Internet aplikacija za podršku kupovnih kartica

Barišić, Nikola

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni studij

INTERNET APLIKACIJA ZA PODRŠKU KUPOVNIH KARTICA

Diplomski rad

Nikola Barišić

Osijek, 2018.

SADRŽAJ

1. UVOD

Zadatak diplomskog rada izrada je internet aplikacije za podršku kupovnih kartica. Tema je odabrana zbog lakšeg snalaženja u svakodnevni budući da se kupovne kartice svakodnevno koriste. Diplomski rad zahtjeva izradu API-ja u server strani te razne forme korisničkog sučelja za krajnjeg korisnika. Internet aplikacija napravljena je s ciljem prikupljanja korisnikovih podataka o kupovnim karticama, na način da je osmišljena baza podataka koja sadrži informacije o korisniku i kupovnim karticama te omogućava lakše snalaženje. Za uspješnu realizaciju diplomskog rada bilo je potrebno realizirati server stranu koja ostvaruje komunikaciju s bazom podataka. Server strana korisniku omogućuje manipulaciju s podacima dok korisničko sučelje omogućuje prikaz željenih podataka iz baze podataka. Tijekom izrade aplikacije korištena su znanja stečena tijekom pet godina studiranja na Fakultetu elektrotehnike računarstva i informacijskih tehnologija u Osijeku. Također se morao potražiti dodatan izvor informacija budući da je internet aplikacija izrađena u programskom okviru Adonisu.js baziranom na Node.js. U konačnici, potrebno je izraditi i dokumentaciju kojom će se detaljno objasniti postupak izrade internet aplikacije za podršku kupovnih kartica. U diplomskom radu objašnjen je koncept vlastitog rješenja navedenog zadatka. Diplomski rad je podijeljen u četiri poglavlja; prvo je uvodno, drugo poglavlje navodi i objašnjava korištene tehnologije za izradu diplomskog rada, u trećem poglavlju opisan je koncept vlastitog rješenja te njegova implementacija dok je s završnim četvrtim poglavljem dan zaključak.

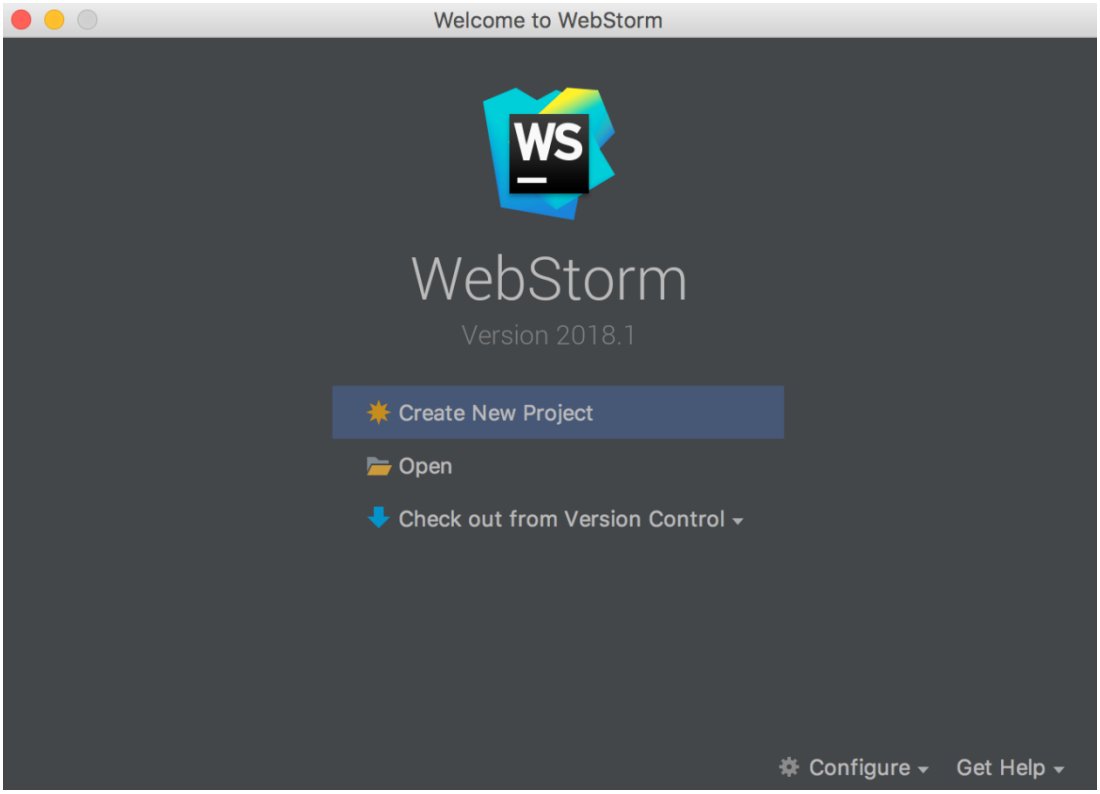
1.1. Zadatak diplomskog rada

Zadatak diplomskog rada bio je napraviti AdonisJs internet aplikaciju koja će pružiti korisniku jednostavno pohranjivanje kupovnih kartica i međusobno dijeljenje kupovnih kartica s drugim korisnicima internet aplikacije. Ideja je da korisnik dobije funkcionalni sustav. Kompletni sustav izrađen je u web programskom okviru AdonisJs te je korištena MySQL baza podataka za međusobno povezivanje tablica.

2. OPIS KORIŠTENIH TEHNOLOGIJA

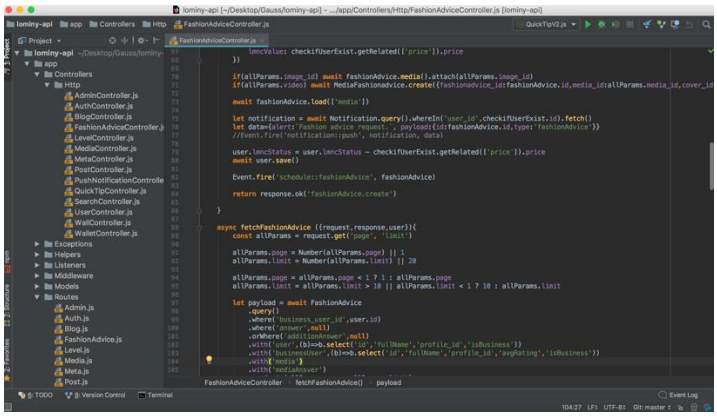
2.1. WebStorm IDE

WebStorm je inteligentan IDE koji pruža pomoć pri pisanju programskog koda za JavaScript, HTML i CSS te široku lepezu modernih web tehnologija uključujući i AdonisJs. WebStorm obuhvaća klijent-stranu prilikom razvoja programskog koda i razvoja na server strani.



Slika 2.1. Izgled WebStorm IDE-a

WebStorm omogućuje uvid u napisane kodove, automatsko dovršenje, značajke ispravaka pogrešaka, prevenciju pogrešaka, višestruko mijenjanje varijabli, pretraživanje i još mnogo toga. Također, integrira instrumente za testiranje, ispravljanje pogrešaka te kodiranje i integraciju s raznim VCS (Slika 2.1) [1].



