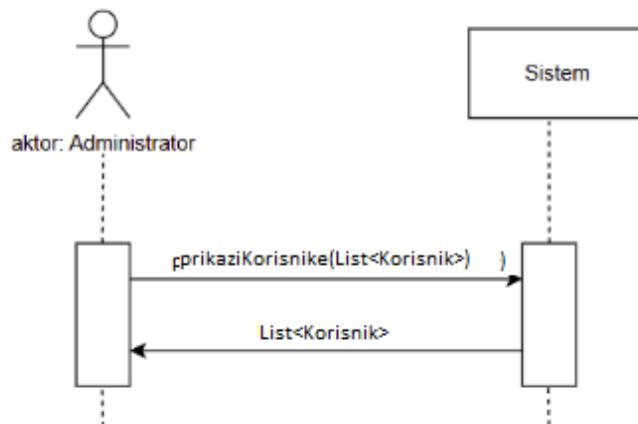
5.3 Ako lozinka ne zadovoljava bezbednosne kriterijume, sistem prikazuje poruku: „Lozinka mora imati najmanje 8 karaktera i sadržati bar jedno veliko slovo i broj.“ (IA)



DS10: Dijagram sekvence slučaja korišćenja - Prikaz korisnika

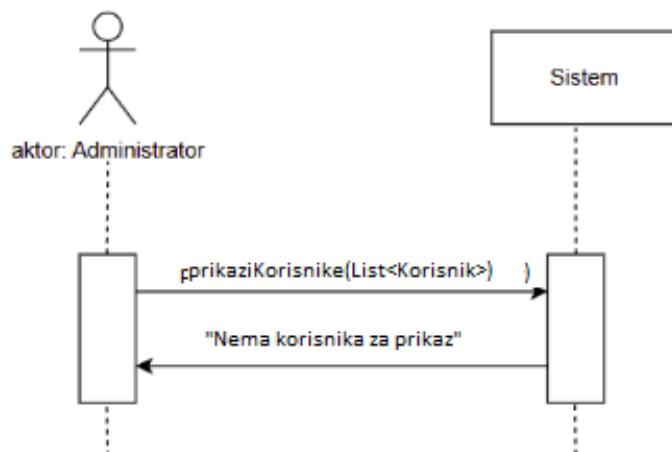
Osnovni scenario SK:

1. Administrator **otvara** stranicu za pregled korisnika. (APUSO)
2. Sistem **učitava** sve korisnike iz baze. (SO)
3. Sistem **prikazuje** listu korisnika. (IA)
4. Administrator **pregleda** detalje korisnika. (APUSO)



Alternativni scenario:

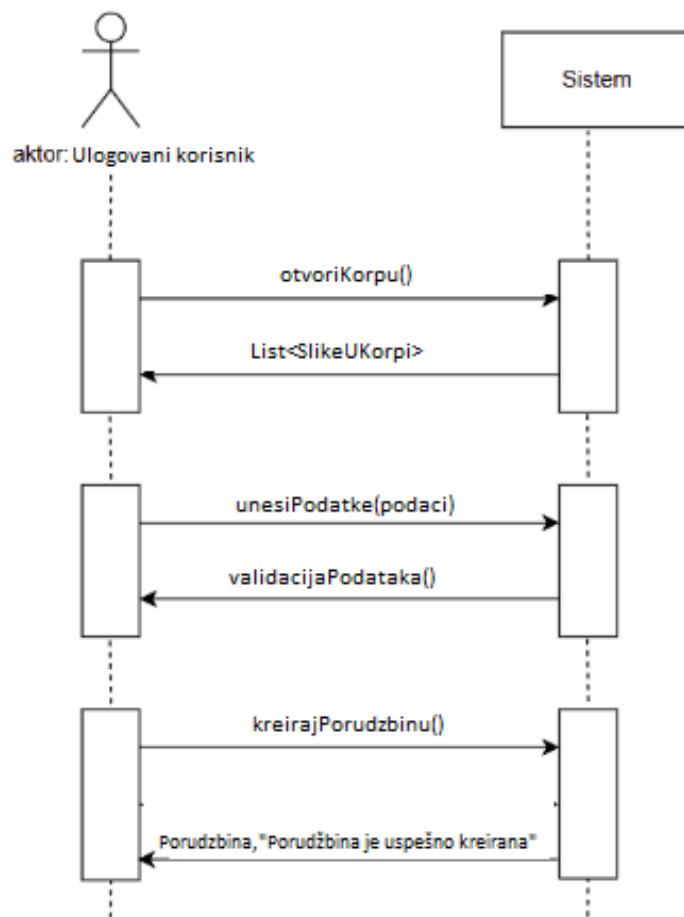
- 2.1 Ako ne postoji nijedan korisnik u sistemu, prikazuje se poruka: „Nema korisnika za prikaz.“ (IA)



DS11: Dijagram sekvence slučaja korišćenja - Kreiranje porudžbine

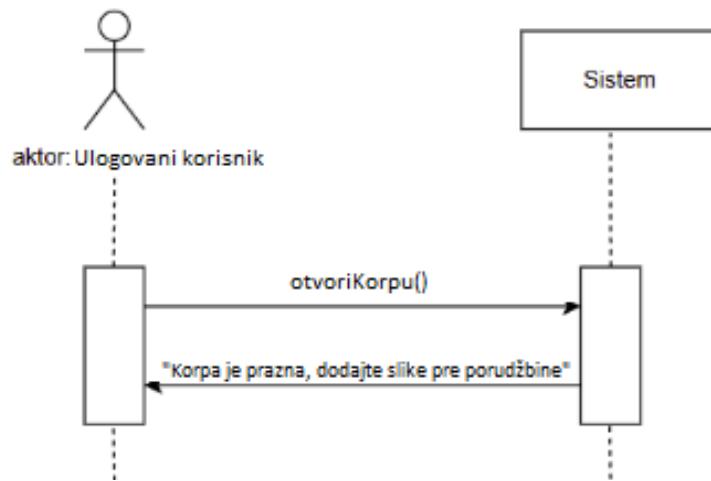
Osnovni scenario SK:

1. Korisnik **otvara** korpu i **pregleda** slike za kupovinu. (APUSO)
2. Korisnik **unosi** potrebne informacije za plaćanje (npr. ime i prezime, broj kartice, datum isteka, CVV kod). (APUSO)
3. Korisnik **proverava** unete podatke o plaćanju i potvrđuje porudžbinu. (ANSO)
4. Sistem **validira** podatke i **kreira** porudžbinu u bazi. (SO)
5. Sistem **prikazuje** poruku: „Porudžbina je uspešno kreirana.“ (IA)

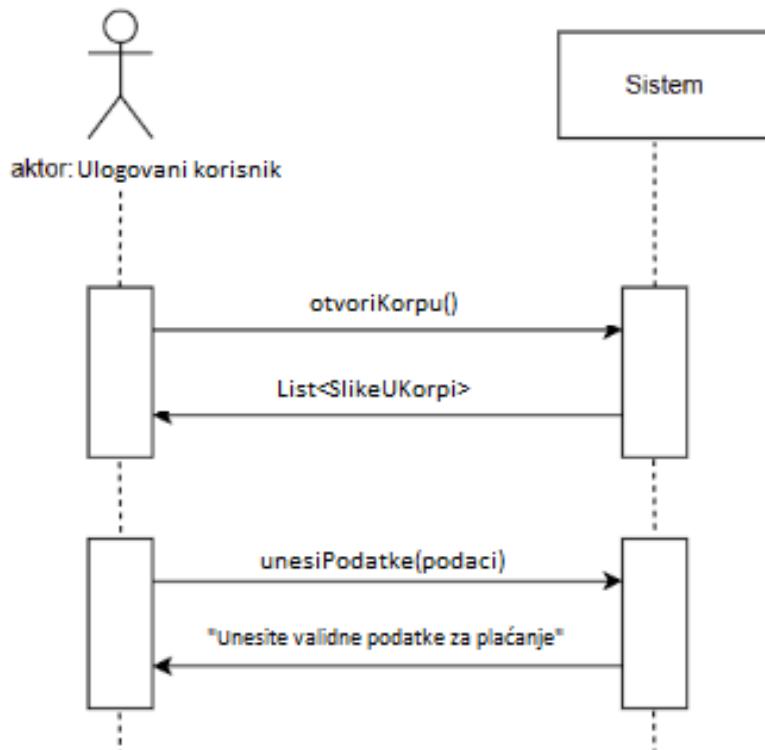


Alternativni scenariji:

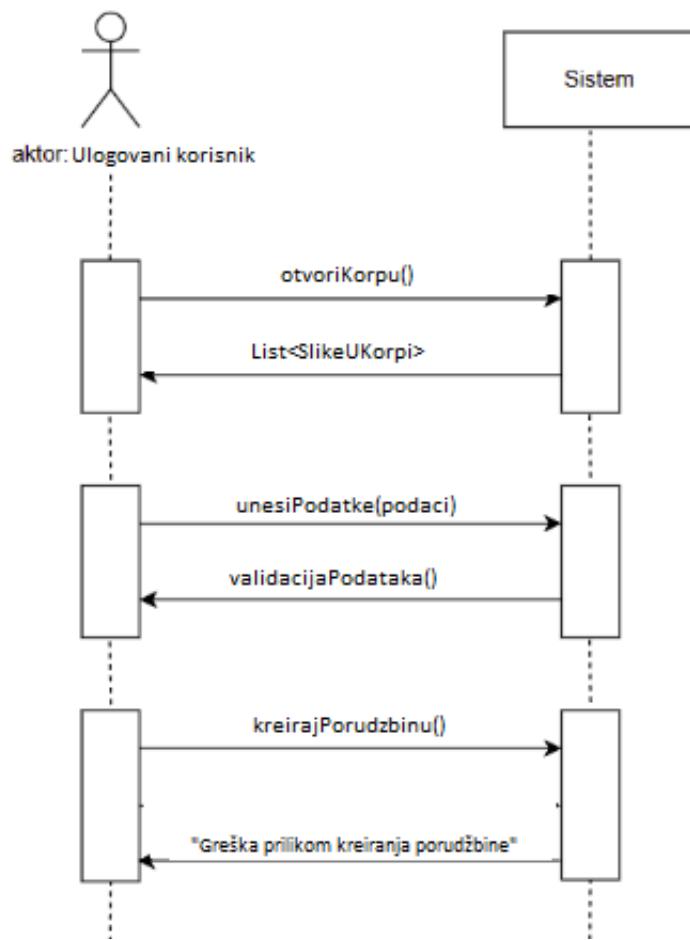
- 2.1 Ako je korpa prazna, sistem prikazuje poruku: „Korpa je prazna, dodajte slike pre porudžbine.“ (IA)



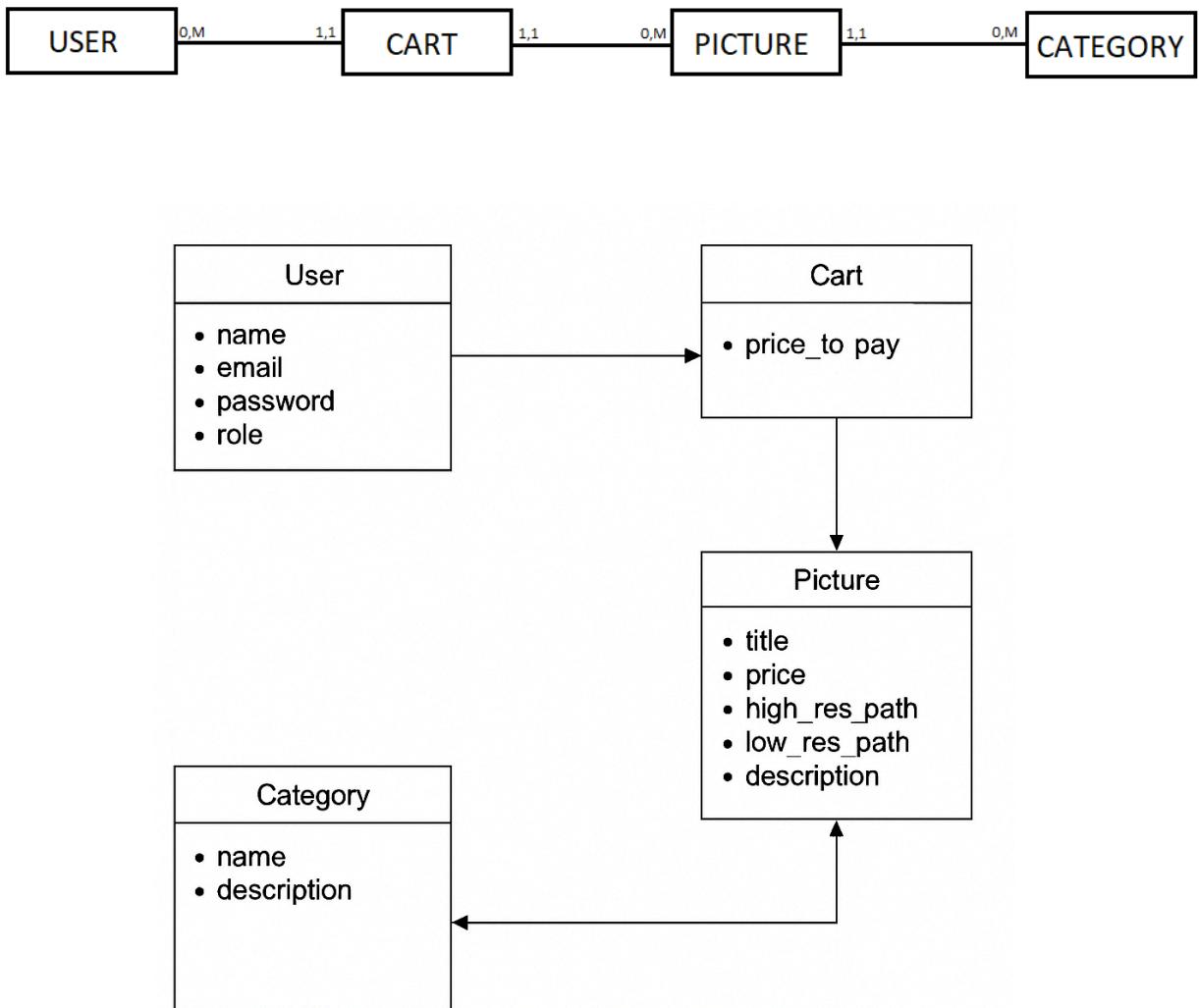
2.2 Ako neki od podataka za plaćanje nije unet ili nije validan, sistem prikazuje poruku: „Unesite validne podatke za plaćanje.“ (IA)



4.1 Ako dođe do greške prilikom kreiranja porudžbine, sistem prikazuje poruku: „Greška prilikom kreiranja porudžbine.“ (IA)



3.2.2. Struktura softverskog sistema



3.3. Faza projektovanja

Faza projektovanja predstavlja ključni korak u razvoju softverskog sistema, u kojem se definiše kako će sistem biti tehnički organizovan i implementiran. Tokom ove faze donose se odluke o arhitekturi sistema, izboru tehnologija, strukturi baze podataka, organizaciji koda, kao i o bezbednosnim mehanizmima. Cilj projektovanja je da obezbedi stabilnu osnovu za dalji razvoj aplikacije.

