Vrsta rada: Stručni rad

Izrada učinkovitog web rješenja za upravljanje slikama i njihovim verzijama

Ivan Zerec

Marina Bagić Babac

Sveučilište u Zagrebu Fakultet elektrotehnike i računarstva

Sažetak

U ovom radu opisan je proces izrade web servisa i pripadajuće klijentske aplikacije koji zajedno čine aplikaciju Flare koja pojednostavljuje proces upravljanja slikama pri razvoju web ili mobilnih aplikacija predstavljajući repozitorij slika. Kako bi korisnici mogli imati više različitih verzija iste slike u svojoj aplikaciji, omogućeno je verzioniranje originalnih slika. Verzije se kreiraju dodavanjem parametara na URL originalne slike. Novokreirana verzija slike sprema se na poslužitelj kako bi se optimizirao svaki idući zahtjev za istom verzijom. Klijentska aplikacija pruža grafičko sučelje za interakciju s web servisom putem HTTPS protokola.

Ključne riječi: web servis, verzioniranje slika, programski jezik Go

Primljeno: 20.5.2022.
Prihvaćeno: 31.7.2022.

DOI: 10.2478/crdj-2022-0008

Uvod

Pri razvoju web ili mobilnih aplikacija često se javlja potreba za repozitorijom slika i drugog medijskog sadržaja. Tim slikama je nerijetko potrebno manipulirati kako bi se prilagodile potrebama samih aplikacija. To može uključivati izrezivanje originalne slike na željene dimenzije uz mogućnost zadržavanja bitnog sadržaja, promjenu formata ili pak razine kvalitete slike.

U ovom radu prikazana je izrada web servisa i pripadajuće klijentske aplikacije koji predstavljaju rješenje za prethodno navedene zahtjeve. Ideja za razvoj ovog servisa nastala je kao potreba za korištenjem dinamički uređenih slika pri razvoju vlastitih web aplikacija.

U prvom poglavlju opisane su korištene tehnologije. Slijedi opis same aplikacije i njezinih funkcionalnosti. Treće poglavlje sadrži prikaz arhitekture sustava te objašnjenje svakog od slojeva arhitekture. U zadnjem poglavlju opisan je algoritam obrade slika prilikom kreiranja njihovih verzija te je prikazano korisničko sučelje klijentske aplikacije uz kratak opis.

Tehnologije i alati

Vue.js

Vue.js je JavaScriptni razvojni okvir (engl. *framework*) za izgradnju korisničkih sučelja pogodan za pisanje jednostraničnih aplikacija (engl. *Single-Page Applications*, skraćeno SPA). SPA je paradigma izrade web aplikacija prema kojoj se sva funkcionalnost web aplikacije ilistranice ugrađuje u jedinstveni web dokument. Glavna prednost takvog pristupa je poboljšano korisničko iskustvo jer korisnik ne mora učitavati svaku pojedinu stranicu koju posjeti na web sjedištu nego se sadržaj inicijalno učitane stranice mijenja po potrebi. Takav pristup omogućuje brže i tečnije navigiranje po web aplikaciji, čime ona više podsjeća na nativne računalne aplikacije.

Vue.js implementira, tzv. *Model-View-ViewModel* arhitekturu koja omogućuje reaktivno vezanje podataka i DOM-a (*Document Object Model*). DOM je model za prikaz i interakciju s objektima u HTML dokumentu. Omogućava jednoznačan i jednostavan pristup i rukovanje dijelovima HTML dokumenta.

Glavna ideja je pojednostaviti sinkronizaciju elemenata DOM-a s podacima. Svaka izmjena podataka automatski rezultira izmjenom sadržaja DOM-a bez dodatne potrebe za manipulacijom HTML-om. Vue.js postiže navedenu funkcionalost vezanjem dodatnih atributa na HTML elemente čime se stvara gore spomenuta veza (Kôd 1).

```
<!-- HTML -->
<div id="example-1">
  Hello {{ name }}!
</div>

// Model
var exampleData = {
  name: 'Vue.js'
}
// Vue instanca (ViewModel)
var exampleVM = new Vue({
  el: '#example-1',
  data: exampleData
}
```

Kôd 1 Vezanje podataka i DOM-a

Stvaranjem instance Vue koja predstavlja ViewModel vežemo podatke Modela sa HTML elementom identifikatora *example-1*. Sadržaj HTML-a koji predstavlja View automatski će se ažurirati ovisno o vrijednosti podataka u Modelu. Prilikom bilo kakve promjene vrijednosti imena u Modelu, izmjene će se uvijek odraziti i u samom HTML-u bez potrebe za dodatnom manipulacijom DOM-a.