Slika 2.2. Izgled WebStorm razvojnog okruženja

2.2. AdonisJs

AdonisJs je Node.js web programski okvir koji radi na svim većim operacijskim sustavima. Razvijen je 2017. godine od strane Harminder Virka te nudi stabilno okruženje za pisanje internet aplikacije na strani poslužitelja kako bi mogli obratiti pažnju na poslovne potrebe [2]. AdonisJs pruža programeru punu podršku prilikom izrade server strane ili korisničkog sučelja internet aplikacije. Trenutna verzija programski okvira je 4.1. dok je verzija 5.0. u razvoju. Uvjeti za uspješno korištenje AdonisJs programskog okvira su poznavanje objektno orijentiranog programiranja, razumijevanje JavaScripta, Async programiranje i Node.js. AdonisJs je modularni programski okvir koji sadrži više pružatelja usluga - građevni blokovi AdonisJs internet aplikacije koji se kao bilo koji drugi npm paket nalaze u zaglavlju koda. AdonisJs se razlikuje od Expressa i Koa po tome što je jednostavniji za korištenje jer slijedi skup standardiziranih konvencija olakšavajući programeru lakše snalaženje u tuđem kodu. Za uspješno korištenje AdonisJs programskog okvira potrebno je imati minamlno Node.js 8.0 verziju te minimalno Npm 3.0 verziju. Nakon što ispunimo zahtjeve za pokretanjem AdonisJs aplikacije, sljedeći korak je instalacija cli alata koji nam pomažu u stvaranju novih AdonisJs projekata. Folderi su strukturirani na sljedeći način:

- app
- ace
- package.json
- public
- server.js
- start
    - app.js

- kernel.js
- routes.js

App datoteka sadrži kontrolore, modele, pretvarače. Ace datoteka sadrži specifične naredbe za projekt. Package.json sadrži informacije o verzijama Node.js npm paketa. Public datoteka služi za spremanje javnih podataka na server-strani i klijent-strani, te za skladištenje CSS i JavaScript datoteka. Server.js datoteka pokreće HTTP poslužitelja. Pristupna točka poslužitelja se definira unutar .env datoteke dok za pokretanje koda koristimo adonis servis –dev. Start/app.js - u ovoj datoteci se nalaze pružatelji, naredbe i aliasi (skraćeni oblik putanje). Start/kernel.js sadrži međuslojeve koji su korišteni unutar aplikacije i u start/routes.js su definirani HTTP rute. Također podržava rad s PostgreSQL, MySQL, SQLite3, MariaDb, Oracle, bazom podataka u datoteci s postavkama gdje postoji mogućnost odabira željene baze, podataka za internet aplikaciju te se u svakom trenutku može odabrati druga baza podataka [3].

2.3. MySQL

MySQL je relacijski sustav baza podataka. Baza podataka je organizirani skup podataka koji predstavlja prikupljanje, obradu i skladištenje podataka. Glavni zadatak baze podataka je da nam u svakom trenutku omogući brz i jednostavan pristup podatcima. Svaka baza podataka posjeduje niz tablica koje se nazivaju sheme. Sheme sadrže vrstu podataka koji su opisani u bazi podataka [4]. Podatci mogu biti različitog tipa:

- integer
- string
- boolean
- timestamp
- float
- date

U ovom diplomskom radu odabran je MySQL relacijski sustav baza podataka. Relacije omogućavaju da se uspostave odnosi između tablica. Tablica se sastoji od stupaca i redova. Stupci se nazivaju atributima. Postoji četiri vrste relacija:

- jedan prema jedan
- jedan prema više
- više prema jedan
- više prema više

Relacije pokazuju u kakvom su odnosnu tablice unutar baze podataka. Redak koji ima oznaku PK predstavlja nositelja tablice odnosno primarne ključeve ili jedinstvene identifikator tablice. Pomoću jedinstvenog identifikatora možemo povezivati međusobno tablice s funkcijama:

- inner join
- left join
- right join
- full join

Svaka od MySQL funkcija drugačije povezuje tablice. Inner join povezuje samo one podatke koji su navedeni u obje tablice. Left join povezuje sve podatke ljeve tablice s zajedničkim tablicama desne tablice, dok right join radi obratno. Full join povezuje sve podatke iz ljeve i desne tablice. Unutar tablice možemo definirati neograničeni broj zamjenskih ključeva koji imaju oznaku FK. Zamjenski ključevi se referenciraju na primarne ključeve drugih tablica [5].

2.4. HTML

HTML je skraćenica za HyperText Markup Language, što znači opisni jezik za izradu mrežne stranica. HTML je besplatni opisni jezik nastao 1991. Godine čiji je kreator Tim Berners-Lee. Postoji velika sličnost s jezikom Runoff koji je razvijen 1960-ih godina od strane Jerome H. Slatzer, Bob Morris i Doug Mcllroy [6]. Hipertekst dokument stvara se pomoću HTML opisnog jezika. HTML opisni jezik služi za prikazivanje i oblikovanje sadržaja. Također stvaraju se hiperveze unutar teksta. HTML je jednostavan za korištenje, te se zbog svoje jednostavnosti uživa veliku popularnost među programerima. Značajka koja je omogućila veliku popularnost među programerima je što je od početka HTML besplatan. Hipertekst dokumenti se mogu prikazati u svim modernim internet preglednicima dok HTML upućuje internet preglednik kako prikazati krajnjem korisniku hipertekst dokument. Zadaća HTML je prikazati u svim internet preglednicima jednako dokument neovisno u kojem operacijskom sustavu pokrećemo internet preglednik. HTML nije programski jezik. Korisnici HTML ne moraju posjedovati tehničko znanje kako bi znali koristit opisni jezik HTML. HTML ne može izvršiti jednostavne zadatke zbrajanja ili oduzimanja. HTML ne posjeduje najjednostavniju logiku, te služi za kreiranje i opisivanje hipertekstualnih dokumenata. Html datoteke su najobičniji tekstualni dokumenti kojima ekstanzija završava na:

- .html
- htm

Osnovni element svakog hipertekstualnog dokumenta je oznaka. Svaki element se sastoji od početne oznake i završne oznake dok je sadržaj elementa upisan između njih. HTML dokumenti su povezani u hijerarhijsku strukturu što omogućuje krajnjem korisniku doživljaj pregleda internet stranice [7].

2.5. CSS

CSS (engl. Cascading Style Sheets) skup je pravila koja govore mrežnom pregledniku kako prikazati određenu HTML oznaku. Prvenstveno je napravljen kako bi se odvojio dizajn mrežne stranice od sadržaja dokumenta. CSS3 je najnovija verzija CSS-a. Korištenjem CSS atributa, HTML elementi mogu biti pomaknuti unutar mrežne stranice te se može utjecati na njihovu visinu, širinu, boju i mnoštvo drugih efekata. Novost u CSS3-u je atribut „transform" pomoću kojeg se elementi mogu skalirati, pomicati bez uporabe pozicioniranja te rotirati. Do uvođenja CSS3-a animacije su bile moguće korištenjem JavaScript-a, a kasnije su animacije moguće i kada je JavaScript onemogućen. Glavni nedostatak vizualnih efekata u CSS3-u je što u različitim web-preglednicima treba iskoristiti prefikse ovisno o tome koji atribut se upotrebljava, pa je istu liniju koda potrebno upisati više puta. CSS3 možemo podijeliti u module:

- selektore
- kutije(engl. Box models)
- pozadina i granice
- tekstualni efekti
- 2D/3D transformacije
- animacije
- izgled s višestrukim stupcima
- korisničko sučelje

Transformacija je efekt koji utječe na element tako da mijenja njegov oblik, veličinu ili poziciju. CSS3 transformacije omogućuju pomicanje, skaliranje, okretanje i rastezanje elemenata [8].

2.6. JavaScript

JavaScript često skraćeno kao JS, je skriptni programski jezik koji se koristi za razvoj funkcionalnosti klijent strane i server strane. Jezik je razvijen od strane tvrtke Netscape, 1993 godine. JavaScript je odigrao važnu ulogu u populariziranju novo nastale World Wide Web. World Wide Web predstavlja cjelokupnu svjetsku mrežu. Prvotna ideja je bila da JavaScript sliči na Javu, ali se od ideje odustaje. Microsoft Internet Explorer od verzije 3.0 interpretiraju JavaScript naredbe

koje su zapisane unutar HTML dokumenta. Jedna od najčešćih grešaka o JavaScript jeziku je da predstavlja jednostavniju verzije Jave, programskog jezika razvijenog od strane Sun Microsystem. Oba jezika dijele jedno zajedničko ime Java. Sličnost između imena je marketinški potez. JavaScript prvobitno je nosio ime LiveScript i u zadnjim trenucima promijenjeno je u JavaScript. JavaScript i Java odličan su tim ako se kombiniraju jer oba jezika imaju različita svojstva. JavaScript ne posjeduje grafička ni mrežna svojstva dok, s druge strane, Java posjeduje navedena svojstva. JavaScript programski kod je dio hipertekstualnog dokumenta i navodi se između oznaka <SCRIPT> i </SCRIPT>. Oznaka <SCRIPT> predstavlja početak JavaScript programskog jezika dok oznaka </SCRIPT> predstavlja kraj JavaScript programskog jezika. Naredbe JavaScript programa navedene između ovih oznaka izvršavat će se prema redoslijedu pojavljivanja. <SCRIPT> naredbu možemo smjestiti u <HEAD> ili <BODY> hipertekstualnog dokumenta. Jedan hipertekstualni dokument može sadržavati više JavaScript parova te <SCRIPT> i </SCRIPT> oznaka. Ove skripte imaju redoslijed izvršavanja onako kako su navedene u hipertekstualnom dokumentu, ali ipak sve zajedno čini dio jednog JavaScript programa što znači da se funkcije i varijable međusobno dijele. JavaScript raspoznaje mala i velika slova. Varijabla diplomski_rad je drugačija od varijable diplomski_Rad. Prilikom definiranja varijabli ne smijemo započeti s brojem. Tri ključne riječi koje se koriste za definiranje varijabli su:

- var
- const
- let

Prilikom prve inicijalizacije podatka varijabli se dodijeli tip varijable [9].