U slučaju digitalne prodavnice slika, sistem je zamišljen kao veb aplikacija zasnovana na klijent-server arhitekturi. Klijentska strana razvijena je korišćenjem React biblioteke i zadužena je za prikaz korisničkog interfejsa i interakciju sa korisnikom, dok je serverska strana implementirana u okviru Laravel PHP okvira i odgovorna je za obradu podataka, upravljanje poslovnom logikom i komunikaciju sa bazom podataka.

Aplikacija koristi MVC (Model-View-Controller) arhitektonski obrazac. Modeli predstavljaju osnovne entitete sistema kao što su korisnici, slike, kategorije i porudžbine. Kontroleri upravljaju logikom sistema i obradom korisničkih zahteva, dok su prikazi (views) realizovani pomoću React komponenti koje korisnicima omogućavaju pregled galerije slika, registraciju, upravljanje korpom, kao i pristup administratorskom panelu.

Struktura baze podataka projektovana je tako da podrži sve ključne funkcionalnosti sistema. Obuhvata tabele za korisnike, slike, kategorije i porudžbine. Sistemska uloga korisnika može biti kupac ili administrator, a pristup određenim delovima aplikacije kontroliše se na osnovu tih uloga.

Posebna pažnja posvećena je validaciji korisničkih unosa, zaštiti aplikacije i obradi grešaka. Primena autentikacije i autorizacije omogućava bezbedan pristup aplikaciji, dok middleware mehanizmi dodatno štite osetljive rute i funkcionalnosti sistema.

3.3.1. Projektovanje korisničkog interfejsa (ekstranskih formi)

Web aplikacija je osmišljena tako da podržava dve vrste korisnika: kupce i administratore. U zavisnosti od uloge korisnika nakon prijave, prikazuju se različite stranice i funkcionalnosti:

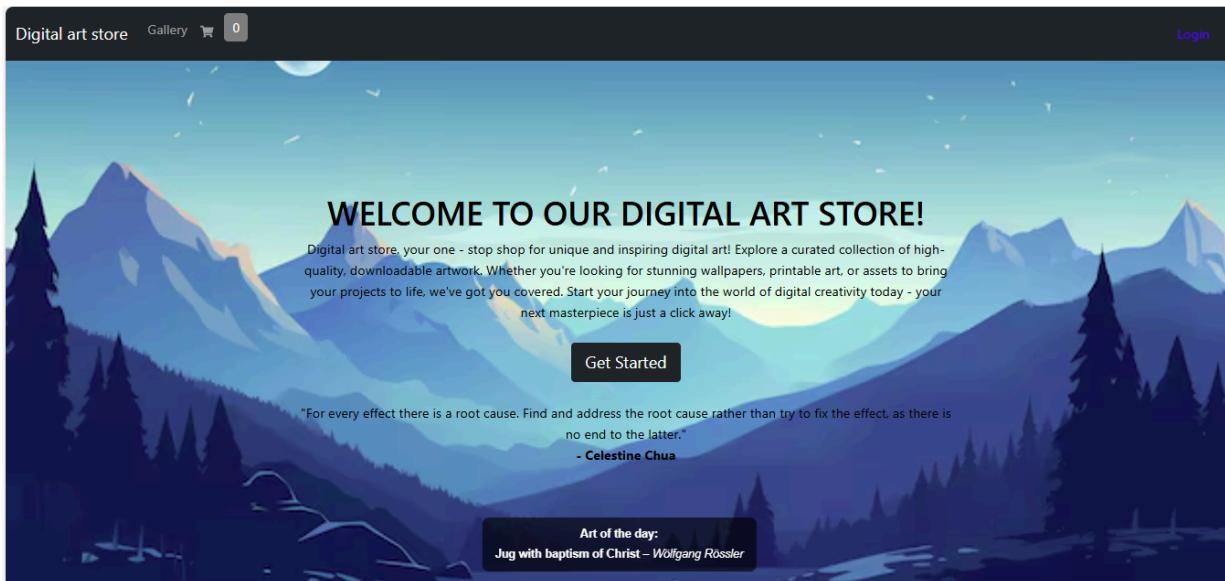
- **Korisnici** imaju pristup klasičnim stranicama kao što su početna stranica (*Home*), galerija (*Gallery*), korpa (*Cart*), prijava i registracija.
- **Administratori** imaju pristup posebnom admin panelu, koji omogućava upravljanje sadržajem sajta, uvid u poslovanje i korisnike.

U nastavku će biti opisane stranice aplikacije, podeljene u dve grupe:

- 1. Korisničke stranice**
- 2. Stranice admin panela**

Korisničke stranice:

Home page

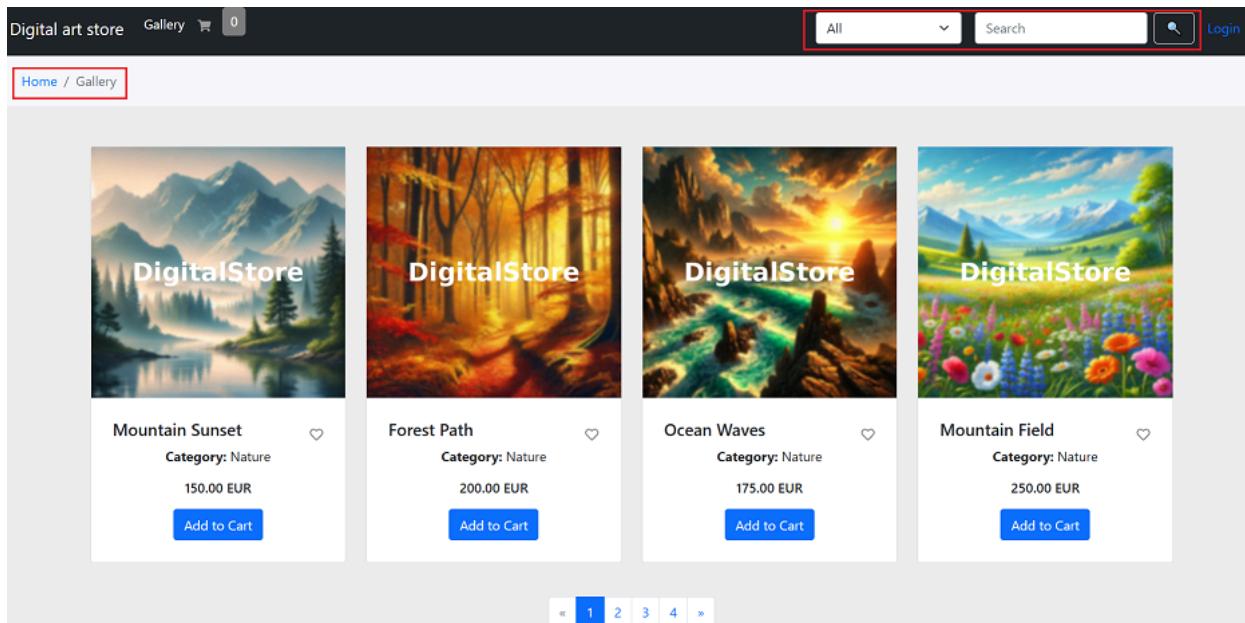


Početna stranica sadrži kratak opis prodavnice. Ispod poruke nalazi se dugme „*Get Started*“ koji vodi korisnika direktno na stranicu galerije. Pozadina stranice prikazuje pejzaž sa planinama i šumom, što doprinosi prijatnom vizuelnom utisku.

U gornjem navigacionom meniju nalaze se sledeći elementi: naziv sajta „*Digital art store*“ koji vodi na početnu stranicu, link ka galeriji, ikonica korpe za kupovinu sa prikazom broja artikala i dugme za prijavu (Login) u gornjem desnom uglu.

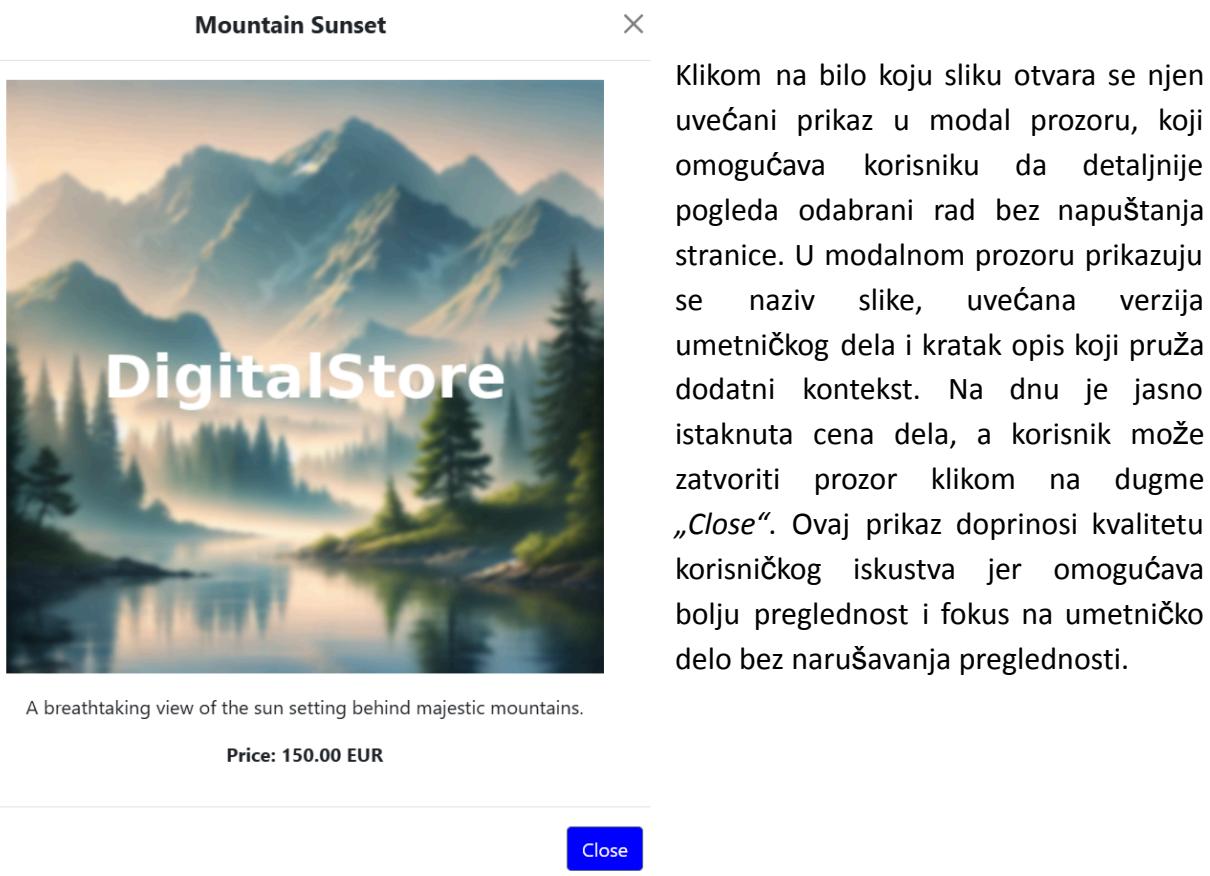
Ispod glavnog sadržaja stranice prikazuje se citat koji se menja prilikom svakog ponovnog učitavanja stranice, kao i umetničko delo dana.

Gallery page



Galerija predstavlja glavni deo aplikacije, gde korisnici mogu pregledati dostupna dela. Na vrhu stranice nalazi se breadcrumb navigacija („Home / Gallery“) koja jasno pokazuje korisniku gde se trenutno nalazi u okviru sajta. Glavni deo stranice sastoji se od kartica sa slikama, pri čemu svaka kartica sadrži vizuelni prikaz umetničkog dela, naziv slike, kategoriju, cenu izraženu u evrima, ikonicu za dodavanje u omiljene i dugme „Add to Cart“ koje omogućava ulogovanim korisnicima brzo dodavanje slika u korpu.

U gornjem desnom uglu je pretraživač sa padajućim menijem za izbor kategorije i poljem za unos teksta, što pomaže korisnicima da brzo pronađu slike po kategoriji ili nazivu. Na dnu galerije implementirana je paginacija koja korisniku omogućava da se kreće kroz više stranica sa umetničkim delima.



Cart page

Ulogovani korisnici imaju opciju dodavanja dela u korpu. Nakon prijave, u navigacionom meniju dobijaju dodatne opcije „Favorites“ i „Purchase History“. Na ovoj stranici korisnik može da vidi sve slike koje je odabrao za kupovinu. Ako korisnik odluči da ne želi određenu sliku, može da je ukloni iz korpe. Prikazana je ukupna cena svih odabralih dela, kao i forma za unos podataka potrebnih za plaćanje. Nakon unošenja podataka, klikom na dugme „Buy now“ pojavljuje se modalni prozor koji prikazuje unesene informacije kao potvrdu porudžbine. Korisnik može da otkaže kupovinu klikom na dugme „Cancel“ ili da potvrdi plaćanje klikom na dugme „Confirm“, čime se kupovina završava.

Digital art store [Gallery](#) [Favorites](#) [Purchase History](#) [2](#) [Logout](#)

[Home](#) / Cart

YOUR PICTURES

	Artistic Woman	X
	500.00 EUR	
	Peaceful River	X
	190.00 EUR	

Total: **690.00 EUR**

PAYMENT

1234 5678 9012 3457	Card Number
John Smith	Name on card
01/22	Expiration
***	CVV

[Buy now](#)

Confirm Data X

Card Number: 1234123412341234
Name on Card: Jana
Expiration: 03/27
CVV: 111

Lokacija localhost:3000 navodi

Purchase confirmed!

[Confirm](#) [Cancel](#)
[U redu](#)

Stranica „*Purchase History*“ je dostupna samo ulogovanim korisnicima i služi za pregled njihove istorije kupovina. Na njoj korisnik može videti listu svih prethodno kupljenih umetničkih dela sa osnovnim informacijama kao što su naziv slike, datum kupovine i cena.

Stranica pomaže korisniku da prati svoje kupovine, proveri detalje i po potrebi pronađe ranije kupljene radove. Takođe, može služiti kao potvrda uspešnih transakcija unutar aplikacije.

My Purchases

Dynamic Flow \$350.00 16. 5. 2025.	Geometric Chaos \$275.00 16. 5. 2025.	Dynamic Flow \$350.00 16. 5. 2025.