Osim prethodno opisane veze, moguće je i dvosmjerno vezivanje (engl. *two-way data binding*) čime izmjene u HTML-u (View) automatski ažuriraju podatke u Modelu. To se postiže v-model Vue direktivnom. Primjer korištenja direktive i dvosmjernog vezivanja prikazan je u nastavku (Kôd 2).

```
<!-- HTML -->
<div id="example-1">
  <input
    type="text"
    v-model="name"
    value="{{ name }}"
  >
  <p>Hello {{ name }}!</p>
</div>

// Model
var exampleData = {
  name: 'Vue.js'
}
// Vue instanca (ViewModel)
var exampleVM = new Vue({
  el: '#example-1',
  data: exampleData
}
```

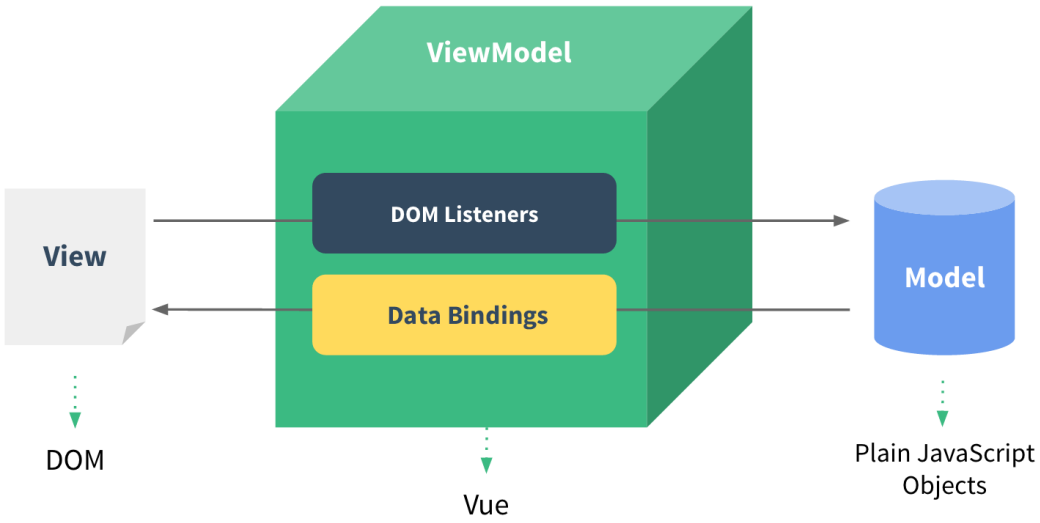
Kôd 2 Dvosmjerno vezivanje u Vue.js razvojnom okviru

Vrijednost upisana u input HTML element povezana je v-model direktivom s varijablom name u Modelu. Svakom promjenom vrijednosti inputa, ažurira se vrijednost podatka u Modelu što rezultira promjenom teskta u paragrafu HTML-a na

trenutnu vrijednost varijable name u Modelu. Prethodno opisana arhitektura prikazana je na slici 1.

Slika 1

Arhitektura Vue.js komponente



Izvor: (Vue, 2020)

Nuxt.js

Nuxt.js je razvojni okvir za razvoj korisničkih sučelja izgrađen u Vue.js-u. Prednost ovog razvojnog okvira, u odnosu na Vue.js, su funkcionalnosti poput generiranja sadržaja na poslužiteljskoj strani (engl. *server side rendering*), automatskog generiranja ruta, boljeg upravljanja meta tagovima, te bolje optimizacije web stranice za internet tražilice (engl. *search engine optimization*).

Kako bi bolje optimizirao web stranicu za internet tražilice, Nuxt.js koristi generiranje sadržaja na poslužiteljskoj strani. To znači da AJAX zahtjevima dohvaća podatke te iz Vue.js komponenti generira HTML tekst na poslužiteljskoj strani (Node.js). HTML tekst se šalje direktno u preglednik kada je svo dohvaćanje podataka obavljeno. Ovaj proces omogućuje nesmetano parsiranje DOM elemenata Google SEO parserom velikom brzinom čim je DOM učitan.

S druge strane, tipične jednostranične aplikacije napravljene u razvojnim okvirima poput Vue.js-a dohvaćaju podatke s poslužitelja nakon što je DOM učitan zbog čega SEO parser ne može parsirati sve DOM elemente jer nisu svi još prikazani.

Dodatna razlika je što Nuxt.js samostalno generira ruter na temelju strukture foldera dok je kod Vue.js-a to potrebno napraviti ručno. Automatski generirani ruter je jednostavan za korištenje jer je jedino potrebno napraviti direktorije i datoteke unutar njih, ali je s druge strane manje fleksibilan od ručno napisanog rutera.