3. RAZVOJ VLASTITOG RJEŠENJA

Čitavi diplomski rad je dogovoren u suradnji s mentorom te je osmišljeno kako će izgledati internet aplikacija. Prvotno se morala osmisliti cjelokupna shema baze podataka, a zatim je ona implementirana pomoću tehnologije MySQL relacijske baze podataka koja je opisana dalje u tekstu.

3.1. MySQL baza podataka

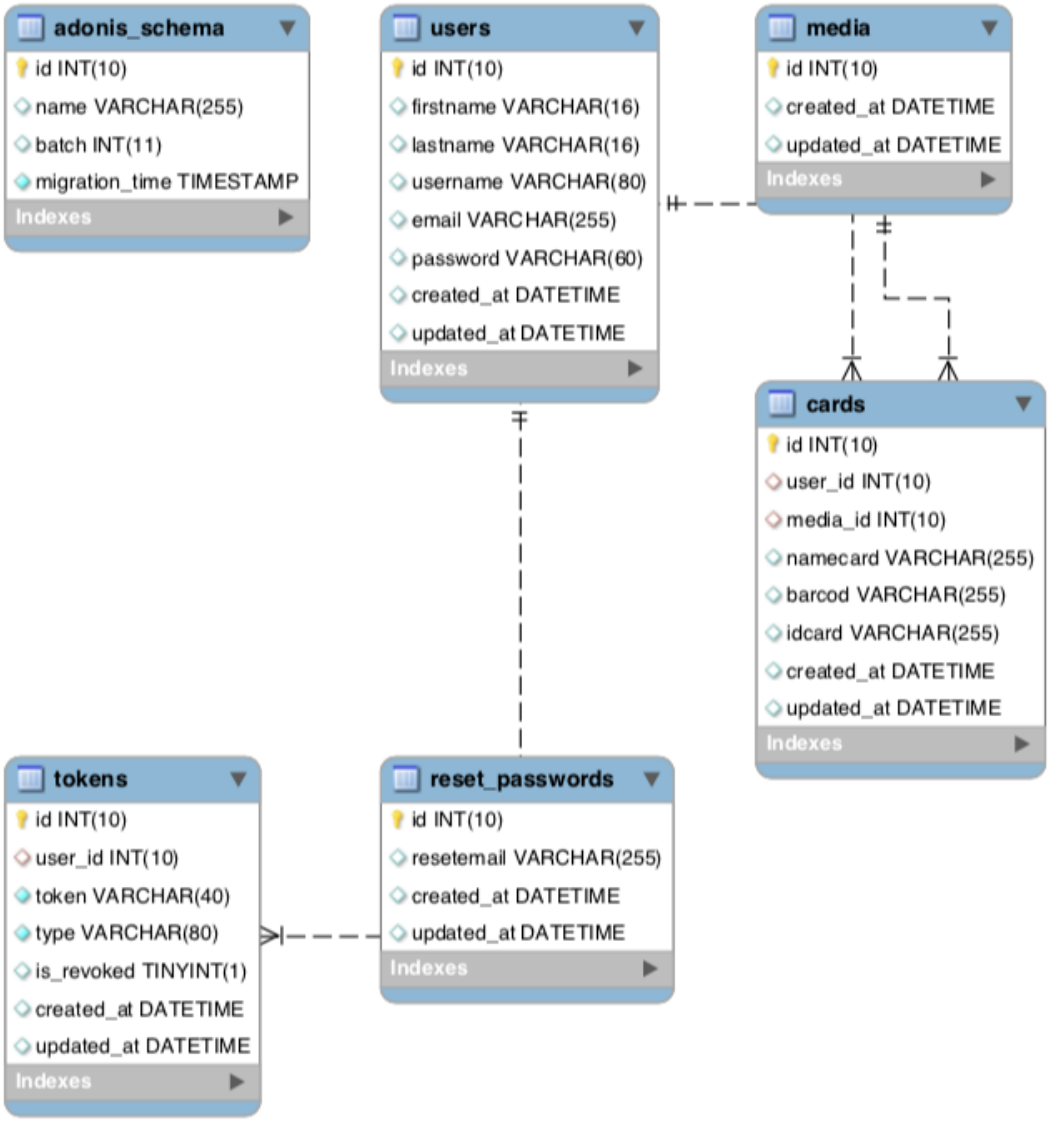
Shema baze podataka kreirana je u MySQL sustavu za upravljanje bazom podataka. Odabrana je iz razloga što je relacijskog tipka koji olakšava skladištenje i pretraživanje velikog broja podataka i predstavlja jezgru internet aplikacije za podršku kupovnih kartica. Shema baze podataka zamišljena je tako da uključuje šest relacijskih tablica koje redom nose naziv: user, media, token, card, reset_password, i user_cards. Tablica user (korisnik) sadrži informacije o korisniku: firstName, lastName, userName, email, password; svi atributi ove tablice su tipa string. Atributi email i username su unikatni, odnosno, samo jedan zapis u bazi podataka može se ponoviti. Atributi firstName i lastName su nulabilni tj. nisu obavezni pri kreiranju novog zapisa u tablici users, dok su atributi userName, email i password obavezni prilikom svakog zapisa podataka u tablicu users [10]. Tablica cards sadrži informacije o kartici koju posjeduje korisnik. Sadrži relaciju jedan prema više u odnosu na tablicu users što znači da jedan korisnik može posjedovati više kartica, ali da samo jedna kartica može pripadati jednom korisniku. Također sadrži relaciju jedan prema jedan u odnosnu na tablicu media. Tablica card sadrži atribute ID, user_id, media_id, nameCard, barCod, cardId. Atributi user_id i media_id su vanjski ključevi prema tablici users i tablici media. Atributi nameCard i cardId su obavezni prilikom svakog kreiranja novog zapisa unutar tablice cards, dok atribut barCode nije obvezan budući da neke kupovne kartice nemaju naveden barcode. Tablica media služi za skladištenje slika određene kartice te sadrži osnovne informacije o slici odnosno atribute: created_at, title, description, user_id, url, path, type, subtype i size. Također, sadrži relaciju jedan prema jedan u odnosnu na tablicu cards, tj. jedna kartica može posjedovati samo jednu željenu sliku, dok nam relacija jedan prema više usera govori da jedan korisnik može posjedovati više media zapisa. Atribut created_at je tipa timestamps. Timestamp je vremenska oznaka niza znakova ili kodiranih podataka koji

identificiraju kada se dogodio određeni događaj, obično dajući datum i vrijeme točno u sekundu. Podaci zapisani u timestampu su lako čitljivi jer se prikazuju u standardiziranom formatu [11]. Atribut title označava željeni naziv pod kojim se želi spremiti podatak koji smo uploadali na server te je on najčešće spoj trenutnog datuma i određenog zapisa jer ne želimo na serveru imati dva zapisa istog titla. Atribut description je opis prenesenog podatka na server koji je tipa text. User_id je vanjski ključ prema tablici users, ali on nije definiran u tablici kao vanjski ključ već mora postojati prilikom svakog unosa u tablicu media (Slika 3.1.). User_id u tablici media oponaša vanjski ključ zbog toga što je tablica media kreirana prije tablice user [12]. Atributi url i putanja su tipa string koji sadrže podatke kako pristupiti određenim podacima na server. Url željenog zapisa je mjesto na serveru gdje je pohranjen podatak, a putanja je skriveni atribut jer ne želimo otkriti korisniku gdje je na serveru njegov podatak spremljen. Atribut type je tipa enum - može poprimiti određene vrijednosti koje su u našem slučaju slika, video i drugi. Ako nije odabrana niti jedna vrijednost, ili željena vrijednost nije ispravno poslana, atribut će poprimiti vrijednost drugi. Atribut subtype je tipa string koji nam govori približni podtip podatka, dok je atribut size tipa integer te u njega server sam upisuje veličinu prenesnog podatka na server.

```
'use strict'

const Schema = use('Schema')

class MediaSchema extends Schema {
  up () {
    this.create('media', (table) => {
      table.increments()

      table.string('title', 60)
      table.text('description')

      table.integer('user_id').unsigned().notNullable()

      table.string('url')
      table.string('path')

      table.enum('type', ['image', 'video', 'other']).defaultTo('other')
      table.string('subtype', 10)
      table.integer('size').unsigned()
      table.timestamps()
    })
  }

  down () {
    this.drop('media')
  }
}

module.exports = MediaSchema
```