Login page

Na stranici za prijavu korisnik unosi svoje podatke, najčešće email ili korisničko ime i lozinku, kako bi pristupio svom nalogu. Ukoliko korisnik nema nalog, na istoj stranici nalazi se link „*Register here*“ koji ga vodi na stranicu za registraciju.

Log in

Email address

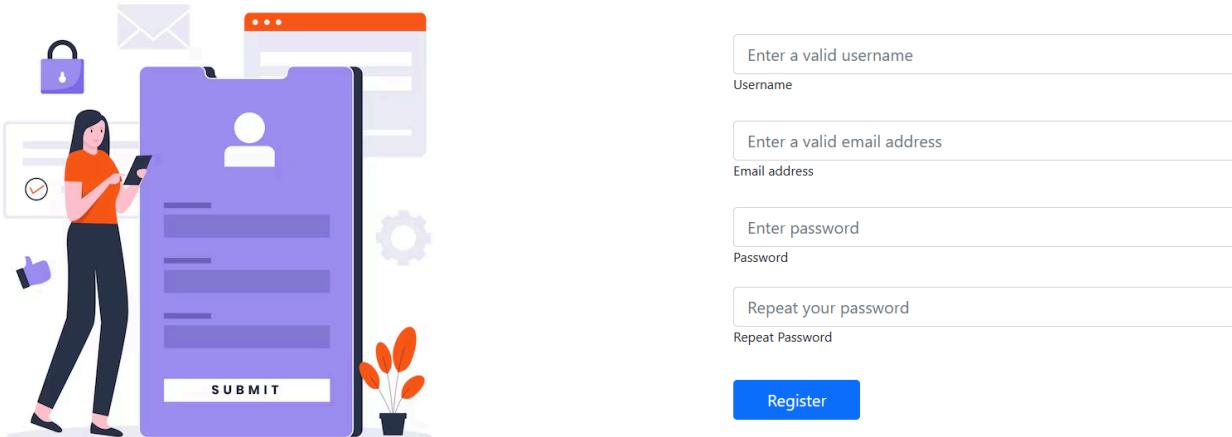
Password

Login

Don't have an account? [Register here](#)

Register page

Stranica za registraciju omogućava kreiranje novog naloga unosom osnovnih podataka kao što su ime, email, lozinka i ostali potrebni podaci. Nakon uspešne registracije, korisnik može da se prijavi i koristi sve funkcionalnosti sajta, uključujući kupovinu, dodavanje u omiljene i pregled istorije kupovina.



Stranice admin panela

Ukoliko se korisnik prijavi kao administrator, automatski mu se prikazuje admin panel umesto klasičnog korisničkog interfejsa.

Admin panel ima sopstveni navigacioni meni sa sledećim stavkama:

Admin Dashboard

početna stranica admin panela sa osnovnim pregledom aktivnosti



Add Pictures

Stranica **Add Picture** u okviru admin panela omogućava administratoru da doda novo umetničko delo u bazu podataka. Na stranici se nalazi forma u kojoj je potrebno uneti naziv slike, kratak opis, cenu izraženu u evrima, kao i izabrati odgovarajuću kategoriju iz padajućeg menija. Pored toga, administrator može da otpremi dve verzije slike — nisku rezoluciju koja će se koristiti za prikaz u galeriji i pregledima, i visoku rezoluciju koja je dostupna korisnicima nakon kupovine.

Add New Picture

Title

Description

Price (€)

Select Category ▾

Low Resolution Image:

Odabir datoteke Nije izabrana nijedna datoteka

High Resolution Image:

Odabir datoteke Nije izabrana nijedna datoteka

Submit Picture

Manage Pictures

Ova stranica sastoji se od redova, pri čemu svaki red predstavlja jednu sliku sa osnovnim informacijama kao što su naziv slike, kategorija i cena. Na kraju svakog reda nalaze se dva dugmeta- „Edit” i „Delete”. Klikom na dugme *Edit* otvara se forma koja prikazuje postojeće podatke o slici, omogućavajući administratoru da izvrši izmene po želji. Sa druge strane, klikom na dugme *Delete* pojavljuje se modalni prozor koji traži potvrdu da li je korisnik siguran da želi da obriše izabrano umetničko delo, čime se sprečavaju slučajna brisanja.

Manage Pictures

Title	Category	Price (EUR)	Actions	
Mountain Sunset	Nature	150.00	<button>Edit</button>	<button>Delete</button>
Forest Path	Nature	200.00	<button>Edit</button>	<button>Delete</button>
Ocean Waves	Nature	175.00	<button>Edit</button>	<button>Delete</button>
Mountain Field	Nature	250.00	<button>Edit</button>	<button>Delete</button>
Peaceful River	Nature	190.00	<button>Edit</button>	<button>Delete</button>
Color Burst	Abstract	300.00	<button>Edit</button>	<button>Delete</button>
Geometric Chaos	Abstract	275.00	<button>Edit</button>	<button>Delete</button>
Dynamic Flow	Abstract	350.00	<button>Edit</button>	<button>Delete</button>
Abstract Dreams	Abstract	400.00	<button>Edit</button>	<button>Delete</button>
Visionary Shapes	Abstract	320.00	<button>Edit</button>	<button>Delete</button>
Artistic Woman	Portraits	500.00	<button>Edit</button>	<button>Delete</button>
Boy	Portraits	450.00	<button>Edit</button>	<button>Delete</button>
Robot	Portraits	480.00	<button>Edit</button>	<button>Delete</button>
Old Soul	Portraits	550.00	<button>Edit</button>	<button>Delete</button>
Elegant Lady	Portraits	520.00	<button>Edit</button>	<button>Delete</button>

Edit Picture

A breathtaking view of the sun setting behind majestic mountains.

▼

[Save Changes](#)

Add Category

Na stranici se nalazi jednostavno polje za unos naziva kategorije, kao i opcionalno polje za kratak opis. Nakon popunjavanja, klikom na dugme „Create Category” nova kategorija se kreira i dodaje u sistem, čime postaje dostupna za izbor prilikom dodavanja ili uređivanja slika.

The screenshot shows a dark-themed admin panel header with various navigation links: Admin Panel, Dashboard, Add Picture, Manage Pictures, Add Category, Manage Categories, Preview Gallery, Analytics, and Show Users. The 'Add Category' link is highlighted with a red box. On the right side of the header is a 'Logout' button. Below the header, there is a large, light-colored input form titled 'Add New Category'. It contains two input fields: 'Category Name' and 'Description (optional)'. At the bottom of the form is a blue 'Create Category' button.

Manage Categories

Prikazuje listu svih postojećih kategorija umetničkih dela. Svaka kategorija je prikazana svojim nazivom, a pored se nalaze dva dugmeta: „Edit” i „Delete”. Klikom na dugme *Edit* administrator može da izmeni naziv kategorije, dok dugme *Delete* omogućava brisanje kategorije iz sistema.

The screenshot shows a dark-themed admin panel header with various navigation links: Admin Panel, Dashboard, Add Picture, Manage Pictures, Add Category, Manage Categories, Preview Gallery, Analytics, and Show Users. The 'Manage Categories' link is highlighted with a red box. On the right side of the header is a 'Logout' button. Below the header, there is a table titled 'Manage Categories'. The table has two columns: 'Category Name' and 'Actions'. The 'Category Name' column lists three categories: 'Nature', 'Abstract', and 'Portraits'. The 'Actions' column for each category contains two buttons: a blue 'Edit' button and a red 'Delete' button.

Category Name	Actions
Nature	Edit Delete
Abstract	Edit Delete
Portraits	Edit Delete

Manage Categories

Category Name	Actions	
Nature	 Save	 Cancel
Abstract	 Edit	 Delete
Portraits	 Edit	 Delete

Preview Gallery

Pruža administratoru mogućnost da vidi kako galerija izgleda iz korisničke perspektive. Ova stranica prikazuje umetnička dela onako kako će biti prikazana u korisničkom delu sajta, bez opcija za uređivanje ili upravljanje, što pomaže administratoru da proveri izgled i funkcionalnost galerije iz perspektive posetilaca.

Admin Panel Dashboard Add Picture Manage Pictures Add Category Manage Categories **Preview Gallery** Analytics Show Users Logout

Gallery Preview



DigitalStore

Mountain Sunset

Category: Nature

150.00 EUR

Add to Cart



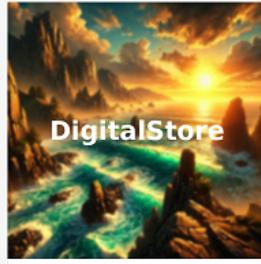
DigitalStore

Forest Path

Category: Nature

200.00 EUR

Add to Cart



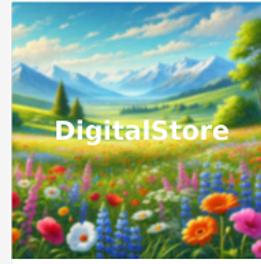
DigitalStore

Ocean Waves

Category: Nature

175.00 EUR

Add to Cart



DigitalStore

Mountain Field

Category: Nature

250.00 EUR

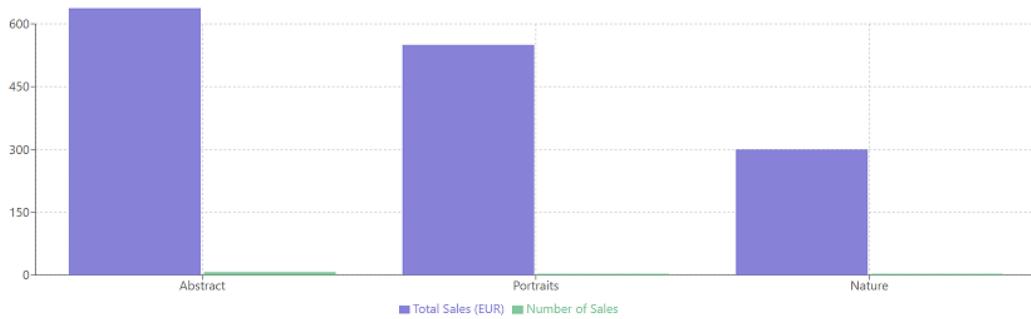
Add to Cart

Analytics

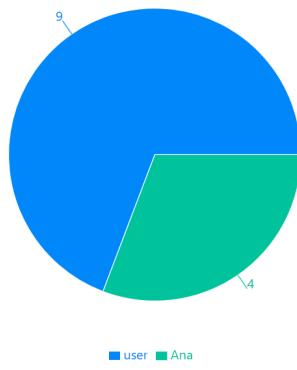
Ova stranica je odgovorna za pregled ključnih podataka o prodaji i korisnicima. Na stranici su prikazani dva glavna grafikona: pie chart koji prikazuje „*Top 5 Buyers by Purchase Count*“ — odnosno pet najaktivnijih kupaca prema broju kupovina, i stubičasti grafikon „*Sales by Category*“ koji prikazuje prodaju po različitim kategorijama umetničkih dela. Ovi grafikoni pomažu administratoru da brzo sagleda najvažnije informacije o poslovanju i korisničkoj aktivnosti.



Sales by Category



Top 5 Buyers by Purchase Count



Show Users

Predstavlja listu svih korisnika sistema. Na njoj se prikazuju osnovni podaci za svakog korisnika: jedinstveni ID, ime, email adresa, uloga (role) u sistemu, kao i datum kada su se pridružili (joined). Ova stranica omogućava administratorima pregled i upravljanje korisnicima na jednostavan i pregledan način.



User List

ID	Name	Email	Role	Joined
11	user	user@gmail.com	user	11. 5. 2025.
10	Maja	maja@gmail.com	admin	11. 5. 2025.
9	Ana	anaa@gmail.com	user	5. 1. 2025.
8	Jana	jana@gmail.com	user	5. 1. 2025.
7	Tamara	taca@gmail.com	guest	5. 1. 2025.
6	Ana	ana@gmail.com	guest	5. 1. 2025.
1	Prof. Tianna Zboncak	hgislason@example.net	user	5. 1. 2025.
2	Lionel Kub	bartoletti.aimee@example.com	user	5. 1. 2025.
3	Sadie Dach	gregory77@example.net	user	5. 1. 2025.
4	Tyreek Halvorson	piper.bogan@example.com	user	5. 1. 2025.
5	Mary Franecki	eloisa.bailey@example.net	user	5. 1. 2025.

3.3.2. Projektovanje aplikacione logike

3.3.2.1. Laravel Kontroleri (Controller)

U Laravel frameworku, **kontroleri** služe kao posrednici između korisničkih zahteva i modela baze podataka. Svaka poslovna logika koja je povezana sa HTTP zahtevima nalazi se upravo u kontrolerima. Olakšavaju organizaciju koda, povećavaju čitljivost i omogućavaju lakše testiranje. Tipično se nalaze u folderu *app/Http/Controllers*.

Kontroleri u Laravelu pružaju jasan i strukturiran način za upravljanje poslovnom logikom aplikacije. Svaki kontroler iz ovog projekta ima svoju specifičnu odgovornost i direktno

komunicira sa odgovarajućim modelima i bazom podataka, uz korišćenje validacija i odgovora u JSON formatu za API komunikaciju.