Kada korisnik posjeti Nuxt.js aplikaciju ili klikne na neki od nuxt linkova na stranici, događa se sljedeće:

1. Kad korisnik inicijalno posjeti aplikaciju, ukoliko je nuxtServerInit akcija definirana u skladištu podataka, Nuxt.js će ju pozvati i ažurirati skladište podataka.

2. Slijedi izvođenje bilo kojeg posredničkog kôda (engl. *middleware*) za posjećenu stranicu.

3. Ukoliko je posjećena ruta dinamička i za nju postoji metoda validate(), ruta će se validirati.

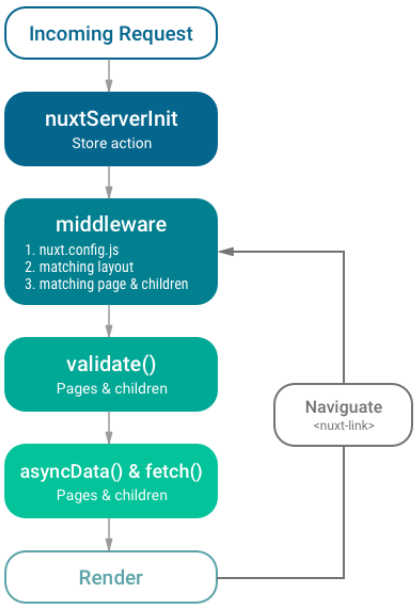
4. Nuxt.js poziva asyncData() i fetch() metode kako bi učitao podatke prije prikazivanja stranice. Metoda asyncData() služi za dohvaćanje podataka sa poslužitelja, dok se metoda fetch() koristi kako bi se napunilo skladište podataka prije prikaza stranice.

5. U posljednjem koraku se prikazuje stranica sa svim pripadajućim podacima.

Gore opisan proces prikazan je na slici 2.

Slika 2

Proces obrade zahtjeva Nuxt.js-om



Izvor: *(Omole, 2019)*

Vuetify

Vuetify je najpopularnija knjižnica komponenti za Vue.js (Vuetify, 2019) koja pruža korisnicima sve što je potrebno za izgradnju web aplikacija po Material dizajn specifikaciji (Material, 2020). Odlikuje se velikim brojem komponenti i drugih alata koji omogućuju brz razvoj i mnogo naprednih funkcionalnosti bez potrebe za njihovim samostalnim kreiranjem. Neke od drugih prednosti ove knjižnice su bogat ekosustav, česte nadogradnje, dugotrajna podrška, velik broj aktivnih korisnika, te usluga podrške poslovnim korisnicima.

Go

Programski jezik Go je projekt koji je nastao kao odgovor na česte probleme prilikom razvoja infrastrukture softvera u Googleu (Golang, 2020). Situacija u računarstvu u današnje vrijeme značajno se razlikuje od sredine u kojoj su kreirani jezici kao što su C, C++ i Python. Problemi poput višejezgrenih procesora, mrežnih sustava, velikih računalnih grozdova te kompleksnih poslužiteljskih aplikacija nisu uzeti u obzir prilikom dizajna samih jezika već su rješavani kako su pristizali. Nit vodilja pri dizajniranju ovog programskog jezika bila je ponuditi moderan programski jezik koji će gore spomenute probleme uzeti u obzir pri samom dizajnu. Ovaj projekt otvorenog kôda postao je javno dostupan 2009. i od tada su brojni ljudi iz zajednice pridonijeli idejama, raspravama i kôdom.

Go je strogo statički tipiziran kompajlerski jezik štogarantira provjeru tipova varijabli prije runtimea, odnosno prilikom samog kompajliranja. Također, to znači da jezik ne dopušta implicitnu promjenu tipa varijabli što je prednost jer otežava mogućnost grešaka i olakšava proces debugiranja.

Jedna od bitnih karatkeristika ovog jezika je jednostavno pokretanje konkurentnih aktivnosti uz minimalnu kompleksnost kôda. U programskom jeziku Go kada je u pitanju konkurentnost, osnovni pojam predstavljaju go-rutine (engl. *goroutines*). Go-rutine su karakteristične samo za jezik Go. One zahtjevaju manje resursa od standardnih dretvi te im je vrijeme pokretanja kraće. To znači da je moguće imati više go-rutina nego standardnih dretvi koje podržava operacijski sustav. U programu može biti samo jedna dretva s tisuću go-rutina. No ukoliko bilo koja go-rutina blokira dretvu, npr. zbog čekanja odgovora od korisnika, onda se pokreće nova dretva i preostale go-rutine se prebacuju u novu dretvu. Sve je to apstrahirano od programera kako bi mu se omogućilo jednostavno korištenje konkurentnosti.