Slika 3.1. Shematski prikaz tablice Media

Tablica token sadrži atribute user_id, token, type i is_revoked. Sadrži relaciju jedan prema više u odnosnu na tablicu users. Atribut user_ id je vanjski ključ prema tablici users. Atribut token je string zapis tokena koji se automatski generira svaki put kada se korisnik prijavi na internet aplikaciju. Atribut token je tipa string. Atribut token je jedinstven te se ne može desiti da postoji više zapisa u tablici token koji imaju isti zapis kao atribut token. Atribut type je tipa string i pokazuje koje je vrste token; u našem slučaju uvijek će biti JWT vrste. Atribut is_revoked tipa je boolean. Pokazuje je li vrijeme trajanja tokena isteklo ili nije. Ova vrijednost ima unaprijed definiranu boolean vrijednost false. Tablica reset_password sadrži podatke o kodu koji korisnik dobije na e-mail te se kod dobiven na e-mail i kod u tablici reset_password moraju poklapati za određenog usera. Tablica user_cards glumi međutablicu između korisnika i kartica. User_cards tablica omogućuje korisniku dijeljenje vlastitih kartica s drugim korisnikom.



Slika 3.2. EER dijagram sustava

Iz slike 3.2. EER dijagrama sustava vidljivi su relacijski ključevi koji su postavljeni unutar MySQL baze podataka [13]. Najvažnija tablica u EER dijagram sustavu je tablica users koja sadrži sve bitne podatke u useru te ona predstavlja samo srce internet aplikacije. Vidljiva su svojstva i ograničenja koje baza podataka posjeduje. Zbog dobre organizacije vanjskih i primarnih ključeva, baza podataka ne posjeduje slabu točku, što je i vidljivo iz EER dijagrama sustava.

3.2. Server strana

Server strana predstavlja funkcionalnosti svih onih segmenata internet aplikacije koju klijent strana ne vidi. Tiče se poslovne logike koja se izvršava na serveru; ostvaruje komunikaciju s bazom podataka te omogućuje klijentu relativne podatke u stvarnom vremenu. Poznavanje server strane

zahtijeva od programera poznavanje širokog spektra tehnologija i spremnost na učenje i prilagođavanje zahtjevima tržišta. Za pisanje server strane internet aplikacije za upravljanje kupovnih kartica odabran je mrežni programski okvir AdonisJs i skriptni jezik JavaScript.

Prve prepreke u izradi server strane internet aplikacije bile se rute za login, registraciju, forgot password i logout. Ruta za registraciju sadrži validaciju koja provjerava točnost unesenih parametara [14]. Od korisnika se zahtijevaju parametri userName, email, password i password_confirmation. Kako bi se validacija parametra uspješno provela, korisnik mora obavezno upisati sve navedene parametre s tim da svaki parametar ima definirana drugačija pravila. Validacija se neće uspješno izvršiti ako korisnik prilikom upisa username i email parametra unese parametar koji već postoji u bazi podataka jer su username i email atributi jedinstveni u bazi podataka. Oba tipa parametra su string, dok kod email parametra se još dodatno izvršava sanitacija emaila i validacija. Provjerava se također je li email parametar odgovarajućeg email zapisa. Password parametar je obvezan za unos. Minimalni broj znakova koji korisnik mora upisati je šest dok maksimalno može upisati trideset znakova. Validator, koji provjerava ispravnost podataka, još u sebi sadrži regex. Regex dozvoljava unos znamenaka od 0 do 9, malih i velikih slova engleske abecede od A do Z, simbol povlaka, simbol crtice i obavezno na kraju parametra mora se nalaziti slovo ili znamenka [15]. Parametar password_confirmation je obvezan, ako je unesen parametar password. Parametar password_confirmation koristi se kako bi korisnik uspješno potvrdio svoju novu lozinku zbog mogućih pogrešaka korisnika te mora biti identičan parametru password.

```
let allParams = sanitize(request.only(['username','email','password','password_confirmation']), {
  email: 'normalize_email'
})

const rule = {
   username: 'required|string|max:255|unique:users',
   email: 'required|string|email|max:255|unique:users',
   password: 'required|min:6|max:30|regex:^[0-9a-zA-Z-_]+$',
   password_confirmation: 'required_if:password|min:6|max:30|same:password'
 }

 const validation = await validateAll(allParams, rule)

 if (validation.fails()) {
    session
      .withErrors(validation.messages())
      .flashExcept(['validationFails'])

    return response.redirect('back')
  }
```

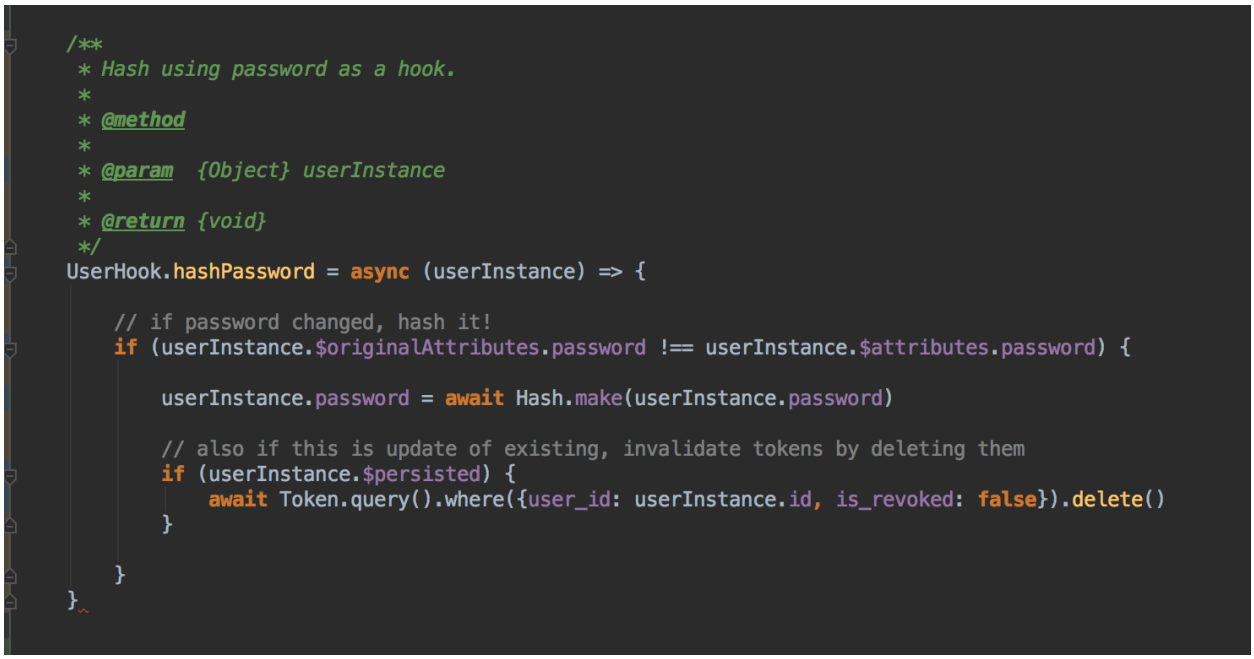
Slika 3.3. Validacija podataka prilikom registracije