1. CartController – Upravljanje korporom i kupovinom

Glavne funkcionalnosti:

- Prikaz stavki u korpi (*index*)
- Dodavanje slike u korpu (*add*)
- Uklanjanje iz korpe (*remove*)
- Checkout i simulacija plaćanja (*checkout*)
- Preuzimanje slike visoke rezolucije (*downloadHighRes*)
- Prikaz istorije kupovine (*purchaseHistory*)

Primer koda:

```
public function add(Request $request)

{
    $validated = $request->validate([
        'picture_id' => 'required|exists:pictures,id',
    ]);

    $picture = Picture::find($validated['picture_id']);

    if (!$picture) {
        return response()->json(['error' => 'Picture not found'], 404);
    }

    $cart = Cart::create([
        'user_id' => auth()->id(),
        'picture_id' => $validated['picture_id'],
        'price_to_pay' => $picture->price,
    ]);

    $picture->increment('downloads');

    return response()->json(['message' => 'Item bought', 'cart' => $cart], 201);
}
```

2. CategoryController – Upravljanje kategorijama

Glavne funkcionalnosti:

- Prikaz svih kategorija (*index*)
- Prikaz pojedinačne kategorije (*show*)
- Dodavanje nove kategorije (*store*)
- Izmena postojeće kategorije (*update*)
- Brisanje kategorije (*destroy*)

Validacija se koristi za čuvanje ispravnih podataka:

```
$validated = $request->validate([
    'name' => 'required|string|max:255|unique:categories',
    'description' => 'nullable|string'
]);
```

3. PictureController – Upravljanje slikama

Glavne funkcionalnosti:

- Prikaz svih slika (*index*)
- Kreiranje nove slike (*store*)
- Ažuriranje slike (*update*)
- Brisanje slike i fajlova sa diska (*destroy*)
- Pretraga slika po nazivu, kategoriji i ceni (*searchBy...*)
- Preuzimanje low/high-res slike (*showLowRes, showHighRes*)
- Paginacija (*paginate_pictures*)
- Upload fajlova (*upload*)

Primer validacije i čuvanja fajla:

```
$validated = $request->validate([
    'high_res_file' => 'required|file|mimes:jpeg,png,jpg|max:5120',
]);
```

```
'low_res_file' => 'required|file|mimes:jpeg,png,jpg|max:2048',  
]);
```

4. SalesController – Analitika prodaje

Glavne funkcionalnosti:

- Prikaz najprodavanijih kategorija (salesByCategory)
- Prikaz top kupaca (topBuyers)

Koristi Čisti SQL i agregatne funkcije:

```
public function salesByCategory()  
  
{    $categories = DB::select("SELECT  
  
        cat.name AS category_name,  
  
        SUM(cart.price_to_pay) AS total_sales,  
  
        COUNT(cart.id) AS total_sales_count,  
  
        AVG(cart.price_to_pay) AS average_sale  
  
    FROM  
  
        carts cart  
  
    JOIN  
  
        pictures pic ON cart.picture_id = pic.id  
  
    JOIN  
  
        categories cat ON pic.category_id = cat.id  
  
    GROUP BY  
  
        cat.id, cat.name  
  
    ORDER BY  
  
        total_sales DESC;  
") ;
```

5. UserController – Upravljanje korisnicima

Glavne funkcionalnosti:

- Prikaz svih korisnika (*index*)
- Prikaz pojedinačnog korisnika (*show*)
- Brisanje korisnika (*destroy*)

Dodatna logika:

Metoda *destroy()* proverava da li korisnik postoji pre nego što ga obriše.

```
public function destroy(User $user)

{
    $user = User::find($id);

    if (!$user) {
        return response()->json(['message' => 'User not found'], 404);
    }

    $user->delete();

    return response()->json(['message' => 'User deleted successfully'], 200);
}
```

3.3.2.2. Laravel Autentifikacija (Authentication)

Laravel pruža ugrađen sistem za autentifikaciju korisnika pomoću *Sanctum* paketa za token-baziranu autentifikaciju u API aplikacijama. U ovom projektu omogućeno je:

- Registracija korisnika
- Login korisnika
- Logout (brisanje tokena)
- Automatsko kreiranje pristupnog tokena

1. Registracija korisnika

Metod: *register(Request \$request)*

Putanja: *POST /api/register*

Logika:

1. Validira se unos korisnika: ime, email, lozinka, uloga.
2. Lozinka mora imati:
 - najmanje 8 karaktera
 - bar jedno veliko slovo
 - bar jedno malo slovo
 - bar jedan broj
3. Ako validacija prođe, korisnik se upisuje u bazu.
4. Kreira se pristupni token (*auth_token*) koji se vraća korisniku.

```
public function register(Request $request) {  
  
    $validator=Validator::make($request->all(), [  
  
        'role'=>'in:admin,user',  
  
        'name'=>'required|string|max:255',  
  
        'email'=>[  
  
            'required',  
  
            'string',  
  
            'max:255',  
  
            'regex:/^[@][^@]+@[^@]+\.[a-zA-Z]{2,}\$/'  
    ]);  
  
    if ($validator->fails()) {  
        return response()->json(['error'=>$validator->errors()]);  
    } else {  
        $user = User::create($request->only(['name', 'email', 'password', 'role']));  
  
        $token = $user->createToken('auth_token');  
  
        return response()->json(['token'=>$token->accessToken]);  
    }  
}
```

```

        'unique:users',
    ],
    'password'=>[
        'required',
        'string',
        'min:8',
        'regex:/[A-Z]/',
        'regex:/[a-z]/',
        'regex:/[0-9]/',
    ],
]) ;

if($validator->fails()) {
    return response()->json([
        'errors' => $validator->errors()
    ], 422);
}

$user=User::create([
    'name'=>$request->name,
    'email'=>$request->email,
    'password'=>Hash::make($request->password),
    'role' => $request->role ?? User::ROLE_USER,
    'email_verified_at' => now()
]);
$token=$user->createToken('auth_token')->plainTextToken;

return response()->json(['data'=>$user, 'access_token'=>$token,
'token_type'=>'Bearer']);
}

```

2. Login korisnika

Metod: *login(Request \$request)*

Putanja: *POST /api/login*

Logika:

- Proverava se da li korisnik postoji i da li su podaci ispravni.
- Ako je uspešno, generiše se novi auth_token.

```
public function login(Request $request) {  
  
    if (!Auth::attempt($request->only('email', 'password'))) {  
  
        return response()->json(['success' => false], 401);  
  
    }  
  
  
    $user = User::where('email', $request['email'])->firstOrFail();  
  
  
    $token = $user->createToken('auth_token')->plainTextToken;  
  
    return response()->json([  
  
        'success' => true,  
  
        'access_token' => $token,  
  
        'token_type' => 'Bearer',  
  
        'user' => [  
  
            'id' => $user->id,  
  
            'name' => $user->name,  
  
            'email' => $user->email,  
  
            'role' => $user->role,  
  
        ]  
    ]);  
}
```

3. Logout korisnika

Metod: `logout(Request $request)`

Putanja: `POST /api/logout (uz token u headeru)`

Logika:

- Brišu se svi tokeni korisnika koji je trenutno ulogovan.

```
public function logout(Request $request)
{
    $user = $request->user();

    if ($user) {
        $user->tokens()->delete();

        return response()->json(['message' => 'Successfully logged out.']);
    }

    return response()->json(['message' => 'Not authenticated.'], 401);
}
```

4. Autorizacija po korisničkoj ulozi

Pored autentifikacije pomoću Laravel Sanctum-a, u ovoj aplikaciji je implementiran i dodatni sloj autorizacije koji omogućava da se pristup određenim rutama ograniči samo na korisnike sa tačno definisanom ulogom (npr. user ili admin).

Za to se koristi custom middleware koji se naziva `RoleMiddleware`.

Lokacija fajla: `app/Http/Middleware/RoleMiddleware.php`

Logika rada:

Middleware proverava da li je korisnik ulogovan (**auth()->user()**) i da li njegova uloga odgovara očekivanoj ulozi prosleđenoj kao argument. Ako jeste, dopušta dalji prolazak do rute. Ako nije, vraća HTTP odgovor **403 Unauthorized**.

```
public function handle(Request $request, Closure $next, $role)

{
    if (auth()->user() && auth()->user()->role === $role) {
        return $next($request);
    }

    return response()->json(['error' => 'Unauthorized'], 403);
}
```

Registracija middleware-a:

Middleware je registrovan u fajlu [app/Http/Kernel.php](#), unutar niza **\$routeMiddleware**:

```
'role'=>\App\Http\Middleware\RoleMiddleware::class
```

Ovo omogućava da se middleware koristi na rutama pomoću naziva **role**.

Primer upotrebe u rutama:

```
Route::group(['middleware' => ['auth:sanctum','role:user']], function () {
    Route::get('/cart/{id}/download', [CartController::class, 'downloadHighRes']);
});
```

U ovom primeru, pristup ruti `/cart/{id}/download` biće dozvoljen samo korisnicima sa ulogom `user` koji su prethodno autentifikovani putem Sanctum tokena.

3.3.2.3. Laravel Resursi (Resources)

U Laravelu, **API Resources** (resursi) služe za transformaciju modela i kolekcija modela u *JSON* format koji se šalje korisnicima. Resursi omogućavaju:

- Kontrolu nad time **Šta se vraća korisniku**
- Formatiranje odgovora u **standardizovani i čitljiv oblik**
- Jednostavno odvajanje **logike baze od strukture odgovora**

Resursi se nalaze u direktorijumu: *app/Http/Resources/*

Primer:

Ova klasa predstavlja resurs koji definiše kako se objekat tipa *Picture* transformiše u *JSON* format prilikom odgovora API-ja. Koristi **\$wrap = 'picture'** da bi u odgovoru JSON bio obavljen imenom objekta. Prikazuje osnovne atrribute slike i relaciju ka kategoriji.

```
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

use App\Models\Picture;

use Illuminate\Http\Request;

class PictureResource extends JsonResource

{
    /**
     * Transform the resource into an array.
     *
     * @param \Illuminate\Http\Request $request
     * @return array|\Illuminate\Contracts\Support\Arrayable|\JsonSerializable
     */
    public static $wrap = 'picture';

    public function toArray($request)
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'category_id' => $this->category_id,
            'category' => $this->category->name,
            'created_at' => $this->created_at->format('Y-m-d H:i:s'),
            'updated_at' => $this->updated_at->format('Y-m-d H:i:s')
        ];
    }
}
```

```
        return[  
  
            'id'=>$this->id,  
  
            'title'=>$this->title,  
  
            'description'=>$this->description,  
  
            'low_res_path'=>$this->low_res_path,  
  
            'high_res_path'=>$this->high_res_path,  
  
            'price'=>$this->price,  
  
            'category'=>$this->category  
  
        ];  
  
    }  
  
}
```

3.3.2.4. Laravel Rutiranje (Api routes)

Rutiranje (*Routing*) u Laravelu predstavlja mehanizam pomoću kojeg se HTTP zahtevi (kao što su *GET*, *POST*, *PUT*, *DELETE*) povezuju sa odgovarajućim funkcijama (metodama) u kontrolerima aplikacije. Drugim rečima, rutiranje određuje šta će se desiti kada korisnik pošalje zahtev na određenu adresu.

U okviru *routes/api.php* fajla definišu se sve rute koje se odnose na **API deo aplikacije**. Ove rute najčešće vraćaju *JSON* odgovore i koriste se kada frontend (npr. React aplikacija) komunicira sa Laravel backend-om.

U nastavku su prikazane sve API rute korišćene u ovom projektu, zajedno sa opisom funkcionalnosti, HTTP metodama, URL adresama, očekivanim parametrima i formatima podataka.

Opis funkcije	Dohvatanje nasumičnog citata sa ZenQuotes API-ja
HTTP metoda	GET
URL	/api/random-quote
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Ruta koristi eksterni servis (ZenQuotes API) kako bi dohvatile inspirativan citat i autora. Ne zahteva autentifikaciju.
Output	{ "quote": "Do all things with love.", "author": "Og Mandino" }

Opis funkcije	Dohvatanje umetničkog dela dana
HTTP metoda	GET
URL	/api/art-of-the-day
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Ruta koristi eksterni servis (Metropolitan Museum Collection API) kako bi prikazala umetničko delo i njegovog autora. Ne zahteva autentifikaciju.
Output	{ "title": "View of the Colosseum and the Arch of Constantine from the Palatine", "artist": "Charles Rémond" }

Opis funkcije	Registracija novog korisnika
HTTP metoda	POST
URL	/api/register
HTTP body parametri	{ "name": "Petar Petrović", "email": "pera@example.com", "password": "Lozinka123", "role": "user" }
Format HTTP body parametara	JSON

Napomena	Polje "role" je opcionalno – ukoliko nije prosleđeno, podrazumevano se dodeljuje vrednost "user". Lozinka mora sadržati bar jedno veliko slovo, jedno malo slovo i jedan broj i imati najmanje 8 karaktera. Sistem vraća token za autentifikaciju.
Output	{ "data": { "name": "Petar Petrović", "email": "pera@example.com", "role": "user", "email_verified_at": "2025-06-20T09:12:10.000000Z", "updated_at": "2025-06-20T09:12:10.000000Z", "created_at": "2025-06-20T09:12:10.000000Z", "id": 13 }, "access_token": "59 rD66bNc93lKP2ZacNcmSQPKDo60tWjda4Kkck839413098a", "token_type": "Bearer" }

Opis funkcije	Prijava (logovanje) korisnika
HTTP metoda	POST
URL	/api/login
HTTP body parametri	{ "email": "petar@example.com", "password": "Lozinka123" }
Format HTTP body parametara	JSON
Napomena	Ukoliko su kredencijali ispravni, vraća se access_token koji je neophodno sačuvati i koristiti za dalji pristup zaštićenim rutama (Authorization: Bearer token).
Output	{ "success": true, "access_token": "60 I1lL77AX453QG8Cv97JnhwgDMynf6l4sbiZYy1rA01533e65", "token_type": "Bearer", "user": { "id": 12, "name": "Petar Petrović", "email": "petar@example.com", "role": "user" } }

Opis funkcije	Prikaz svih slika iz baze
HTTP metoda	GET
URL	/api/pictures
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Ova ruta vraća sve slike koje se nalaze u bazi, zajedno sa podacima o kategoriji, ceni, i putanjama bilo da je reč o slikama niske ili visoke rezolucije. Koristi se za prikaz svih radova u galeriji korisnicima.
Output	<pre>{ "pictures": [{ "id": 1, "title": "Mountain Sunset", "description": "A breathtaking view of the sun setting behind majestic mountains.", "category_id": 1, "category_name": "Nature", "price": "150.00", "low_res_path": "http://127.0.0.1:8000/images/low_res/devetal.png", "high_res_path": "http://127.0.0.1:8000/images/high_res/deveta.png" }, { "id": 2, "title": "Forest Path", "description": "A peaceful trail winding through a lush, green forest.", "category_id": 1, "category_name": "Nature", "price": "200.00", "low_res_path": "http://127.0.0.1:8000/images/low_res/osmal.png", "high_res_path": "http://127.0.0.1:8000/images/high_res/osma.png" }, { "id": 3, "title": "Ocean Waves", "description": "Gentle waves crashing onto a serene beach at sunset.", "category_id": 1, "category_name": "Nature", "price": "175.00", "low_res_path": "http://127.0.0.1:8000/images/low_res/osmal.png", "high_res_path": "http://127.0.0.1:8000/images/high_res/osma.png" }] }</pre>

```

        "low_res_path": "http://127.0.0.1:8000/images/low_res/sedmal.png",
        "high_res_path": "http://127.0.0.1:8000/images/high_res/sedma.png"
    },
    .....
    {
        "id": 14,
        "title": "Old Soul",
        "description": "A moving portrait of wisdom and experience captured in an elder's gaze.",
        "category_id": 3,
        "category_name": "Portraits",
        "price": "550.00",
        "low_res_path": "http://127.0.0.1:8000/images/low_res/jedanaestal.png",
        "high_res_path": "http://127.0.0.1:8000/images/high_res/jedanaesta.png"
    },
    {
        "id": 15,
        "title": "Elegant Lady",
        "description": "A refined and graceful portrayal of an elegant woman.",
        "category_id": 3,
        "category_name": "Portraits",
        "price": "520.00",
        "low_res_path": "http://127.0.0.1:8000/images/low_res/dvanaestal.png",
        "high_res_path": "http://127.0.0.1:8000/images/high_res/dvanaesta.png"
    }
]
}