Kad se program pokrene, jedina go-rutina koja postoji je glavna go-rutina koja poziva funkciju main. Nove go-rutine kreiraju se upotrebom ključne riječi go i navođenjem funkcije koja će se izvoditi konkurentno u novoj go-rutini.

Da bismo učinkovito radili s višedretvenim softverom, potreban nam je neki oblik sinkronizacije. Go nudi odličan koncept kanala (engl. channel) za potrebe dijeljenja

podataka između go-rutina, kao i mogućnost korištenja mutexa ukoliko podatke nije potrebno dijeliti, ali i ostale primitive za sinkronizaciju.

Budući da je Go orijentiran na višedretvenost, nudi brojne alate za podršku razvoja takvih aplikacija. Primjer jednog takvog alata je alat za detekciju utrke za resursima (Race detector, 2020). Utrka za resursima spada u najkompleksnije probleme pri debugiranju višedretvenih aplikacija. Detektor utrke za resursima koji je razvijen za Go aplikacije smatra se jednim od najkvalitetnijih u struci.

Opis aplikacije Flare

Aplikacija Flare sastoji se od web servisa za upravljanje i verzioniranje slika te od front-end aplikacije koja predstavlja korisničko sučelje.

Web servis

Web servis je centralni dio ove aplikacije koji pojednostavljuje upravljanje slikama pri razvoju web ili mobilnih aplikacija predstavljajući repozitorij slika.

Servis je napravljen kao aplikacijsko programsko sučelje (engl. *Application Programming Interface*, skraćeno API) koje slijedi REST (engl. *Representational State Transfer*) načela. REST predstavlja arhitekturalni stil koji definira način komunikacije između klijenta i poslužitelja prilikom korištenja mrežnih resursa pomoću HTTP protokola. Jednostavnije rečeno, resurs je svaki objekt kojem se može pristupiti korištenjem hiperveze. Svaki resurs ima uniformni identifikator resursa (skraćeno URI).

U aplikaciji Flare postoje dva tipa resursa, direktorij i slika. Za pristup tim resursima koriste se standardne HTTP metode kao što su GET, POST, PUT i DELETE. Razmjena podataka sa API-jem se odvija u JSON formatu. Za korištenje svih API krajnjih točaka, osim za dohvaćanje slika, potrebna je autorizacija JWT tokenom. JWT tokeni su otvorena, industrijski standardizirana metoda za zaštićenu razmjenu informacija između dva sudionika (Jones et al., 2020). U JWT token pohranjuje se korisničko ime koje služi za identifikaciju korisnika prilikom njegovog pristupa API-u.

Definirane su sljedeće API krajnje točke:

- GET /api/folders

popis postojećih direktorija u direktoriju navedenom kao parametar

- POST /api/folders

kreiranje novog direktorija

- DELETE /api/folders/:name

brisanje postojećeg direktorija i njegovog sadržaja

- GET /api/images

popis postojećih slika u direktoriju navedenom kao parametar

- POST /api/images

učitavanje nove slike

- PUT /api/images/:id

uređivanje informacija o postojećoj slici

- DELETE /api/images/:id

brisanje postojeće slike

- GET /:client/images/:id

dohvaćanje postojeće slike

API krajnje točke, definirane nad resursom direktorij, predstavljaju standardne operacije poput pregleda postojećih direktorija, dodavanje novog ili brisanje postojećeg direktorija.

Pri učitavanju slika obvezan parametar je slika koja se šalje šifrirana u base64 formatu (Josefsson, 2020). Podržani formati slike su jpeg i png. Pomoću opcionalnih parametara može se specificirati direktorij u koji je sliku potrebno učitati te tagovi u svrhu bolje organizacije slika. Ukoliko direktorij nije naveden u zahtjevu, slika će biti pohranjena u korijenski direktorij. U odgovoru servisa nalaze se sljedeće informacije o učitanoj slici:

- javni identifikator
- originalno ime slike
- visina
- širina
- format
- direktorij u koji je slika smještena
- tagovi
- vrijeme kreiranja slike
- tri dominantne boje u slici
- veličina slike u bajtovima
- URL do slike
- popis verzija slike

Primjer odgovora na zahtjev nalazi se u nastavku (Slika 3).