Na slici 3.3. vidljivo je ako validacija ne prođe, korisnik će automatski biti vraćen na zadnju stranicu koju je uspješno posjetio. Nakon uspješne validacije parametara pristupamo User modelu preko kojeg kreiramo novog korisnika u tablici users. Razlog zbog kojeg se koriste modeli za upis podataka u bazu podataka je zbog jednostavne upotrebe i zbog brzog upita baza podataka o željenim stanjima. Modeli su usko vezani za shemu tablica u bazi podataka te su pohranjeni kao ES6 klasa u app/Models direktoriju gdje svaki model predstavlja tablicu baze podataka. Tako model User predstavlja tablicu users u bazi podataka. Modeli nam omogućavaju održavanje našeg koda čistim, daju mogućnost promjene podataka prilikom pohrane pomoću modelske funkcije i unaprijed postavljenje vrijednosti atributa u modelu. Također, omogućavaju formiranje podataka pomoću serijskih pretvarača, postavljanja datum formata, isključivanje/uključivanje timestampa po potrebi te skrivanja određenih atributa od krajnjeg korisnika (npr. skrivanje lozinke korisnika). Najkorisnija stvar koju nam omogućuju modeli, pisanje je SQL upita na bazu podataka. AdonisJs nam daje jedinstvenu sintaksu za interakciju s SQL bazama podataka koristeći JavaSript modele. Upiti u bazu podataka osiguravaju nam rad funkcije query-a bez obzira na odabranu SQL tehnologije, te nam omogućuje lako prebacivanje baze podataka s MySQL na PostgreSQL ili obratno [16].

```
await User.create ({
  username : allParams.username,
  email : allParams.email,
  password : allParams.password,
  firstname : allParams.firstname,
  lastname: allParams.lastname

})
```

Slika 3.4. Kreiranje novog korisnika

Na slici 3.4. vidi se dio koda u kojem se kreira novi korisnik internet aplikacije nakon uspješno provedene validacije parametara. Također, na samom modelu User definirali smo hook koji se izvršava prije kreiranja novog zapisa u bazu podataka. Hook služi za održavanje koda čistim, ali kada se radi s hookom, trebalo bi se paziti kako i gdje se koristi. Hook nam omogućava hashiranje parametra password. Hash pruža usluge šifriranja vrijednosti i šifriranja podataka. Vrijednosti odstupanja razlikuju se od enkripcije jer se hashirane vrijednosti ne mogu dešifrirati nakon hashiranja.



Slika 3.5. UserHook

Na slici 3.5. vidljiv je dio koda koji se koristi za brisanje JWT tokena kojima je vrijeme isteklo, pristupa se preko Token modela bazi podataka i brišu se svi tokeni koji odgovaraju trenutno logiranom useru i koji imaju boolean tipa is_revoked postavljen na false. Ruta za editiranje korisnikovih podataka zaštićena je s međusloljem getUser. Međusloljevi su skup funkcija koje se izvršavaju redom kako su poredane. Sekvencijalni lanac čini ga snažnim za preoblikovanje zahtijeva. Razlikujemo "global" i "named" međuslojeve. Imenovani međusloj objekt je parova ključa i vrijednosti te se može definirati na pojedinačne rute ili grupu ruta. Koristi se samo kada želimo izvršiti određene radnje na određenim rutama. Globalni međusloj izvršava se na svaki zahtjev rute koju korisnik zatraži te s globalnim međuslojem moramo oprezno raditi [17]. Međusloj getUser je imenovani međusloj koji dohvaća podatke o trenutno prijavljenom korisniku te ako korisnik nije prijavljen ili mu je isteklo vrijeme trajanja JWT tokena, ne može koristit rute koje zahtijevaju provjeru prijavljenog korisnika. Međusloj getUser koristan je jer nam omogućava bespotrebno pisanje koda u svakoj ruti za dohvaćanje trenutno prijavljenog korisnika te možemo u željenim rutama dohvatiti podatke o trenutnom korisniku koji je logiran. Ruta za editiranje korisnikovih podataka se sastoji od sanitacije koja definira željene tipove podataka i validacije koji je nužna za provjeru unesenih pravila za određene parametre. Korisniku je omogućeno editiranje entiteta lozinka i korisničko ime. Prilikom editiranja lozinke korisnik mora upisati parametar currentPassword te se parametar hashira i uspoređuje s zapisanom korisnikovom lozinkom u bazi podataka. Korisnik može promjeniti lozinku ako konstanta isSame poprimi vrijednost true tj. ako su currentPassword i korisnikova lozinka iste hashirane vrijednosti.

```
if(allParams.currentPassword){
  const isSame = await Hash.verify(allParams.currentPassword, user.password)

  if(isSame) user.merge({password: allParams.password})
}

await user.save()
```

Slika 3.6. Usporedba currentPassowrda i korisnikova passworda

Slika 3.6. pokazuje način spremanja korisnikovih promjena pomoću save () funkcije dok se await koristi kao barijera koja čeka na izvršenje određenog dijela programskog koda bez kojeg se određeni dio koda ne bi uspješno obavio tj. ne bi poprimio očekivane vrijednosti. Nakon uspješno editiranih podataka korisniku se vraća korisnički objekt koji sadrži stare i nove podatke kako bi se mogli prikazati na klijent strani internet aplikacije. Ruta logout se koristi za odjavu trenutnog korištenje odnosno za odbacivanje svih korisnikovih važećih JWT tokena. Korisnik se može uspješno odjaviti tek kada se uspješno provede promjena svih korisnikovih boolean zapisa is_revoked s false na true.

Kontroler Card sadrži rute za kreiranje nove kartice, dijeljenje kartice, editiranje kartice, prikaz svih korisnikovih kartica i brisanje željene kartice. Kao što smo prije naveli, tablica cards sadrži podatke: nameCard, barCode, cardId i user_id. Prilikom kreiranja novog zapisa kartice u bazi podataka potrebno je provesti validaciju podataka. Validiramo parametre nameCard i cardId. Parametar namecard je obvezan, tipa string je, minimalne dužine tri znaka te maksimalne dužine dvadeset znakova. Parametar cardId je isto obvezan i taj je parametar tipa integer. Sanitizacija parametra vrši se prilikom unosa da bi se osiguralo da na validaciju dođe parametar tipa intiger. Prilikom neuspješne validacije podataka korisnik će biti vraćen na korisničko sučelje kreiranja nove kartice, ali ako dođe do uspješne validacije podataka pristupamo prenošenju slike preko kreiranog mikro servisa. Slika je također obavezan parametar koju svaki korisnik mora odabrati prilikom podizanja na server. Ruta očekuje parametar tipa image, koji ima dozvoljene ekstenzije slike jpg, png i jpeg te maksimalno dozvoljene veličine 4mb (Slika 3.7.).

```
const file = request.file('file', {
  types: ['image'],
  allowedExtensions: ['jpg', 'png', 'jpeg'],
  size: '4mb'
})

if(!file) return response.badRequest('post.noFile')
```

Slika 3.7. Validacija slike

Nakon uspješno provedene validacije, konstantu file i zahtjev prosljeđujemo mikro servisu za prenošenje slike na server. Mikro servis provjerava jesu li poslani svi parametri te ako navedeni parametri nisu poslani vraća false vrijednost. Poslije toga pomoću funkcije move() koja prima dva parametra. Prvi parametar je odredišna datoteka u koju želimo premjestiti datoteku na serveru i naziv premještene datoteke na serveru. Nakon uspješnog premještanja datoteke na server pristupamo kreiranju zapisa u tablici media pomoću modela Media. Zapis sadrži informacije o korisniku koji je prenio datoteku na server, naziv datoteke, putanju do datoteke na serveru, tip datoteke, podtip datoteke i veličinu. Kreira se media konstanta koja se vraća poslije uspješno obavljenog kreiranja zapisa u bazi podataka. Pomoću card modela kreira se novi zapis u bazi podataka. Korisnikov ID dobiva se iz međusloja getUser, dok se iz vraćenog objekta media uspješno iščita ID prenesene datoteke na server. Korisnik nakon uspješnog kreiranja kartice vidi kreiranu karticu na listi svih kartica. Ruta showCard prikazuje sve korisnikove kartice koje su kreirane. Karticama se pristupa pomoću modela Card te se vršie upiti na bazu podataka gdje se, pomoću paginacije, vraća korisnikovih zadnjih deset kreiranih kartica. Paginaciju koristimo jer korisnik nema definirani limit koliko može imati kreiranih kartica te zbog toga ne znamo kakav rezultat očekujemo na upitu. Paginacija se sastoji od parametara:

- page – trenutna stranica na kojoj se nalazimo
- perPage – broj rezultata po stranici
- limit – željeni broj podataka po stranici
- total – ukupni broj zapisa u bazi podataka
- data – polje objekata odnosno zapisi iz baze podataka

Paginacija u sebi također sadrži Vanilla Serializers. AdonisJs pruža mogućnost kreiranja vlastitoga pretvarača, ali po definiranim postavkama koristimo Vanilla Serializers [18]. Pretvarači pružaju čistoću dohvaćenog koda svaki put kada preko modela napravi se upit u bazu podataka. Povratna vrijednost je uvijek instanca Vanilla Serializera te uvijek vraća JavaScript Object Notation izraz.