```

Opis funkcije	Pretraga slika po naslovu
HTTP metoda	GET
URL	/api/pictures/search/{title}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća sve slike koje u nazivu sadrže reč {title}
Output	title : Mountain [{

```

    "id": 1,
    "title": "Mountain Sunset",
    "description": "A breathtaking view of the sun setting behind majestic mountains.",
    "low_res_path": "images/low_res/devetal.png",
    "high_res_path": "images/high_res/deveta.png",
    "price": "150.00",
    "downloads": 34,
    "created_at": "2025-01-05T08:59:53.000000Z",
    "updated_at": "2025-05-11T12:05:17.000000Z",
    "category_id": 1
},
{
    "id": 4,
    "title": "Mountain Field",
    "description": "A vibrant field nestled between towering mountain peaks.",
    "low_res_path": "images/low_res/desetal.png",
    "high_res_path": "images/high_res/deseta.png",
    "price": "250.00",
    "downloads": 32,
    "created_at": "2025-01-05T08:59:53.000000Z",
    "updated_at": "2025-01-05T08:59:53.000000Z",
    "category_id": 1
}
]
ako nijedna slika ne sadrži prosleđenu reč:
{
    "error": "No picture found"
}

```

Opis funkcije	Pretraga slika koje su jeftinije od zadate cene
HTTP metoda	GET
URL	/api/pictures/under/{price}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća sve slike sa cenom manjom ili jednakom od {price}
Output	price: 200 [

```
{
  "id": 1,
  "title": "Mountain Sunset",
  "description": "A breathtaking view of the sun setting behind majestic mountains.",
  "low_res_path": "images/low_res/devetal.png",
  "high_res_path": "images/high_res/deveta.png",
  "price": "150.00",
  "downloads": 34,
  "created_at": "2025-01-05T08:59:53.000000Z",
  "updated_at": "2025-05-11T12:05:17.000000Z",
  "category_id": 1
},
{
  "id": 2,
  "title": "Forest Path",
  "description": "A peaceful trail winding through a lush, green forest.",
  "low_res_path": "images/low_res/osmal.png",
  "high_res_path": "images/high_res/osma.png",
  "price": "200.00",
  "downloads": 3,
  "created_at": "2025-01-05T08:59:53.000000Z",
  "updated_at": "2025-01-05T08:59:53.000000Z",
  "category_id": 1
},
{
  "id": 3,
  "title": "Ocean Waves",
  "description": "Gentle waves crashing onto a serene beach at sunset.",
  "low_res_path": "images/low_res/sedmal.png",
  "high_res_path": "images/high_res/sedma.png",
  "price": "175.00",
  "downloads": 22,
  "created_at": "2025-01-05T08:59:53.000000Z",
  "updated_at": "2025-01-05T08:59:53.000000Z",
  "category_id": 1
},
{
  "id": 5,
  "title": "Peaceful River",

```

	<pre> "description": "A tranquil river flowing through a picturesque valley.", "low_res_path": "images/low_res/sestal.png", "high_res_path": "images/high_res/sesta.png", "price": "190.00", "downloads": 6, "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z", "category_id": 1 }] </pre> <p>ako ne postoje slike koje imaju jednaku ili nižu cenu od prosleđene:</p> <pre> { "error": "No pictures found under this price" } </pre>
--	---

Opis funkcije	Pristup slici niske rezolucije
HTTP metoda	GET
URL	/api/pictures/{id}/low-res
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Korisno za prikaz preview verzije slike na frontend-u.
Output	<pre> { "low_res_path": "images/low_res/devetal.png" } </pre> <p>ako id ne odgovara nijednoj slici:</p> <pre> { "message": "Picture not found" } </pre>

Opis funkcije	Prikaz svih slika koje pripadaju određenoj kategoriji
HTTP metoda	GET
URL	/api/categories/{id}/pictures
HTTP body parametri	nema
Format HTTP body parametara	N/A

Napomena	Vraća sve slike koje pripadaju kategoriji sa odgovarajućim ID-jem
Output	category id: 3 [{ "id": 11, "title": "Artistic Woman", "description": "A striking portrait of a woman with an artistic flair.", "low_res_path": "images/low_res/petnaestal.png", "high_res_path": "images/high_res/petnaesta.png", "price": "500.00", "downloads": 2, "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z", "category_id": 3 }, { "id": 12, "title": "Boy", "description": "A charming portrait capturing the innocence of youth.", "low_res_path": "images/low_res/trinaestal.png", "high_res_path": "images/high_res/trinaesta.png", "price": "450.00", "downloads": 27, "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z", "category_id": 3 }, { "id": 13, "title": "Robot", "description": "A creative depiction of a futuristic humanoid robot.", "low_res_path": "images/low_res/cetrnaestal.png", "high_res_path": "images/high_res/cetrnaest.png", "price": "480.00", "downloads": 35, "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z", "category_id": 3 },]

```
{
  "id": 14,
  "title": "Old Soul",
  "description": "A moving portrait of wisdom and experience captured in an elder's gaze.",
  "low_res_path": "images/low_res/jedanaestal.png",
  "high_res_path": "images/high_res/jedanaesta.png",
  "price": "550.00",
  "downloads": 39,
  "created_at": "2025-01-05T08:59:53.000000Z",
  "updated_at": "2025-01-05T08:59:53.000000Z",
  "category_id": 3
},
{
  "id": 15,
  "title": "Elegant Lady",
  "description": "A refined and graceful portrayal of an elegant woman.",
  "low_res_path": "images/low_res/dvanaestal.png",
  "high_res_path": "images/high_res/dvanaesta.png",
  "price": "520.00",
  "downloads": 29,
  "created_at": "2025-01-05T08:59:53.000000Z",
  "updated_at": "2025-01-05T08:59:53.000000Z",
  "category_id": 3
}
]
za nepostojeći id:
{
  "error": "No pictures found in this category"
}
```

Opis funkcije	Prikaz svih kategorija
HTTP metoda	GET
URL	/api/categories
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Ruta se koristi za prikazivanje liste svih dostupnih kategorija u sistemu
Output	[{ "id": 1, "name": "Nature", "description": "Experience the beauty of the natural world through stunning depictions of landscapes, wildlife, forests, mountains, rivers, and serene scenes that capture the essence of our environment. Perfect for nature lovers and those seeking a sense of tranquility.", "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z" }, { "id": 2, "name": "Abstract", "description": "Immerse yourself in a world of creativity and imagination with abstract art. Featuring bold colors, dynamic shapes, and intricate patterns, these pieces evoke emotions and inspire thought, making them perfect for modern spaces and art enthusiasts.", "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z" }, { "id": 3, "name": "Portraits", "description": "Discover the art of human expression through captivating portraits. From realistic depictions to imaginative interpretations, these artworks capture the emotions, personalities, and stories of individuals, bringing them to life with every detail.", "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z" }]

Opis funkcije	Prikaz jedne kategorije po ID-ju
HTTP metoda	GET
URL	/api/categories/{id}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća informacije o jednoj kategoriji.
Output	<pre> id: 1 { "id": 1, "name": "Nature", "description": "Experience the beauty of the natural world through stunning depictions of landscapes, wildlife, forests, mountains, rivers, and serene scenes that capture the essence of our environment. Perfect for nature lovers and those seeking a sense of tranquility.", "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-01-05T08:59:53.000000Z" } za nepostojeći category id: { "message": "Category not found" } </pre>

Opis funkcije	Odjava korisnika i poništavanje tokena
HTTP metoda	POST
URL	/api/logout
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Korisnik mora biti prijavljen. Nakon poziva ove rute, svi aktivni tokeni za korisnika se brišu i mora se ponovo ulogovati da bi pristupio zaštićenim rutama.
Output	<pre>{ "message": "Successfully logged out." }</pre>

Opis funkcije	Prikazivanje informacija o trenutno prijavljenom korisniku
HTTP metoda	GET
URL	/api/profile
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća ime, email i ulogu trenutno prijavljenog korisnika.
Output	<pre>{ "id": 11, "name": "user", "email": "user@gmail.com", "email_verified_at": "2025-05-11T08:19:20.000000Z", "created_at": "2025-05-11T08:19:20.000000Z", "updated_at": "2025-05-11T08:19:20.000000Z", "role": "user" }</pre>

Opis funkcije	Prikaz istorije svih prethodnih kupovina korisnika
HTTP metoda	GET
URL	/api/purchase-history
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraca sve slike koje je korisnik kupio, sa podacima o slici i datumu kupovine.
Output	<pre>[{ "id": 16, "price_to_pay": "350.00", "created_at": "2025-05-16T19:23:12.000000Z", "updated_at": "2025-05-16T19:23:12.000000Z", "user_id": 11, "picture_id": 8, "picture": { "id": 8, "title": "Dynamic Flow", "description": "Fluid abstract waves that evoke movement and energy." } }]</pre>

```

    "low_res_path": "images/low_res/cetvrtal.png",
    "high_res_path": "images/high_res/cetvrta.png",
    "price": "350.00",
    "downloads": 8,
    "created_at": "2025-01-05T08:59:53.000000Z",
    "updated_at": "2025-05-16T19:23:12.000000Z",
    "category_id": 2
  }
},
{
  "id": 15,
  "price_to_pay": "275.00",
  "created_at": "2025-05-16T19:23:04.000000Z",
  "updated_at": "2025-05-16T19:23:04.000000Z",
  "user_id": 11,
  "picture_id": 7,
  "picture": {
    "id": 7,
    "title": "Geometric Chaos",
    "description": "An intricate blend of geometric patterns in vivid hues.",
    "low_res_path": "images/low_res/petal.png",
    "high_res_path": "images/high_res/peta.png",
    "price": "275.00",
    "downloads": 18,
    "created_at": "2025-01-05T08:59:53.000000Z",
    "updated_at": "2025-05-16T19:23:04.000000Z",
    "category_id": 2
  }
},
{
  "id": 14,
  "price_to_pay": "350.00",
  "created_at": "2025-05-16T19:22:58.000000Z",
  "updated_at": "2025-05-16T19:22:58.000000Z",
  "user_id": 11,
  "picture_id": 8,
  "picture": {
    "id": 8,
    "title": "Dynamic Flow",

```

	<pre> "description": "Fluid abstract waves that evoke movement and energy.", "low_res_path": "images/low_res/cetvrtal.png", "high_res_path": "images/high_res/cetvrta.png", "price": "350.00", "downloads": 8, "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-05-16T19:23:12.000000Z", "category_id": 2 } },] </pre>
--	---

Opis funkcije	Prikazivanje stavki koje se nalaze u korisničkoj korpi
HTTP metoda	GET
URL	/api/cart
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Korisnik mora biti ulogovan. Vraća listu svih slika koje se nalaze u korpi.
Output	<pre>[{ "id": 8, "price_to_pay": "150.00", "created_at": "2025-05-11T09:22:06.000000Z", "updated_at": "2025-05-11T09:22:06.000000Z", "user_id": 11, "picture_id": 1, "picture": { "id": 1, "title": "Mountain Sunset", "description": "A breathtaking view of the sun setting behind majestic mountains.", "low_res_path": "images/low_res/devetal.png", "high_res_path": "images/high_res/deveta.png", "price": "150.00", "downloads": 34, "created_at": "2025-01-05T08:59:53.000000Z", "updated_at": "2025-05-16T19:23:12.000000Z" } }]</pre>

```

        "updated_at": "2025-05-11T12:05:17.000000Z",
        "category_id": 1
    }
},
{
    "id": 16,
    "price_to_pay": "350.00",
    "created_at": "2025-05-16T19:23:12.000000Z",
    "updated_at": "2025-05-16T19:23:12.000000Z",
    "user_id": 11,
    "picture_id": 8,
    "picture": {
        "id": 8,
        "title": "Dynamic Flow",
        "description": "Fluid abstract waves that evoke movement and energy.",
        "low_res_path": "images/low_res/cetvrtal.png",
        "high_res_path": "images/high_res/cetvrta.png",
        "price": "350.00",
        "downloads": 8,
        "created_at": "2025-01-05T08:59:53.000000Z",
        "updated_at": "2025-05-16T19:23:12.000000Z",
        "category_id": 2
    }
}
]

ako još uvek nema ništa u korpi:
{
    "message": "Cart is empty"
}

```

Opis funkcije	Dodavanje slike u korpu (kupovina)
HTTP metoda	POST
URL	/api/cart
HTTP body parametri	{ "picture_id": 5 }
Format HTTP body parametara	JSON
Napomena	Korisnik mora biti ulogovan. Nakon kupovine, slika se dodaje u korisničku korpu i automatski se povećava broj preuzimanja za tu sliku.
Output	{ "message": "Item bought", "cart": { "user_id": 8, "picture_id": "5", "price_to_pay": "190.00", "updated_at": "2025-06-24T08:33:01.000000Z", "created_at": "2025-06-24T08:33:01.000000Z", "id": 75 } }

Opis funkcije	Brisanje stavke iz korpe
HTTP metoda	DELETE
URL	/api/cart/{id}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Korisnik može izbrisati samo stavke koje pripadaju njegovoj korpi.
Output	{ "message": "Item removed from cart" }

Opis funkcije	Završetak kupovine – plaćanje i brisanje korpe
HTTP metoda	POST
URL	/api/cart/checkout
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Simulira uspešno plaćanje i briše sve stavke iz korpe. Vraća ukupnu cenu kupovine.
Output	{ "message": "Checkout complete. Thank you for your purchase!", "total_price": 2375 }

Opis funkcije	Preuzimanje slike visoke rezolucije iz korpe
HTTP metoda	GET
URL	/api/cart/{id}/download
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Korisnik može preuzeti samo slike koje je kupio. Datoteka se preuzima direktno sa servera.
Output	Korisnik preuzima kupljenu sliku direktno sa servera. Nema JSON odgovora, već se vraća fajl (.jpg, .png, .webp).