Slika 3

Odgovor servisa na učitavanje slike


```
{
  "public_id": "da01d75aa1b842a9ce070f04792d81c5",
  "original_name": "slika1.jpg",
  "width": 200,
  "height": 200,
  "format": "png",
  "folder": "/",
  "tags": [],
  "created_at": "2020-06-05T17:40:53.581056024+02:00",
  "dominant_colors": [
    "F9F9F9",
    "080808",
    "5F5F5F"
  ],
  "bytes": 408,
  "sef": "http://localhost:8000/korisnik1/image/da01d75aa1b842a9ce070f04792d81c5.png",
  "transformations": []
}
```


Kako bi se pristupilo učitanoj slici, potrebno je posjetiti URL naveden u odgovoru zahtjeva. U prethodno spomenutom primjeru taj URL glasi: http://loaclhost:8000/korisnik1/image/da01d75a1b842a9ce080f04792d81c5.png

Glavna ideja ovog servisa je da se korisnicima omogući dinamičko kreiranje verzija prethodno učtane slike dodavši željene atribute slike u URL kao parametre.

Podržani atributi prikazani su u tablici u nastavku.

Tablica 1

Podržane transformacije nad slikama

ime	**URL parametar**	**primjer vrijednosti**	**opis**
širina	w	400 0.7	- širina verzionirane slike - može biti zadana egzaktna brojka u pikselima ili postotak širine u odnosu na originalnu sliku
visina	h	400 0.7	- visina verzionirane slike - može biti zadana egzaktna brojka u pikselima ili postotak visine u odnosu na originalnu sliku

izrez	c	scale fit fill pad crop	- tip izreza slike koji određuje način transformacije slike u zadanu širinu i visinu - scale - zadržavanje cijelog sadržaja slike bez očuvanja razmjera proporcija - fit - zadržan cijeli sadržaj slike i očuvan razmjer proporcija; slika je maksimalne veličine unutar zadanih parametara za visinu i širinu - fill – skaliranje slike da se popuni zadana visina i širina, višak se izrezuje - pad – zadržan cijeli sadržaj slike i očuvan razmjer proporcija te je dodan *padding* gdje je potrebno - crop – izrezana slika na zadanu visinu i širinu
razmjer proporcija	ar	1.5 16:9	- izrezivanje slike prema zadanom razmjeru proporcija - može se zadati kao decimalan broj ili omjer
pozadinska boja	b	9e9e9e	- ukoliko je kao tip izreza odabran *pad*, može se zadati boja popune kao hex vrijednost
gravitacija	g	lt l lb rt r rb t b c face faces	- ukoliko je odabran tip izreza zbog kojeg nije zadržan cijeli sadržaj slike, može se definirati dio slike koji se želi zadržati - opcije su svi kutevi i stranice slike, centar slike, gravitiranje licu ili licima na slici
format	f	jpg png auto	- format transformirane slike - ukoliko je format postavljen u *auto,* slika će biti formatirana ovisno o pregledniku korisnika, u svrhu boljih performansi

kvaliteta	q	80	- postotak kvalitete transformirane slike
radijus	r	20	- radijus kuteva slike u pikselima

Primjer kreiranja verzije slike bilo bi dohvaćanje originalne slike izrezane na dimenzije 720x480, gravitirajući prema gornjem desnom dijelu slike s radijusom kuteva od 50 piksela i s promjenom formata iz jpg u png. Na URL originalne slike potrebno je na kraj dodati sljedeće parametre: ?w=720&h=480&c=crop&g=rt&r=50&f=png. Ukoliko ime ili vrijednost nekog parametra nije ispravna, korisnik dobiva odgovarajuću povratnu poruku. Originalna slika (Slika 4) i verzionirana slika (Slika 5) prikazane su u nastavku.

Slika 4

Originalna slika

Slika 5

Verzionirana slika

Jednom dohvaćena verzija slike bit će pohranjena na poslužitelju u svrhu bržeg odgovora prilikom idućih zahtjeva za istom verzijom.

Kako bi se optimizirao proces prvog dohvaćanja neke verzije slike, pri učitavanju slike mogu se definirati verzije koje će biti asinkrono napravljene.

Postojeće slike i verzije moguće je obrisati direktno putem API-ja ili koristeći administratorsko sučelje.

Klijentska aplikacija

Osim direktnog konzumiranja API-ja, korisnik može pristupiti korisničkom sučelju putem front-end aplikacije. Aplikacija podržava samo registrirane korisnike. Cijeli opseg programske podrške može se koristiti samo kada korisnik napravi svoj račun.