JSON izraz je lak za čitanje ljudima, služi ponajviše za prijenos podataka objekta koji se sastoji od parova atributa i polja tipa podataka. Predstavlja uobičajeni format podataka koji se koristi za asinkronu komunikaciju između server-strane i klijent-strane. Prilikom pisanja servisa mora se paziti u kakvom tipu se vraćaju podaci te se korisniku ne trebaju vratiti podaci koji su neupotrebljivi i zbog toga se moraju slijediti pravila za strukturiranje podataka. Vanilla Serializers rješava problem strukturiranja podataka na način:

- dodaje sve relacije unutar modela
- sve podatke sa strane dodaje u __meta__ objekt
- oblikuje rezultate paginacije

Ruta za dijeljenje kartice zahtjeva unos parametara ID kartice i ID korisnika s kojim se želi podijeliti karticu, dok se ID korisnika koji dijeli karticu dobije pomoću međusloja getUser. Prije nego li se podijeli kartica s korisnikom, mora se provjeriti postojanje kartice s željenim ID-em u bazi podataka i postojanje korisnika s kojim se kartica želi podijelitit. Kada se uspješno provjere svi podaci, kreira se zapis u tablici user_cards. User_cards tablica sadrži ID korisnika kojem se omogućuje pristup kartici te ID podijeljene kartice s korisnikom. Kako bi korisnik uspješno vidio koje su kartice s njim podijeljene, pomoću modela Card mora se napisati funkcija query u kojoj se traže sve kartice koje drugi korisnici imaju podijeljenje s korisnikom te se također provjerava da korisnik nije vlasnik navedenih kartica. U ovoj ruti također je korištena paginacija koja je sortirana po vremenu kreiranja zapisa od najnovijeg prema najstarijem.

,
```
let Cards = await Card
  .query()
  .whereHas('userCards',(b)=>{
    b.where(user_id,user.id)
  })
  .whereNot('user_id',user.id)
  .paginate(allParams.page,allParams.limit)
```

Slika 3.8. Dijeljen kartice s korisnikom

Iz slike 3.8. vidimo da funkcija paginate() prima parametar page koji se odnosi na željeni broj stranice i parametar limit, koji se odnosi na broj rezultata po stranici pretrage.

3.3. Korisničko sučelje

Korisničko sučelje internet aplikacije za podršku kupovnih karticama sastoji se od javne datoteke style.css koja sadrži CSS forme koje se koriste u internet aplikaciji. Datoteka style.css

sadrži informacije o klasama html, body, footer. Svaka CSS klasa sadrži svoj naziv. Najčešće kao naziv klase navodi se selektor klase koji se može odabrati; unutar vitičastih zagrada definira se kako će se određena klasa ponašati kada bude pozvana unutar programskog dijela aplikacije. Klase html, body i footer su generičke klase koje nisu vezane za određene HTML tagove i mogu se bezbroj puta pozvati unutar našeg koda. Definiraju se imenom kojem pridružujemo točku ispred imena klase. CSS klasa html sadrži svojstva position i min-height. Svojstvo position određuje mjesto pozicioniranja na klijent-strani odnosno to je krajnji rezultat kojeg korisnik vidi. Postoji pet različitih vrijednosti položaja:

- static
- relativ
- fixed
- absolute
- sticky

Vrijednost svojstva position u klasi html je relative, što znači da će prilikom poziva klase element biti postavljen relativno u odnosnu na normalni položaj. Sljedeće svojstvo klase je min-height. Svojstvo se može postaviti na automatsko što znači da internet preglednik automatski izračunava visinu; u našem slučaju svojstvo je postavljeno na 100%. Svojstvo min-height će se primijeniti samo ako je sadržaj manji od minimalne visine (Slika 3.9.).

Slika 3.9. CSS klasa html

CSS klasa body sadrži svojstva font-family, padding-top, padding-bottom i color. Font-family svojstvo određuje font koji će se koristit. Family-font svojstvo može sadržavati više fontova te ako internet preglednik ne prepozna prvi font, pokušava sa sljedećim fontom. Postoje dvije vrste fontova:

- family-font
- generic-family

Za internet aplikaciju odabran je family-font Lato. Sljedeće svojstvo je padding. Padding je prostor između sadržaja i granice. Također stvara dodatni prostor unutar elementa. Razlikujemo:

- padding-top
- padding-right
- padding-bottom
- padding-left

Svojstvo padding-top je postavljeno na 5rem dok je svojstvo padding-bottom postavljeno na 3rem. Jedincia rem znači "root em" te nam pomaže u postizanju uravnoteženog dizajna. Ta jedinica pomaže za izračun vrijednosti veličine slova na root elementu kada je navedena veličina fonta root elementa. Jedinica se odnosi na početnu vrijednost svojstva. Veličina 1 rem-a jednaka je velični slova HTML elementa koja za internet preglednike ima zadanu vrijednost od 16px [19]. Zadnje svojstvo u klasi je color. Color se u CSS-u može odrediti sljedećim metodama:

- hexadecimal colors
- RGB colors
- RHBA colors
- HSL colors
- HSLA colors

Svojstvo color je određeno metodom hexadecimal colors. Hexadecimal colors je specificiran s #RRGGBB gdje RR predstavlja crvenu boju, GG predstavlja zelenu boju i BB predstavlja plavu boju. Sve heksadecimalne vrijednosti moraju biti između 00 i FF. U našem je slučaju uzeta specifikaciju #5a5a5a [20]. CSS klasa footer sadrži svojstva position, bottom, width, height, line-height i background-color. Svojstvo position postavljeno je na vrijednost absolute. Svojstvo bottom služi za postavljanje donjeg ruba elementa na željenu vrijednost iznad donjeg ruba najbližeg roditeljskog elementa. Bottom svojstvo utječe na vertikalni položaj postavljenog elementa. Bottom svojstvo postavljeno je na vrijednost 0 [21]. Svojstvo height poprima vrijednost 2.75rem, line-height vrijednost od 2.75rem, a background-color postavljen je na heksadecimalnu vrijednost #f5f5f5 (Slika 3.10.).

Slika 3.10. CSS klasa footer

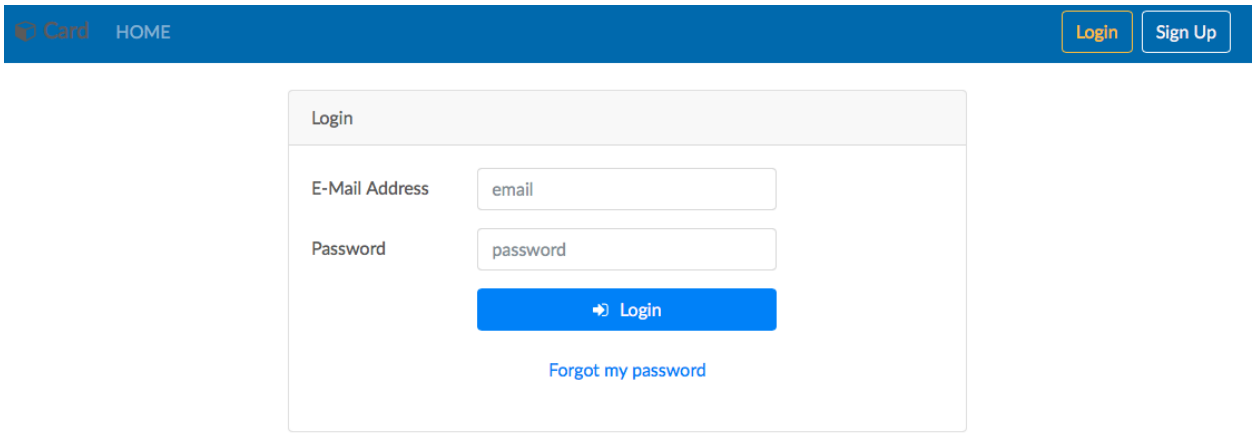
Forme koje krajnji korisnik vidi podijeljene su u predloške koje su edge podtipa podataka [22]. Sam edge podijeljen je na dva dijela:

- Compile time
- Runtime

Svi predlošci se izvrše samo jednom, a zatim se preuzimaju iz predmemorije. Zbog razvoja možemo isključiti predmemoriju kako bi se predlošci sastavljali svaki puta kada se napravi promjena. U edgu se proces podijeli na dva različita stupnja stvaranja:

- AST
- pretvaranje vrijednosti u izraze.