Opis funkcije	Dodavanje nove slike u sistem
HTTP metoda	POST
URL	/api/pictures
HTTP body parametri	{ "title": "Zalazak sunca", "description": "Prelepi zalazak iznad planine", "category_id": 1, "low_res_path": "images/low_res/sunset.jpg", "high_res_path": "images/high_res/sunset.jpg", "price": 150 }
Format HTTP body parametara	JSON

Napomena	Koristi se samo ako admin već ima putanje do slika. Preporučeni način je preko upload rute.
Output	["Picture is created successfully", { "id": 23, "title": "Zalazak sunca", "description": "Prelepi zalazak iznad planine", "low_res_path": "images/low_res/sunset.jpg", "high_res_path": "images/high_res/sunset.jpg", "price": "150", "category": { "id": 1, "name": "Nature", "description": "Experience the beauty of the natural world through stunning depictions of landscapes, wildlife, forests, mountains, rivers, and serene scenes that capture the essence of our environment. Perfect for nature lovers and those seeking a sense of tranquility.", "created_at": "2025-01-04T11:54:02.000000Z", "updated_at": "2025-01-04T11:54:02.000000Z" } }]]

Opis funkcije	Ažuriranje postojeće slike
HTTP metoda	PUT
URL	/api/pictures/{id}
HTTP body parametri	{ "title": "Zalazak sunca u planini", "description": "Ispravljen opis", "category_id": 1, "low_res_path": "images/low_res/sunset.jpg", "high_res_path": "images/high_res/sunset.jpg", "price": 180 }
Format HTTP body parametara	JSON
Napomena	Admin mora proslediti ceo sadržaj slike, ne samo deo koji želi da izmeni.
Output	["Picture is updated successfully", {

	<pre> "id": 23, "title": "Zalazak sunca u planini", "description": "Ispravljen opis", "low_res_path": "images/low_res/sunset.jpg", "high_res_path": "images/high_res/sunset.jpg", "price": "180", "category": { "id": 1, "name": "Nature", "description": "Experience the beauty of the natural world through stunning depictions of landscapes, wildlife, forests, mountains, rivers, and serene scenes that capture the essence of our environment. Perfect for nature lovers and those seeking a sense of tranquility.", "created_at": "2025-01-04T11:54:02.000000Z", "updated_at": "2025-01-04T11:54:02.000000Z" } }] </pre>
--	---

Opis funkcije	Brisanje slike iz sistema
HTTP metoda	DELETE
URL	/api/pictures/{id}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Brisanje automatski uklanja i fajlove sa diska ako postoje
Output	"Picture and files deleted successfully"

Opis funkcije	Upload nove slike sa fajlovima
HTTP metoda	POST
URL	/api/pictures/upload
HTTP body parametri	<p>title: <i>string</i> – naslov slike</p> <p>description: <i>string</i> (opcionalko) – opis slike</p> <p>category_id: <i>integer</i> – ID kategorije</p> <p>high_res_file: <i>file</i> – fajl visoke rezolucije (jpg, png, max 5MB)</p> <p>low_res_file: <i>file</i> – fajl niske rezolucije (jpg, png, max 2MB)</p> <p>price: <i>decimal</i> – cena slike</p>
Format HTTP body parametara	multipart/form-data

Napomena	Ova ruta automatski smešta fajlove u odgovarajuće foldere i kreira novu sliku sa putanjama.
Output	<pre>{ "message": "Picture uploaded successfully", "picture": { "low_res_path": "images/low_res/probaa.png", "high_res_path": "images/high_res/proba.jpg", "title": "UploadovanaSlika", "description": "Opis uploadovane slike", "category_id": "1", "price": "150", "updated_at": "2025-06-20T09:48:29.000000Z", "created_at": "2025-06-20T09:48:29.000000Z", "id": 24 } }</pre>

Opis funkcije	Kreiranje nove kategorije
HTTP metoda	POST
URL	/api/categories
HTTP body parametri	<pre>{ "name": "Apstraktno", "description": "Moderne apstraktne slike" }</pre>
Format HTTP body parametara	JSON
Napomena	Polje description je opcionalno. Naziv kategorije mora biti jedinstven.
Output	<pre>{ "message": "Category created successfully", "category": { "name": "Apstraktno", "description": "Moderne apstraktne slike", "updated_at": "2025-06-20T09:23:37.000000Z", "created_at": "2025-06-20T09:23:37.000000Z", "id": 11 } }</pre>

Opis funkcije	Ažuriranje postojeće kategorije
HTTP metoda	PUT
URL	/api/categories/{id}
HTTP body parametri	{ "name": "Apstraktna umetnost", "description": "Izmenjeni opis kategorije" }
Format HTTP body parametara	JSON
Napomena	Kategorija se identificuje po svom id-u. Naziv i dalje mora ostati jedinstven (ne sme se poklapati sa drugom kategorijom).
Output	{ "message": "Category updated successfully", "category": { "id": 11, "name": Apstraktna umetnost, "description": "Izmenjeni opis kategorije", "created_at": "2025-06-20T09:23:37.000000Z", "updated_at": "2025-06-20T09:25:55.000000Z" } }

Opis funkcije	Brisanje postojeće kategorije
HTTP metoda	DELETE
URL	/api/categories/{id}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Nije moguće obrisati kategoriju ako ima slike povezane sa sobom. Potrebno je prvo ukloniti slike.
Output	Uspešno brisanje: { "message": "Category deleted successfully" } Neuspešno brisanje: { "error": "Cannot delete category with assigned pictures" }

Opis funkcije	Prikazivanje liste svih korisnika
HTTP metoda	GET
URL	/api/users
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća sve korisnike sistema (ID, ime, email, uloga, datum kreiranja). Dostupno samo adminu.
Output	[{ "id": 16, "name": "Tamara", "email": "tasa1234@gmail.com", "role": "user", "created_at": "2025-05-12T12:47:28.000000Z" }, { "id": 15, "name": "tasa", "email": "tasa@gmail.com", "role": "user", "created_at": "2025-05-11T15:21:22.000000Z" }, { "id": 14, "name": "Djole", "email": "djole@gmail.com", "role": "user", "created_at": "2025-05-11T10:45:10.000000Z" }, { "id": 11, "name": "Katarina", "email": "kaca@gmail.com", "role": "user", "created_at": "2025-05-09T12:12:54.000000Z" }, {

```

    "id": 10,
    "name": "Luka",
    "email": "luka@gmail.com",
    "role": "user",
    "created_at": "2025-03-11T15:45:48.000000Z"
},
{
    "id": 9,
    "name": "Jana",
    "email": "jana@gmail.com",
    "role": "admin",
    "created_at": "2025-03-11T15:41:28.000000Z"
},
.....
]

```

Opis funkcije	Dohvatanje informacija o konkretnom korisniku
HTTP metoda	GET
URL	/api/users/{id}
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Ako korisnik sa zadatim ID-jem ne postoji, vraća se poruka o grešci. Dostupno samo adminu.
Output	<p>Za id= 10:</p> <pre>{ "id": 10, "name": "Luka", "email": "luka@gmail.com", "email_verified_at": "2025-03-11T15:45:48.000000Z", "created_at": "2025-03-11T15:45:48.000000Z", "updated_at": "2025-03-11T15:45:48.000000Z", "role": "user" }</pre> <p>Za nepostojeći id:</p> <p>"Data not found"</p>

Opis funkcije	Dohvatanje paginiranih slika sa brojem po stranici
HTTP metoda	GET
URL	/api/paginate
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Ruta vraća slike raspoređene po stranama. Ako se per_page ne navede, koristi se podrazumevana vrednost 4.
Output	<pre>{ "current_page": 1, "data": [{ "id": 1, "title": "Mountain Sunset", "description": "A breathtaking view of the sun setting behind majestic mountains.", "low_res_path": "http://127.0.0.1:8000/images/low_res/devetal.png", "high_res_path": "http://127.0.0.1:8000/images/high_res/deveta.png", "price": "150.00", "downloads": 36, "created_at": "2025-01-04T11:54:02.000000Z", "updated_at": "2025-05-11T15:53:27.000000Z", "category_id": 1 }, { "id": 2, "title": "Forest Path", "description": "A peaceful trail winding through a lush, green forest.", "low_res_path": "images/low_res/osmal.png", "high_res_path": "images/high_res/osma.png", "price": "200.00", "downloads": 3, "created_at": "2025-01-04T11:54:02.000000Z", "updated_at": "2025-06-17T12:17:39.000000Z", "category_id": 1 }, { "id": 3, "title": "Desert Dunes", "description": "A vast, sandy desert landscape with large sand dunes under a clear sky.", "low_res_path": "images/low_res/osma.png", "high_res_path": "images/high_res/osmal.png", "price": "180.00", "downloads": 25, "created_at": "2025-02-15T10:30:00.000000Z", "updated_at": "2025-07-03T14:20:00.000000Z", "category_id": 2 }] }</pre>

```
        "title": "Ocean Waves",
        "description": "Gentle waves crashing onto a serene beach at sunset.",
        "low_res_path": "images/low_res/sedmal.png",
        "high_res_path": "images/high_res/sedma.png",
        "price": "175.00",
        "downloads": 3,
        "created_at": "2025-01-04T11:54:02.000000Z",
        "updated_at": "2025-06-17T12:17:42.000000Z",
        "category_id": 1
    },
    {
        "id": 4,
        "title": "Mountain Field",
        "description": "A vibrant field nestled between towering mountain peaks.",
        "low_res_path": "images/low_res/desetal.png",
        "high_res_path": "images/high_res/deseta.png",
        "price": "250.00",
        "downloads": 4,
        "created_at": "2025-01-04T11:54:02.000000Z",
        "updated_at": "2025-05-12T11:48:48.000000Z",
        "category_id": 1
    }
],
"per_page": 4,
"total": 15,
"last_page": 4
}
```

Opis funkcije	Prikaz ukupne prodaje po kategorijama
HTTP metoda	GET
URL	/api/sales-by-category
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća naziv kategorije, ukupan iznos prodaje, broj prodatih slika i prosečnu cenu po kategoriji.
Output	[{ "category_name": "Nature", "total_sales": "8315.00", "total_sales_count": 42, "average_sale": "197.976190" }, { "category_name": "Portrait", "total_sales": "7950.00", "total_sales_count": 16, "average_sale": "496.875000" }, { "category_name": "Abstract", "total_sales": "2560.00", "total_sales_count": 9, "average_sale": "284.444444" }]

Opis funkcije	Prikaz top 5 korisnika po broju kupovina
HTTP metoda	GET
URL	/api/top-buyers
HTTP body parametri	nema
Format HTTP body parametara	N/A
Napomena	Vraća listu 5 korisnika koji su najviše puta kupovali, zajedno sa brojem kupovina. Korisno za administrativne uvide i analitiku.
Output	[{ "name": "Katarina", "purchase_count": 36 }, { "name": "User", "purchase_count": 20 }, { "name": "tasa", "purchase_count": 5 }, { "name": "Tamara", "purchase_count": 3 }, { "name": "Tamara", "purchase_count": 2 }]

3.3.2.5. Laravel Database Seeder

U Laravelu, **seederi** se koriste za automatsko punjenje baze podataka testnim ili početnim podacima. Ovo je korisno prilikom:

- inicijalne postavke baze,
- testiranja funkcionalnosti,
- demonstracije rada aplikacije.

Seederi se nalaze u direktorijumu: *database/seeders/*

Primer:

```
public function run()

{
    $user=User::factory(5)->create();

    $category1 = Category::create(['name' => 'Nature','description' => 'Experience the
beauty of the natural world through stunning depictions of landscapes, wildlife, forests,
mountains, rivers, and serene scenes that capture the essence of our environment. Perfect for
nature lovers and those seeking a sense of tranquility.']);

    $category2 = Category::create(['name' => 'Abstract','description' => 'Immerse yourself
in a world of creativity and imagination with abstract art. Featuring bold colors, dynamic
shapes, and intricate patterns, these pieces evoke emotions and inspire thought, making them
perfect for modern spaces and art enthusiasts.']);

    $category3 = Category::create(['name' => 'Portraits','description' => 'Discover the
art of human expression through captivating portraits. From realistic depictions to
imaginative interpretations, these artworks capture the emotions, personalities, and stories
of individuals, bringing them to life with every detail.']);

    $picturesNature = [
        [
            'title' => 'Mountain Sunset',
            'low_res_path' => 'images/low_res/devetal.png',
            'high_res_path' => 'images/high_res/deveta.png',
            'price' => 150.00,
            'description' => 'A breathtaking view of the sun setting behind majestic
mountains.',
        ]
    ];
}
```

```

];
foreach ($picturesNature as $picture) {

    Picture::create(array_merge($picture, ['category_id' => $category1->id,
'downloads' => rand(0, 50)]));

}

$picturesAbstract = [
[
    'title' => 'Color Burst',
    'low_res_path' => 'images/low_res/trecal.png',
    'high_res_path' => 'images/high_res/treca.png',
    'price' => 300.00,
    'description' => 'A dynamic explosion of vibrant colors and bold shapes.',
]
];
foreach ($picturesAbstract as $picture) {

    Picture::create(array_merge($picture, ['category_id' => $category2->id,
'downloads' => rand(0, 50)]));

}

$picturesPortraits = [
[
    'title' => 'Artistic Woman',
    'low_res_path' => 'images/low_res/petnaestal.png',
    'high_res_path' => 'images/high_res/petnaesta.png',
    'price' => 500.00,
    'description' => 'A striking portrait of a woman with an artistic flair.',
]
];
foreach ($picturesPortraits as $picture) {

```

```

        Picture::create(array_merge($picture, ['category_id' => $category3->id,
'downloads' => rand(0, 50)]));

    }

}

```

3.3.2.6. React Hooks

React Hooks su funkcije koje omogućavaju korišćenje **React funkcionalnosti** (kao što su stanje i efekti) unutar **funkcionalnih komponenti**, bez potrebe za klasama. Uvedeni su u verziji React 16.8, i od tada su postali standard za pisanje modernih React aplikacija.

Custom Hooks su **twoje funkcije** koje počinju sa *use* i koriste druge hook-ove unutra (*useState*, *useEffect*, itd.).

Njihova svrha je ponovno korišćenje logike kroz više komponenti na jasan način.

Primer:

```

import { useState } from "react";

function useLocalStorage(key, initialValue) {
    const [storedValue, setStoredValue] = useState(() => {
        try {
            const item = window.localStorage.getItem(key);
            return item ? JSON.parse(item) : initialValue;
        } catch (error) {
            console.error("Error reading localStorage key:", key, error);
            return initialValue;
        }
    });
    const setValue = (value) => {
        try {
            const valueToString =
                value instanceof Function ? value(storedValue) : value;
            setStoredValue(valueToString);
            window.localStorage.setItem(key, JSON.stringify(valueToString));
        }
    };
    return [storedValue, setValue];
}

```

```
    } catch (error) {
      console.error("Error setting localStorage key:", key, error);
    }
  );
  return [storedValue, setValue];
}
export default useLocalStorage;
```

3.3.2.7 React Komponente (Components)

U React-u, **komponente** su **osnovne gradivne jedinice korisničkog interfejsa (UI)**. One predstavljaju **ponovno iskoristive delove koda** koji mogu sadržati HTML, CSS i JavaScript logiku u okviru jedne funkcije ili klase.

Komponente mogu biti jednostavne (prikaz jedne slike ili dugmeta), ali i složene (forme, stranice, tabele, galerije).