Nakon provjere autentičnosti identiteta i uspješne prijave, korisnik je preusmjeren na glavni pogled koji predstavlja korijenski direktorij. U ovom pogledu korisnik vidi sadržaj trenutnog direktorija koji uključuje direktorije i slike. Klikom na bilo koji direktorij, prikazuje se njegov sadržaj, što korisniku simulira kretanje po direktorijima kao i u datotečnom sustavu samog operacijskog sustava. U svakom direktoriju korisnik može dodati novi direktorij ili učitati novu sliku. Korisnik također može pretraživati slike po imenima i po unaprijed zadanim tagovima.

Nad svakom slikom korisnik može izvoditi sljedeće akcije:

- brisanje postojećih verzija
- dodavanje novih verzija
- kopiranje URL-a slike u međuspremnik
- brisanje slike

Prilikom brisanja slike brišu se i sve pripadajuće verzije.

U pogledu profila korisnika prikazane su osnovne informacije o korisničkom računu. Osim tih podataka, korisnik može vidjeti podatke o korištenju servisa poput broja učitanih slika, broja napravljenih verzija te ukupno zauzeće memorijskog prostora.

Arhitektura sustava

Arhitektura sustava je troslojna. Sastoji se od sljedećih cjelina:

- klijentska aplikacija pisana u Vue.js razvojnom okviru
- poslužitelj pisan u programskom jeziku Go
- MongoDB baza podataka

Klijentska aplikacija

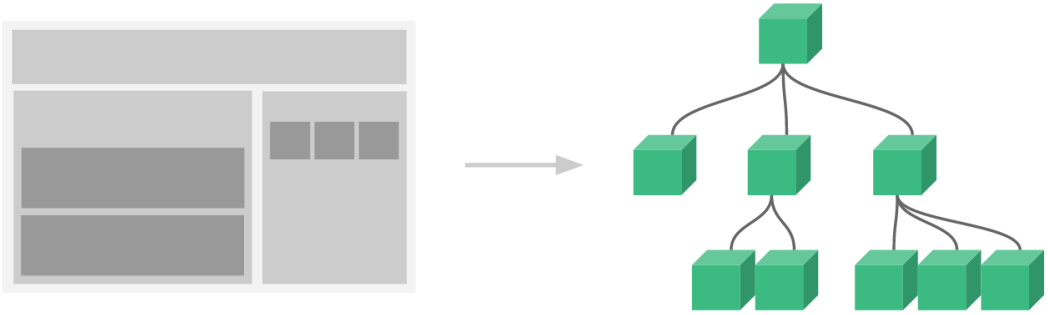
Klijentska aplikacija omogućuje korisniku interakciju s aplikacijom preko grafičkog sučelja. Pri razvoju takvih sučelja dostupni su brojni razvojni okviri. Njihova je namjena omogućiti brži razvoj nudeći skup često korištenih funkcionalnosti. Programeri ne moraju sami implementirati sve funkcionalne zahtjeve već mogu koristiti rješenja razvojnog okvira. To za posljedicu ima manje napisanog kôda te manju potrebu za testiranjem jer je ispravnost funkcionalnosti razvojnog okvira već detaljno ispitana.

Razvojni okvir korišten u ovom radu je Vue.js. Osim prethodno spomenutih prednosti, prednost korištenja takvog razvojnog okvira je i arhitektura ponovno iskoristivih komponenti. Komponentama se smatraju sučelja i neovisni blokovi programske logike koji se mogu koristiti na više mjesta u web aplikaciji. Svaka komponenta prima

samo potrebne podatke te implementira samo vlastitu funkcionalnost. Komponente čine hierarhijsku strukturu (Slika 6).

Slika 6

Struktura komponenata razvojnog okvira



Izvor: (Vue.js components, 2020)

Poslužitelj

Pri razvoju poslužitelja korišten je programski jezik Go. Poslužitelj je realiziran kao aplikacijsko programsko sučelje koje komunikaciju s korisnicima i klijentskom aplikacijom obavlja preko krajnjih točaka koje slijede REST načela (REST, 2020).

Glavni principi REST-a su sljedeći:

- klijent - poslužitelj komunikacija

odvajanje korisničkog sučelja od obrade podataka

- bez stanja (engl. stateless)

svaki zahtjev klijenta prema poslužitelju mora sadržavati sve potrebne informacije kako bi poslužitelj razumio zahtjev, te kako se ne bi mogle zlorabiti nikakve informacije o klijentu budući da su one pohranjene kod samog klijenta

- moguće korištenje priručne memorije (engl. cache)

o u odgovoru na zahtjev potrebno je naznačiti mogu li se podaci vraćeni u odgovoru spremiti u priručnu memoriju i koristiti kao odgovor pri ponovljenim zahtjevima za istim resursom