AST razumije kako funkcioniraju oznake u edgeu. Zbog toga ih je pametno ugnijezditi unutar tijela oznake kao objekte. Prvi zaslon kojeg korisnik vidi prilikom pokretanja localhost servera na portu 3333 (Slika 3.11.).



Slika 3.11. Početni zaslon

Početni zaslon sadrži dva predloška:

- login.edge
- footer.edge

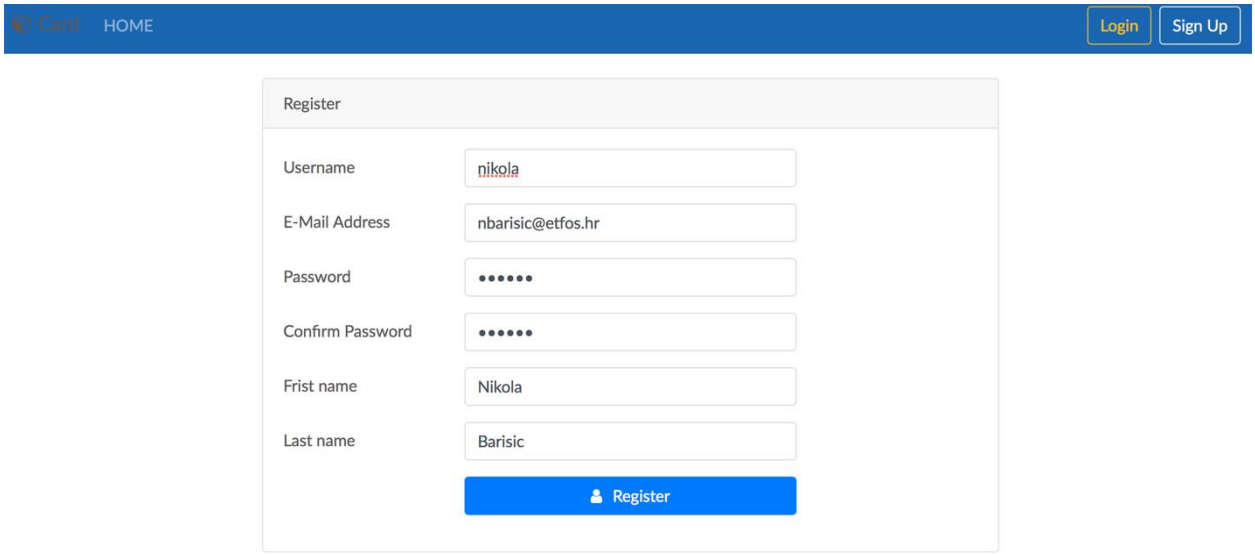
Na početnom zaslonu vidljive su funkcije „login”, „ sign up”, „forgot my password” i tipka „home”. Tipka home ima dva različita načina rada. Prvi način je da služi kao tipka za osvježavanje internet preglednika ako korisnik nije prijavljen. Drugi način je da vraća na prvu stranicu poslije prijave ako se korisnik već uspješno prijavio u internet aplikaciju (Slika 3.12.).

```
<form method="POST" action="{{ route('login.store') }}">
  {{ csrfField() }}
  <div class="form-group row">
    <label class="col-md-3 col-form-label" for="email">
      E-Mail Address
    </label>
    <div class="col-md-6">
      <input type="email" name="email" placeholder="email" id="email"
        class="form-control {{ elIf('is-invalid', getErrorFor('email'), hasErrorFor('email')) }}"
        value="{{ old('email', '') }}" required>
      <div class="invalid-feedback">{{ getErrorFor('email') }}</div>
    </div>
  </div>
```

Slika 3.12. Dio koda login.edge predloška

Tipka Login je post metoda koja na svoju akciju izvršava server stranu. Od klijenta se traži unos e-mail adrese i lozinke ako postoji. Parametar email je obavezan prilikom unosa, zaštićen je na korisničkom sučelju te mora biti odgovarajućeg e-mail zapisa. Korisnik ne može upisati string zapis bez „@” znaka [22]. Label također ispisuje HTML zapis e-mail adrese. U input tipu smo navodi se tip inputa, ime, placeholder i ID. Korisniku je nakon uspješno obavljene prijave vidljiv predložak „menu.edge”. Ako korisnik odabere značajku „Sign up” otvara se predložak za kreiranje

novog korisnika. Slika 3.13. prikazuje popunjeni predložak za kreiranje novog korisnika. Željeni korisnik će se spremiti u bazu podataka nakon uspješno provedene validacije podataka.



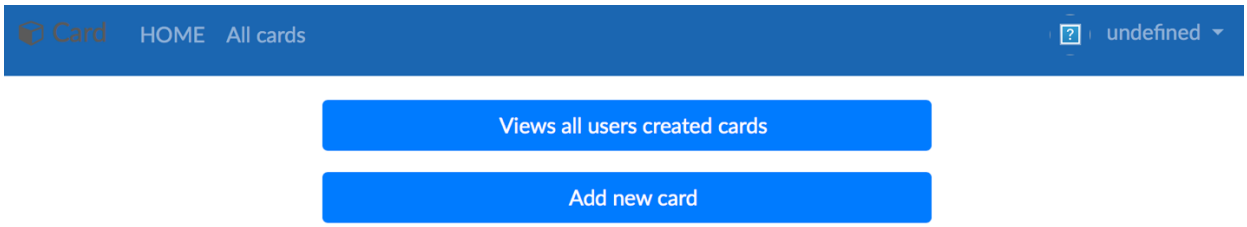
Slika 3.13. Registracija novog korisnika

Korisnik pritiskom na tipku „Register" prosljeđuje podatke server strani te kreira zapis u bazi podataka.

id	firstname	lastname	username	email	password	created_at	updated_at
2	Nikola	Barisic	nikola	nbarisic@etfos.hr	$2a$10$gi9X05G9QacnlGWrZjPDg.nh/r1zXXusWw7L...	2018-09-11 22:40:24	2018-09-11 22:40:24

Slika 3.14. Novi korisnik u bazi podataka

Slika 3.14. prikazuje novi uspješan zapis u bazi podataka. Vidljivo je da je parametar password zapisan s hashiranom vrijednošću. Sljedeći predložak kojeg vidi prijavljeni korisnik vidljiv je na slici 3.13.



Slika 3.13. Logirani korisnik predložak

Korisnik ima mogućnost odabira više opcija:

- Home
- All cards
- Settings
- Views all users created cards
- Add new card

Navigacijski bar ovisnosti o tome je li korisnik prijavljen ili nije dodaje značajku „All cards” koja vodi na predložak svih korisnikovih kartica. Značajka „Add new card” vodi korisnika na predložak za kreiranje nove kupovne kartice u bazi podataka.

```
<div class="form-group row">
  <label class="col-md-3 col-form-label" for="namecard">
    Card name
  </label>
  <div class="col-md-6">
    <input type="text" name="namecard" placeholder="namecard" id="namecard"
      class="form-control {{ elIf('is-invalid', getErrorFor('namecard'), hasErrorFor('namecard')) }} required"
      value="{{ old('namecard', '') }}">
    <div class="invalid-feedback">{{ getErrorFor('namecard') }}</div>
  </div>
</div>
```

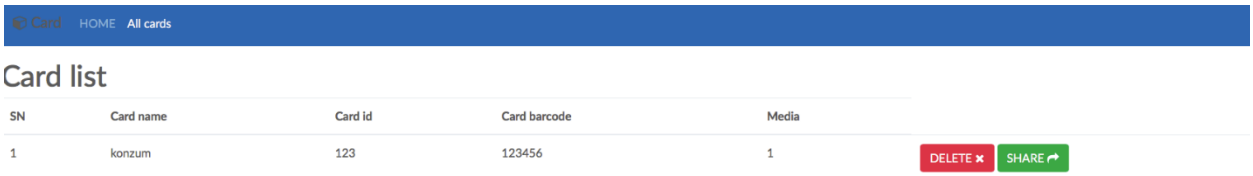
Slika 3.14. Upis imena kartice

Slika 3.14. prikazuje dio predloška za kreiranje nove kartice tj. formu za unos parametra namecard u bazu podataka. Input tip prima parametar tipa text, placeholder je namecard, ime parametra je namecard. Korisniku se prikazuje greška ako parametar nije važećeg zapisa te se nakon uspješne validacije parametara pristupa kreiranju novog zapisa u MySQL bazi podataka (Slika 3.15.).

id	user_id	media_id	namecard	barcod	idcard	created_at	updated_at
1	2	1	konzum	123456	123	2018-09-11 22:59:23	2018-09-11 22:59:23

Slika 3.15. Kreirana kartica u bazi podataka

Korisnik pritiskom na značajku „All cards” (Slika 3.16.) vidi listu svih kartica te ima mogućnost dijeljenja kartica s drugim korisnicima i brisanje postojeće kartice.