Svaka komponenta može da sadrži:

- *JSX* - HTML unutar JavaScript-a
- *Props* - Ulagani podaci koje komponenta prima
- *State* - Interno stanje komponente (koristi *useState*)
- *Events* - Reakcija na klik, submit, hover itd.
- *Lifecycle* metode - Automatske funkcije tokom "života" komponente (npr. *useEffect*)

Primer HTML-a unutar JavaScript-a - JSX:

JSX omogućava pisanje HTML strukture direktno u JavaScript kodu komponente. U ovom primeru koristi se HTML (`<nav>`, `<div>`, ``, ``, `<a>`) unutar React funkcije, uz dodatnu logiku za prikaz linkova u zavisnosti od stanja (token). Klase se pišu kao `className`, a dinamičke vrednosti i uslovi se koriste unutar `{}`.

```
<nav className="navbar navbar-expand-xl navbar-dark bg-dark">

  <div className="container-fluid">

    <a className="navbar-brand" href="/">
      Digital art store
    </a>

    <button
      className="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarBasic"
      aria-controls="navbarBasic"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >

      <span className="navbar-toggler-icon"></span>
    </button>

    <div className="collapse navbar-collapse show" id="navbarBasic">

      <ul className="navbar-nav me-auto mb-2 mb-xl-0">

        <li className="nav-item">
          <Link
            to="/gallery"
            className={`${nav-link ${

              location.pathname === "/gallery" ? "active" : ""`}}`>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

```
    }`}

>

  Gallery

</Link>

</li>

<li className="nav-item">

  <Link

    to="/purchase-history"

    className={`${nav-link ${

      location.pathname === "/purchase-history" ? "active" : ""

    }}`}

>

  Purchase History

</Link>

</li>

</>

) `

</div>

</nav>
```

Primer prikaza podataka kroz props:

```
const Pictures = ({  
  
  pictures,  
  
  onAdd,  
  
  selectedCategory,  
  
  searchText,  
  
  onPictureClick,  
  
  favorites,  
  
  toggleFavorite,  
  
  token,  
  
  currentPage,  
  
  setCurrentPage,  
  
}) => { ... }
```

Primer LoginPage komponente:

Komponenta LoginPage koristi **useState** za Čuvanje podataka o korisniku, **useNavigate** za redirekciju nakon uspešnog logovanja i **axios** za pozivanje backend rute /api/login.

```
const [userData, setUserData] = useState({  
  
  email: "",  
  
  password: "",  
  
}) ;  
  
  
let navigate = useNavigate();  
  
  
function handleLogin(e) {  
  
  e.preventDefault();  
  
  axios  
    .post("api/login", userData)  
    .then((res) => {
```

```
console.log("Login response:", res.data);

if (res.data.success === true) {

    const token = res.data.access_token;
    const role = res.data.user.role;

    window.sessionStorage.setItem("auth_token", token);
    window.sessionStorage.setItem("user_role", role);
    addToken(token);

    if (role === "admin") {
        navigate("/admin/dashboard");
    } else {
        navigate("/gallery");
    }
}

})
.catch((e) => {
    console.log(e);
    alert("Login failed!");
});
}
```

Primer Lifecycle metode – useEffect za učitavanje podataka:

useEffect se ovde koristi za automatsko učitavanje istorije kupovine kada se komponenta **PurchaseHistory** prikaže.

```
useEffect(() => {  
  
  fetch("http://127.0.0.1:8000/api/purchase-history", {  
  
    headers: {  
  
      Authorization: `Bearer ${token}`,  
  
    },  
  
  })  
  
.then((res) => res.json())  
  
.then((data) => setHistory(data))  
  
.catch((err) => console.error("Error loading purchase history:", err));  
  
}, []);
```

Primer reakcije na klik i submit - Events:

Kada korisnik klikne na ikonicu srca, poziva se funkcija **toggleFavorite(picture)**. Boja srca se menja u zavisnosti od stanja **isFavorite**.

```
<button  
  
  className="btn btn-link p-0"  
  
  onClick={() => toggleFavorite(picture)}  

```

3.3.2.8. React Rutiranje (BrowserRouter)

Rutiranje omogućava aplikaciji da se ponaša kao višestranična SPA aplikacija (*Single Page Application*), iako se zapravo ne učitavaju nove HTML stranice – samo se menja sadržaj komponente u zavisnosti od URL putanje.

U React-u se za rutiranje najčešće koristi biblioteka *react-router-dom*.

Osnovne komponente rutiranja:

- `<BrowserRouter>` - glavna komponenta koja "omotava" aplikaciju i omogućava rad sa URL-om
- `<Routes>` - definiše sve moguće rute (putanje)
- `<Route path="..." />` - određuje koja komponenta će biti prikazana za koji URL
- `useLocation()` - hook koji vraća trenutni URL (koristan za logiku u meniju, breadcrumb, itd.)

Primer:

```
return (
  <>
  {!isAdmin && (
    <NavBar
      cartNum={cartNum}
      categories={categories}
      onFilter={handleFilter}
      selectedCategory={selectedCategory}
      onSearch={handleSearch}
      token={token}
      onLogout={handleLogout}
    />
  )} {!isAdmin && (
    <Breadcrumbs
  )}
)
```

```

        selectedCategory={selectedCategory}

        resetCategory={resetCategory}

    />

) }

<Routes>

<Route path="/" element={<HomePage />} />

<Route path="/login" element={<LoginPage addToken={addToken} />} />

<Route path="/register" element={<RegisterPage />} />

<Route

    path="/gallery"

    element={

        <Pictures

            pictures={pictures}

            onAdd={addPicture}

            selectedCategory={selectedCategory}

            searchText={searchText}

            onPictureClick={setSelectedPicture}

            favorites={favorites}

            toggleFavorite={toggleFavorite}

            token={token}

            currentPage={currentPage}

            setCurrentPage={setCurrentPage}

        />

    }

/>

<Route

    path="/cart"

```

```

element={

  <Cart

    pictures={cartPictures}

    remove={removePicture}

    updateCartNum={updateCartNum}

    resetCategory={resetCategory}

    resetSearch={resetSearch}

  />

}

/>

<Route path="/purchase-history" element={<PurchaseHistory />} />

<Route

  path="/admin"

  element={<AdminLayout onLogout={handleLogout} />}

>

<Route path="dashboard" element={<AdminDashboard />} />

<Route path="add-picture" element={<AdminAddPicture />} />

<Route path="manage-pictures" element={<AdminManagePictures />} />

<Route path="edit-picture/:id" element={<AdminEditPicture />} />

<Route path="add-category" element={<AdminAddCategory />} />

<Route path="preview" element={<AdminPreviewGallery />} />

<Route path="sales-chart" element={<AdminSalesChart />} />

<Route path="users" element={<AdminUserList />} />

<Route

  path="manage-categories"

  element={<AdminManageCategories />}

/>

```

```

        </Route>

        <Route
            path="/favorites"
            element={
                <Favorites
                    favorites={favorites}
                    onAdd={addPicture}
                    toggleFavorite={toggleFavorite}
                />
            }
        />

    </Routes>

    {selectedPicture && (
        <Modal picture={selectedPicture} onClose={closeModal} />
    )}

    {!isAdmin && <Footer />}

```

</>

) ;

}

return (

<BrowserRouter>

<AppContent />

</BrowserRouter>

) ;

3.3.2.9 Laravel i React

U ovoj aplikaciji, Laravel se koristi kao backend (API servis), a React kao frontend (korisnički interfejs). Povezivanje između njih se ostvaruje pomoću REST API poziva koje React aplikacija šalje Laravel serveru.

- React (frontend) → šalje HTTP zahteve (GET, POST, DELETE...)
- Laravel (backend) → prima zahtev, obrađuje ga i vraća odgovor (JSON)

Axios je JavaScript biblioteka za slanje HTTP zahteva.

Koristi se za komunikaciju između **frontend-a (React)** i **backend-a (Laravel)**.

Axios omogućava:

- slanje *GET, POST, PUT, DELETE* zahteva
- automatsko pretvaranje podataka iz *JSON* formata
- jednostavno upravljanje zagлавljima (*headers*), greškama i tokenima.

Uvoz u komponentu:

```
import axios from "axios";
```

Tipični slučajevi povezivanja:

Radnja na React-u	Ruta na Laravel strani	HTTP metoda	Opis odgovora
Dohvatanje svih slika	/api/pictures	GET	Vraća JSON niz svih slika
Logovanje korisnika	/api/login	POST	Vraća token i korisničke podatke
Dodavanje slike u korpu	/api/cart	POST	Kreira stavku u korpi
Preuzimanje kupljene slike	/api/cart/{id}/download	GET	Vraća fajl (sliku)
Brisanje kategorije	/api/categories/{id}	DELETE	Vraća poruku o uspehu ili grešci

React koristi **Bearer** token (iz *sessionStorage*) za autentifikaciju API zahteva. Laravel koristi *Sanctum* za validaciju.

```
function handleLogin(e) {  
  e.preventDefault();  
  
  axios  
    .post("api/login", userData)  
    .then((res) => {  
      console.log("Login response:", res.data);  
  
      if (res.data.success === true) {  
        const token = res.data.access_token;  
        const role = res.data.user.role;  
  
        window.sessionStorage.setItem("auth_token", token);  
        window.sessionStorage.setItem("user_role", role);  
        addToken(token);  
  
        if (role === "admin") {  
          navigate("/admin/dashboard");  
        } else {  
          navigate("/gallery");  
        }  
      }  
    })  
    .catch((e) => {  
      console.log(e);  
      alert("Login failed!");  
    });  
}  
}
```

Primer prikaza svih slika:

- React kod

```
useEffect(() => {
  axios
    .get("/api/pictures")
    .then((res) => {
      console.log("Slike učitane:", res.data);
      setPictures(res.data.pictures || []);
    })
    .catch((error) => {
      console.error("Greška pri dobijanju slika:", error);
    });
}, []);
```

- Laravel ruta

```
Route::get('pictures', [PictureController::class, 'index']);
```

- Laravel kontroler

```
public function index() {
  $pictures = Picture::with('category')->get()->map(function ($picture) {
    return [
      'id' => $picture->id,
      'title' => $picture->title,
      'description' => $picture->description,
      'category_id' => $picture->category_id,
      'category_name' => $picture->category ? $picture->category->name : "Unknown",
      'price' => $picture->price,
      'low_res_path' => asset($picture->low_res_path),
      'high_res_path' => asset($picture->high_res_path),
    ];
  });
  return response()->json(['pictures' => $pictures], 200);
}
```

3.3.3. Projektovanje strukture softverskog sistema

3.3.3.1. Migracije

Migracije u Laravelu predstavljaju **verzije šeme baze podataka**, koje se prate kroz PHP fajlove. One omogućavaju timski rad, testiranje, vraćanje (rollback), i lako podešavanje strukture baze kroz kod.

Migracije se nalaze u folderu: *database/migrations/*

Za pokretanje migracija koristi se komanda:

php artisan migrate

Za vraćanje prethodnog stanja baze:

php artisan migrate:rollback

Svaka migracija ima dve metode:

1. *up()* - Izvodi promene (npr. kreira tabelu, dodaje kolonu, postavlja relaciju)
2. *down()* - Vraća promene (npr. briše tabelu ili kolonu, uklanja relaciju)

Primer:

Migracija *create_users_table.php*

```
Schema::create('users', function (Blueprint $table) {  
  
    $table->id();  
  
    $table->string('name');  
  
    $table->string('email');  
  
    $table->timestamp('email_verified_at')->nullable();  
  
    $table->string('password');  
  
    $table->rememberToken();  
  
    $table->timestamps();  
  
});
```

Primer migracije za relacije i strane ključeve:

```
Schema::table('pictures', function (Blueprint $table) {  
  
    $table->foreignId('category_id')->constrained()->onDelete('cascade');  
  
});  
  
Schema::table('carts', function (Blueprint $table) {  
  
    $table->foreignId('user_id')->constrained()->onDelete('cascade');  
  
    $table->foreignId('picture_id')->constrained()->onDelete('cascade');  
  
});
```

3.3.3.2. Modeli

Modeli u Laravelu predstavljaju **sloj aplikacije koji komunicira sa bazom podataka**. Svaki model je povezan sa konkretnom tabelom i omogućava:

- Čitanje i upis podataka
- kreiranje veza između tabela (relacija)
- definisanje atributa i pravila ponašanja modela

Modeli se nalaze u folderu: *app/Models/*

U Laravelu modeli koriste *Eloquent ORM*, što znači da se pomoći jednostavnog PHP koda mogu vršiti upiti nad bazom bez potrebe za ručnim *SQL-om*.

Osnovne osobine modela:

- *use HasFactory* - omogućava korišćenje fabričkih podataka (*factories*)
- *protected \$fillable* - lista kolona koje se mogu masovno popunjavati (*mass assignment*)
- *public function xyz()* - definisanje relacija sa drugim modelima (*hasMany, belongsTo, itd.*)

1. *User.php*

```
class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    protected $fillable = [

        'name',
        'email',
        'password',
        'email_verified_at',
        'role'

    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    protected $casts = [
        'email_verified_at' => 'datetime',
    ];

    public function carts() {

        return $this->hasMany(Cart::class);
    }
}
```

2. *Cart.php*

```
class Cart extends Model

{

    use HasFactory;

    protected $fillable = [
        'user_id',
        'picture_id',
        'price_to_pay',
    ];

    public function user() {

        return $this->belongsTo(User::class);
    }

    public function picture() {

        return $this->belongsTo(Picture::class);
    }
}
```

3. *Picture.php*

```
class Picture extends Model

{

    use HasFactory;

    protected $fillable =
['title','price','high_res_path','low_res_path','category_id','description'];

    protected $attributes = [
        'low_res_path' => 'images/low_res/11.png',
        'high_res_path' => 'images/high_res/1.png',
    ];
}
```

```
public function category() {  
  
    return $this->belongsTo(Category::class);  
  
}  
  
public function carts() {  
  
    return $this->hasMany(Cart::class);  
  