- uniformno sučelje
- slojevit sustav

slojevitost omogućuje arhitekturu koja je sastavljena od hijerarhijskih slojeva ograničavajući ponašanje komponenti na način da svaka komponenta ne može „vidjeti“ ništa osim susjednog sloja s kojim je u interakciji

- kôd na zahtjev (opcionalno)

REST omogućuje proširenje funkcionalnosti klijenta preuzimanjem i izvođenjem kôda u formi apleta ili skripti

Klijent i poslužitelj komuniciraju tako da klijent pošalje zahtjev poslužitelju koji taj zahtjev obradi te odgovor šalje nazad klijentu. Klijent-poslužitelj stil arhitekture tipičan je za web komunikaciju jer odjeljuje problematiku na dvije strane. Klijentska strana nije ovisna o načinu obrade i pohrane informacija na poslužitelju dok god se ne mijenja način pristupa tim resursima. Isto tako, poslužitelj je neovisan o implementaciji korisničkog sučelja i o trenutnom stanju klijenta (engl. stateless).

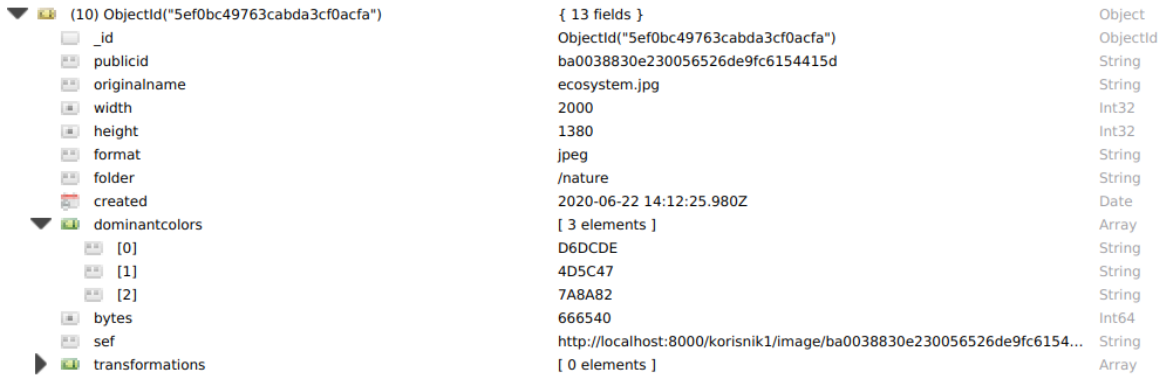
Baza podataka

Korištena baza podataka je nerelacijska. Jedno od glavnih obilježja koje odvaja relacijske od nerelacijskih baza je način na koji strukturiraju podatke. Relacijske baze ili SQL baze podataka spremaju podatke na strukturiran i unaprijed definiran način. Prije nego što se počnu unositi podaci u takvu bazu, shema mora biti jasno definirana. Kod nerelacijskih (NoSQL) baza to nije slučaj. Postoji nekoliko vrsta NoSQL baza podataka. Za izradu aplikacije Flaire korištena je MongoDB baza podataka koja se temelji na dokumentima (MongoDB, 2020). Osnovni element ovog modela podataka predstavlja dokument: uređeni skup ključeva s pridruženim vrijednostima. Dokumenti se smještavaju u kolekcije. Za razliku od relacijskih baza podataka, ovakva baza dopušta dinamičke sheme, što znači da svaki dokument u nekoj kolekciji može imati drugačiju strukturu.

U bazi podataka aplikacije Flare postoji kolekcija pod nazivom „korisnici“ koja sadrži dokumente sa podacima o korisnicima. Osim kolekcije korisnika, pri registraciji svakog korisnika u sustav, kreira se nova kolekcija koja služi za spremanje dokumenata koji će sadržavati podatke o korisnikovim učitanim slikama.Primjer izgleda jednog dokumenta koji predstavlja podatke o uploadanoj slici prikazan je u nastavku (Slika 7).

Slika 7

Primjer dokumenta u NoSQL bazi



Implementacija sustava

Algoritam verzioniranja slike

Svaka slika učitana na poslužitelj ima svoj URL preko kojeg joj se može pristupiti. Ukoliko korisnik želi dohvatiti određenu verziju slike, to čini dodavanjem parametara u URL originalne slike. Pri svakom prvom dohvaćanju verzionirane slike, ona se sprema na poslužitelj radi bržeg dohvaćanja pri idućim zahtjevima za istom verzijom.