Slika 3.16.

4. ZAKLJUČAK

Uspješnim završetkom ovog rada stvorena je internet aplikacija za podršku kupovnih kartica. Internet aplikacija predstavlja sustav za vođenje, upravljanje, dodavanje i brisanje kupovnih kartica. Sadrži također ostale mogućnosti koje internet aplikacija mora posjedovati: registracija, login, logout, settings, refresh token logika itd. Također ukratko su opisane korištene tehnologije za izradu internet aplikacije. Zadatak ovog diplomskog rada bio je razvoj internet aplikacije za podršku kupovnih kartica te je detaljno opisan postupak kreiranja svih segmenata koje takva internet aplikacija mora sadržavati. Za bazu podataka odabrana je MySQL relacijska baza podataka zbog mogućnosti uspostave relacija između različitih tablica baze podataka. Server-strana je implementirana u AdonisJs, Node.js programskom okrviru. Svi zadaci server strane su uspješno odrađeni te se poslije uspješno kreirane server strane moglo pristupiti finalnom koraku - kreiranje klijent-strane. Klijent strana odrađena je također pomoću AdonisJs, Node.js programskog okvira uz kombinaciju s HTML-om, CSS-om i JavaSript-om. Kao rezultat uspješne implementacije klijent-strane, kreirana je funkcionalna internet aplikacija za podršku kupovnih kartica. Testiranjem server strane posebno uz pomoć alata Posteman utvrđen je uspješan rad server-strane. Prilikom kreiranja klijent-strane ponovno je provjerena server-strana uz pomoć internet preglednika te dolazimo do istih zaključaka o uspješnoj implementaciji. S time se diplomski rad smatra uspješno napravljenim.

LITERATURA

[1] WebStorm, https://www.jetbrains.com/webstorm/ (stranica posjećena 20. lipnja 2018.)

[2] AdonisJs, patreon https://www.patreon.com/adonisframework (stranica posjećena 20. Lipnja 2018.)

[3] AdonisJs, Node.js web programski okvir http://adonisjs.com (stranica posjećena: 22. lipnja 2018.)

[4] Wikipedia, MySQL (baza podataka) https://hr.wikipedia.org/wiki/MySQL (stranica posjećena: 20. lipnja 2018.)

[5] SQL Joins (baza podataka), https://www.w3schools.com/sql/sql_join.asp

(stranica posjećena: 20. lipnja 2018.)

[6] Wikipedia, HTML (opisni jezik) https://hr.wikipedia.org/wiki/HTML (stranica posjećena: 20. lipnja 2018.)

[7] HTML Elementi, https://www.w3schools.com/Html/html_elements.asp (stranica posjećena: 20. lipnja 2018.)

[8] ZEMRIS, CSS (stilski jezik) http://www.zemris.fer.hr/predmeti/irg/Zavrsni/13_Komar/sadrzaj_css.html (stranica posjećena: 20. lipnja 2018.)

[9] JavaScript, http://marjan.fesb.hr/~maja/internet/vjezba10.html (stranica posjećena: 20. lipnja 2018.)

[10] Knex.js, https://knexjs.org/#Chainable (stranica posjećena: 21. lipnja 2018.)

[11] Timestamp, http://momentjs.com/docs/#/parsing/unix-timestamp-milliseconds/ (stranica posjećena: 21. lipnja 2018.)

[12] Vanjski ključ, https://forum.adonisjs.com/t/cannot-add-foreign-key-constraint/1335/11(stranica posjećena: 21. lipnja 2018.)

[13] MySQL EER dijagram, https://dev.mysql.com/doc/workbench/en/wb-creating-eer-diagram.html (stranica posjećena: 21. lipnja 2018.)

[14] Validator, https://adonisjs.com/docs/4.1/validator (stranica posjećena: 21. lipnja 2018.)

[15] Regex, https://regexr.com (stranica posjećena: 22. lipnja 2018.)

[16] AdonisJs Modeli, https://adonisjs.com/docs/4.1/relationships (stranica posjećena: 24. lipnja 2018.)

[17] AdonisJs Međusloj, https://adonisjs.com/docs/4.1/middleware (stranica posjećena: 24. lipnja 2018.)

[18] AdonisJs Pagination, https://adonisjs.com/docs/4.1/lucid#_pagination (stranica posjećena: 24. lipnja 2018.)

[19] Jedinica REM u CSS-u, https://www.sitepoint.com/understanding-and-using-rem-units-in-css/ (stranica posjećena: 24. lipnja 2018.)

[20] Heksadecimalne vrijednosti boja, https://www.color-hex.com/color (stranica posjećena: 24. lipnja 2018.)

[21] Bottom svojstvo, https://www.w3schools.com/cssref/pr_pos_bottom.asp (stranica posjećena: 25. lipnja 2018.)

[22] Edge, Node.js template engine https://edge.adonisjs.com (stranica posjećena: 27. lipnja 2018.)

[23] Sanitizacija emaila, https://indicative.adonisjs.com/docs/normalizeemail (stranica posjećena: 29. lipnja 2018.)

SAŽETAK

Naslov: Internet aplikacija za podršku kupovnih kartica

U ovom diplomskom radu izrađena je internet aplikacija za podršku kupovnih kartica. Internet aplikacija je izrađena pomoću MySQL relacijske baze podataka te AdonisJs, Node.js programski okvir. Internet aplikacija sadrži značajke za: registraciju, login, logout, kreiranje nove kartice, brisanje kartice, dijeljenje kartice itd. Izradu aplikacije moguće je podijeliti na tri dijela. Prvi dio je kreiranje i osmišljanje MySQL relacijske baze podataka. Drugi dio je pisanje backend strane, te treci i zadnji dio je pisanje fronted stran. Nakon uspješne provedene tri faze diplomskog rada svi problemi su riješeni te se možemo koristit internet aplikacijom za podršku kupovnih kartica.

Ključne riječi: AdonisJs, Node.js, MySQL, edge, validacija, kupovne kartice

ABSTRACT

Title: Web application for shopping cards support

In this master's thesis a web application was made for shopping cards support. Web application was made with the help of a MySQL relational database and AdonisJs, a Node.js web framework. Web application contains features for: registration, login, logout, creating a new account, deleting a shopping card, sharing a shopping card, etc. Process of creating the application can be split intro three parts. First part is creating and devising a MySQL relational database. Second part is writing the backend code, and the third part is writing the frontend. After successfully conducting the three phases, all problems are solved and the web application can be used for shopping cards support.

Keywords: AdonisJs, Node.js, MySQL, edge, validation, shopping cards

ŽIVOTOPIS

Nikola Barišić rođen je 25. travnja u Đakovu. U Đakovu 2009. završava osnovnu školu „Josipa Antuna Ćolnića“. Iste godine upisuje srednju strukovnu školu u Đakovu koju završava 2013. godine. Tijekom cijelog srednjoškolskog obrazovanja ostvaruje odličan uspjeh. 2013. godine upisuje preddiplomski sveučilišni studij računarstva na Elektrotehničkom fakultetu u Osijeku. 2017. godine završava preddiplomski sveučilišni studij računarstva na Elektrotehničkom fakultetu u Osijeku te upisuje diplomski studij smjer DRA – Računalno inženjerstvo. Tijekom 2. godine diplomskog studija odrađuje praksu u Gauss development, nakon uspješno odrađene prakse zapošljava se u istoj firmi kao backend developer. Nikola Barišić tijekom studiranja aktivno se bavio vaterpolom i tajlandskim boksom.

_____

Nikola Barišić