}  
}
```

4. Category.php

```
class Category extends Model  
{  
  
    use HasFactory;  
  
    protected $fillable = [  
        'name',  
        'description',  
    ];  
  
    public function pictures() {  
  
        return $this->hasMany(Picture::class);  
    }  
}
```

3.3.4. Projektovanje skladišta podataka

Skladište podataka u Laravel aplikaciji se oslanja na relacioni model baze podataka, koji je izведен iz konceptualnog modela (*PMOV* – prikaz modela objekata i veza). U nastavku je prikazan relacioni model kroz konkretne tabele, njihove kolone i relacije među njima.

1. Tabela users

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 📄	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	name	varchar(255)	utf8mb4_unicode_ci		No	None		
3	email 📩	varchar(255)	utf8mb4_unicode_ci		No	None		
4	email_verified_at	timestamp			Yes	NULL		
5	password	varchar(255)	utf8mb4_unicode_ci		No	None		
6	remember_token	varchar(100)	utf8mb4_unicode_ci		Yes	NULL		
7	created_at	timestamp			Yes	NULL		
8	updated_at	timestamp			Yes	NULL		
9	role	varchar(255)	utf8mb4_unicode_ci		No	user		

2. Tabela pictures

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 📄	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	title	varchar(255)	utf8mb4_unicode_ci		No	None		
3	description	text	utf8mb4_unicode_ci		Yes	NULL		
4	low_res_path	varchar(255)	utf8mb4_unicode_ci		No	images/low_res/default.png		
5	high_res_path	varchar(255)	utf8mb4_unicode_ci		No	images/high_res/default.png		
6	price	decimal(10,2)			No	None		
7	downloads	bigint(20)		UNSIGNED	No	0		
8	created_at	timestamp			Yes	NULL		
9	updated_at	timestamp			Yes	NULL		
10	category_id 📁	bigint(20)		UNSIGNED	No	None		

3. Tabela categories

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	name 	varchar(255)	utf8mb4_unicode_ci		No	None		
3	description	text	utf8mb4_unicode_ci		Yes	NULL		
4	created_at	timestamp			Yes	NULL		
5	updated_at	timestamp			Yes	NULL		

4. Tabela carts

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	id 	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT
2	price_to_pay	decimal(20,2)			No	0.00		
3	created_at	timestamp			Yes	NULL		
4	updated_at	timestamp			Yes	NULL		
5	user_id 	bigint(20)		UNSIGNED	No	None		
6	picture_id 	bigint(20)		UNSIGNED	No	None		

3.4. Faza implementacije

U ovoj fazi implementirana je kompletna struktura projekta korišćenjem Laravel-a za backend i React-a za frontend. Projekat je podeljen u dva glavna dela, pri čemu svaki deo ima jasno definisani organizaciju fajlova radi lakšeg održavanja i razvoja.

Frontend – React aplikacija

Frontend aplikacija se nalazi u direktorijumu *react-prodavnica*. Glavni direktorijum za logiku aplikacije je *src*, koji sadrži:

- **components/** – svi višekratni prikazni elementi aplikacije poput *NavBar*, *Cart*, *HomePage*, *Modal*, *Pictures*, itd.
- **pages/admin/** – admin deo aplikacije sa komponentama za dodavanje i izmenu slika, kategorija i korisnika (*AdminAddPicture*, *AdminManageCategories*, *AdminUserList*...).
- **hooks/** – korisnički definisani React hookovi, poput *useLocalStorage*.
- **App.js** – glavni fajl aplikacije u kojem se nalaze sve rute.
- **index.js** – je fajl u kojem se React aplikacija povezuje sa HTML-om i pokreće.

React koristi biblioteku *axios* za slanje HTTP zahteva ka backendu (*Laravel API*).

Organizacija fajlova:

```
react-prodavnica/
|   └── public/
|       |   └── index.html
|       |   └── favicon.ico
|       |   └── manifest.json
|       └── robots.txt
└── src/
    └── components/
        |   └── AdminNavBar.jsx
        |   └── Breadcrumbs.jsx
        |   └── Cart.jsx
        |   └── Favorites.jsx
```

```
|   |   └── Footer.jsx
|   |   └── HomePage.jsx
|   |   └── LoginPage.jsx
|   |   └── Modal.jsx
|   |   └── NavBar.jsx
|   |   └── OnePicture.jsx
|   |   └── Pictures.jsx
|   |   └── PurchaseHistory.jsx
|   |   └── RegisterPage.jsx
|   └── hooks/
|       └── useLocalStorage.jsx
|   └── pages/
|       └── admin/
|           ├── AdminAddCategory.jsx
|           ├── AdminAddPicture.jsx
|           ├── AdminDashboard.jsx
|           ├── AdminEditPicture.jsx
|           ├── AdminLayout.jsx
|           ├── AdminManageCategories.jsx
|           ├── AdminManagePictures.jsx
|           ├── AdminPreviewGallery.jsx
|           ├── AdminSalesChart.jsx
|           ├── AdminUserList.jsx
|           └── RequireAdmin.jsx
|   └── App.js
|   └── App.css
|   └── index.js
|   └── logo.svg
|   └── ...
|── package.json
|── package-lock.json
```

```
└── README.md
```

Backend – Laravel aplikacija

Backend je razvijen u Laravel PHP framework-u i nalazi se u direktoriju *prodavnica_digitalnih_proizvoda*.

Ključni folderi su:

- **app/Http/Controllers** – sadrži sve kontrolere, podeljene na API (*CartController*, *CategoryController*, *PictureController* itd.).
- **app/Models/** – svi modeli (*User*, *Picture*, *Category*, *Cart*), koji definišu odnose između tabela.
- **database/migrations/** – migracije za pravljenje i modifikaciju tabela.
- **database/seeders/** – seeder fajlovi za popunjavanje baza početnim podacima.
- **routes/api.php** – definisane sve API rute koje frontend poziva.

Laravel koristi middleware za zaštitu API ruta, tako da samo autentifikovani korisnici (ili administratori) mogu pristupiti određenim funkcijama (npr. dodavanje slika, brisanje korisnika).

Organizacija fajlova:

```
prodavnica_digitalnih_proizvoda/
```

```
└── app/
    ├── Console/
    │   └── Kernel.php
    ├── Exceptions/
    │   └── Handler.php
    ├── Http/
    │   ├── Controllers/
    │   │   ├── API/
    │   │   │   ├── AuthController.php
    │   │   │   ├── CartController.php
    │   │   │   ├── CategoryController.php
```

```
| | | | └── PictureController.php  
| | | | └── SalesController.php  
| | | └── UserController.php  
| | └── Controller.php  
| └── Middleware  
| └── Kernel.php  
└── Models/  
    ├── Cart.php  
    ├── Category.php  
    ├── Picture.php  
    └── User.php  
└── Providers/  
    ├── AppServiceProvider.php  
    ├── AuthServiceProvider.php  
    ├── BroadcastServiceProvider.php  
    ├── EventServiceProvider.php  
    └── RouteServiceProvider.php  
└── bootstrap/  
└── config/  
└── database/  
    ├── factories/  
    |   ├── CartFactory.php  
    |   ├── CategoryFactory.php  
    |   ├── PictureFactory.php  
    |   └── UserFactory.php  
    └── migrations/  
        ├── 2014_10_12_000000_create_users_table.php  
        ├── 2014_10_12_100000_create_password_resets_table.php  
        ├── 2019_12_14_000001_create_personal_access_tokens_table.php  
        ├── 2024_12_24_104734_create_pictures_table.php  
        └── 2024_12_25_105155_create_categories_table.php
```

```
|   |   └── 2024_12_27_094424_create_carts_table.php
|   |   └── 2024_12_27_105526_add_description_to_pictures_table.php
|   |   └── 2024_12_27_110125_drop_tags_from_pictures_table.php
|   |   └── 2025_01_04_073730_add_foreign_keys_to_tables.php
|   |   └── 2025_01_04_081537_modify_downloads_column_in_pictures_table.php
|   |   └── 2025_01_05_082205_add_role_to_users_table.php
|   └── 2025_01_11_153102_add_unique_to_email_in_users_table.php
|   └── seeders/
|       └── DatabaseSeeder.php
└── public/
    ├── images/
    |   ├── high_res/
    |   └── low_res/
    ├── favicon.ico
    ├── index.php
    └── robots.txt
    └── resources/
        └── views/ (prazno ili po potrebi)
    └── routes/
        ├── api.php
        ├── channels.php
        ├── console.php
        └── web.php
    └── storage/
    └── tests/
    └── vendor/
    └── .env
    └── .gitignore
    └── artisan
    └── composer.json
    └── composer.lock
```

```
└── package.json  
└── package-lock.json  
└── phpunit.xml
```

Organizacija koda u ovom projektu omogućava odvajanje odgovornosti između frontend-a i backend-a. React aplikacija vodi računa o korisničkom interfejsu, dok Laravel backend upravlja podacima i poslovnom logikom.

3.5. Faza testiranja

Nakon implementacije funkcionalnosti frontend i backend delova aplikacije, sprovedena je faza testiranja u cilju verifikacije ispravnosti sistema. Testiranje je obuhvatalo **funkcionalno, ručno i API testiranje**, kao i **osnovnu validaciju sigurnosnih mehanizama**.

Funkcionalno testiranje

Funkcionalnosti su testirane kroz korisnički interfejs koristeći različite scenarije:

- registracija i login korisnika
- pregled galerije i filtriranje slika po kategorijama
- dodavanje slika u korpu
- proces kupovine i validacija unosa podataka za plaćanje
- preuzimanje slika visoke rezolucije samo za ulogovane korisnike
- prikaz istorije kupovine

Svi koraci testirani su u više browsera (Chrome, Firefox), pri čemu nije bilo većih problema.

API testiranje

Za testiranje komunikacije između frontend-a i backend-a korišćen je **Postman**:

- testirani su svi API endpoint-i za slike (*/api/pictures*), kategorije (*/api/categories*), korpu (*/api/cart*), korisnike (*/api/user*) i autentifikaciju (*/api/login*, */api/register*)
- izvršena su *GET*, *POST* i *DELETE* testiranja
- validirani su odgovori (*status kodovi 200, 201, 400, 404, 422...*)

- potvrđeno je da korisnici bez tokena ne mogu pristupiti zaštićenim rutama

Validacija i sigurnosno testiranje

Testirane su forme za:

- registraciju (dužina lozinke, format emaila, nepotpunjena polja)
- login sa netačnim podacima
- unos podataka za plaćanje (kartica, datum isteka, CVV)

Uočeno je da aplikacija pravilno prikazuje korisniku greške i da se nepoželjni podaci ne šalju na backend bez validacije.

Rezultati testiranja

- Sve osnovne funkcionalnosti aplikacije rade ispravno
- Prava pristupa su korektno implementirana (npr. neulogovani korisnik ne može kupovati)
- Nisu uočene kritične greške tokom ručnog testiranja
- API zahtevi su vraćali očekivane odgovore

Testiranjem je potvrđeno da aplikacija ispunjava sve funkcionalne zahteve definisane u toku dizajna. Sistem je stabilan, odgovara očekivanjima korisnika, i spreman je za korišćenje.

4. ZAKLJUČAK

U ovom projektnom radu realizovan je funkcionalan informacioni sistem za prodaju digitalnih umetničkih dela, implementiran korišćenjem savremenih web tehnologija: Laravel kao backend framework i **React** kao frontend biblioteka. Aplikacija omogućava korisnicima da na jednostavan i intuitivan način pregledaju, filtriraju, pretražuju i kupuju digitalne radove, dok administratorima pruža funkcionalnosti za upravljanje sadržajem i korisnicima.

Kroz implementaciju sistema postignuti su sledeći rezultati:

- Napravljen je **interaktivni korisnički interfejs** sa podrškom za prikaz slika u više rezolucija, filtriranje po kategorijama i prikaz istorije kupovine.
- **RESTful API** omogućava razmenu podataka između klijentske i serverske strane.
- Implementirana je **autentifikacija i autorizacija korisnika** pomoću *Laravel Sanctum-a*, čime je omogućeno razlikovanje običnih korisnika i administratora.
- Omogućeno je **preuzimanje kupljenih slika** visoke rezolucije samo onim korisnicima koji su ih zaista kupili.
- Napravljena je **admin sekcija** koja omogućava upravljanje slikama, kategorijama i pregled podataka o prodaji.
- Organizovana je struktura koda u skladu sa **principima dobre softverske arhitekture**, što omogućava lako održavanje i buduće proširenje funkcionalnosti.

Backend aplikacija je izgrađena u **Laravel PHP framework-u**, koji je omogućio brzu i efikasnu izradu API-ja, validaciju podataka, rad sa bazom kroz *Eloquent ORM*, kao i upotrebu middleware-a za zaštitu ruta.

Frontend aplikacija, razvijena u **React-u**, koristi komponente za modularni razvoj, *react-router-dom* za rutiranje i *axios* biblioteku za komunikaciju sa backend-om. Korisnički podaci (kao što su tokeni i omiljene slike) čuvaju se pomoću lokalnog skladišta (*localStorage*) uz pomoć *custom hook-a (useLocalStorage)*, čime je omogućen bolji korisnički doživljaj.

Tokom razvoja, susrele smo se sa različitim izazovima koji su zahtevali dodatno istraživanje i učenje:

1. **Integracija React i Laravel okruženja** – U početku je bilo izazovno pravilno konfigurisati komunikaciju između frontend i backend delova aplikacije. Poseban izazov predstavljalo je rukovanje autentifikacionim tokenima i slanje autorizovanih zahteva
2. **Prikaz i čuvanje slika** – Kod rada sa fajlovima visoke rezolucije, bilo je potrebno naučiti kako da se pravilno čuvaju, prikazuju i preuzimaju uz poštovanje pravila pristupa i bezbednosti.

3. **Organizacija projekta** – Kako je aplikacija rasla, postajalo je sve važnije držati se jasne strukture i konvencija, naročito kod razdvajanja logike za korisnike i administratore.
4. **Upravljanje stanjem u React-u** – Praćenje broja stavki u korpi, omiljenih slika i trenutnih filtera zahtevalo je pažljivo rukovanje stanjima, što je dodatno usložnjeno kombinovanjem React-a sa localStorage-om i API podacima.

Rad na ovom projektu bio je izazovan, ali izuzetno koristan. Naučile smo kako da povežemo frontend i backend aplikaciju u jednu funkcionalnu celinu. Stekle smo praktično znanje o:

- REST API komunikaciji
- upravljanju autentifikacijom
- radu sa bazama podataka kroz ORM
- organizaciji kompleksnog React projekta sa komponentama, stranicama i rutama
- obradi korisničkih zahteva i validaciji
- kao i pisanju koda koji je čitljiv, organizovan i skalabilan

Kroz realizaciju ovog projekta napravile smo aplikaciju koja se može koristiti kao osnova za komercijalni servis. U budućnosti, projekat se može dodatno unaprediti:

- uvođenjem sistema za ocenjivanje i komentarisanje slika
- implementacijom sistema za slanje email potvrda nakon kupovine
- dodavanjem mogućnosti za višejezičnost
- dodavanjem naprednih analitika za praćenje prodaje i aktivnosti korisnika

5. LITERATURA

1. React -<https://react.dev>
2. Laravel -[Installation - Laravel 12.x - The PHP Framework For Web Artisans](#)
3. JavaScript -[MDN Web Docs](#)
4. Bootstrap - [Bootstrap · The most popular HTML, CSS, and JS library in the world.](#)
5. Materijali u e-formi, sa sajta www.elab.rs