Kada poslužitelj dobije zahtjev za slikom s postavljenim URL parametrima, od parametara i njihovih vrijednosti kreira se objekt. Kako bi se provjerila ispravnost svih poslanih parametara, objekt se validira te, ukoliko postoje greške, proces kreiranja verzije se prekida i korisnika se obavještava o greškama u upitu. Ako je objekt ispravan, kreira se MD5 hash od njegove tekstualne reprezentacije i imena originalne slike.. MD5 hash je algoritam koji se koristi za računanje sažetka poruke; stvara 128 bitni sažetak ulaznog podatka proizvoljne duljine. Dobiveni hash koristi se kao ime verzionirane slike. Prije kreiranja nove verzije provjerava se postoji li slika s istim imenom na disku. U slučaju da postoji, ista se vraća kao odgovor. Hash se generira iz objekta, a ne direktno iz URL parametara kako drukčiji poredak parametara ne bi prouzročio kreiranje nove verzije ukoliko ona već postoji. Verzija slike se kreira obavljajući niz operacija nad originalnom slikom.

Jedna takva operacija je automatski odabir formata slike. Ukoliko je korisnik za format slike odabrao vrijednost auto, provjerava se s kojeg preglednika je upućen zahtjev te se potom odabire format slike koji je optimalan za taj preglednik.Nakon uspješno kreirane verzije, podaci o njoj se spremaju u bazu.

Kako bi se optimizirao proces prvog dohvaćaja neke verzije, moguće je definirati željene verzije pri učitavanju slike. Kreiranje takvih verzija obavlja se asinkrono nakon učitavanja slike u zasebnoj go-rutini. Ta go-rutina za svaku definiranu verziju kreira novu go-rutinu u kojoj će nova verzija biti kreirana. Po završetku kreiranja verzija, informacije o njima pohranjuju se u bazu podataka.

Prepoznavanje lica

Poseban tip operacije je izrezivanje slike na način da se gravitira licu ili licima na slici.

Ukoliko je korisnik odabrao gravitiranje licu, u slučaju da na slici postoji više lica, gravitirat će se najvećem. Ako je pak odabrano gravitiranje licima, uzima se aritmetička sredina točaka od kojih svaka predstavlja centar pojedinog lica.

Za prepoznavanje lica na slici koristi se Go knjižnica zvana Pigo koja je temeljena na radu u čijoj je kreaciji sudjelovao FER (Markuš et al., 2014). Rad opisuje detekciju objekata usporedbom intenziteta piksela koristeći stabla odluke. Odabrana knjižnica ne ovisi o drugim vanjskim paketima te je pisana u programskom jeziku Go zbog čega se odlikuje brzinom.

Za ispunjenje iste zadaće kao opcija postoji i paket GoCV. Nedostatak tog paketa je što ovisi o knjižnici za računalni vid OpenCV koju je potrebno instalirati na samom operacijskom sustavu (OpenCV, 2020).

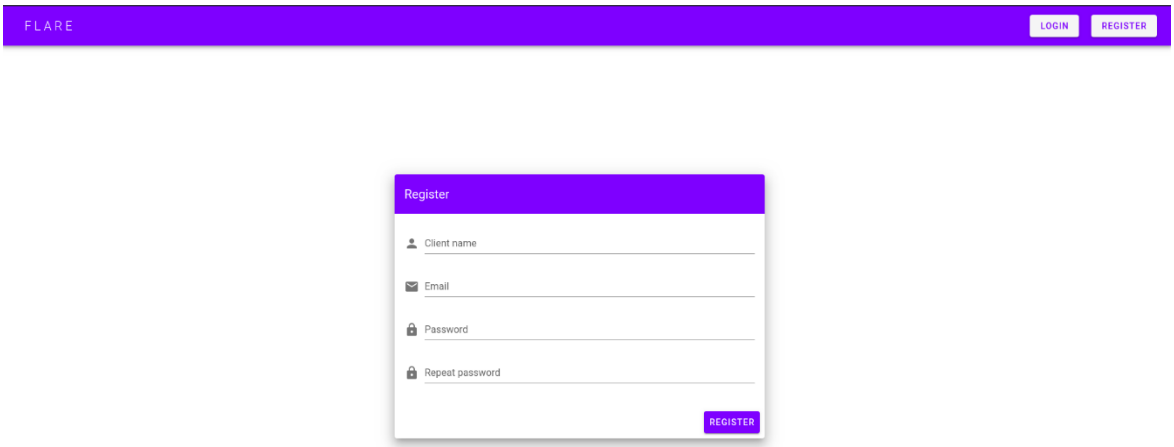
Korisničko sučelje

Registracija

Registracija je postupak kojim se u sustavu kreiraju novi korisnici. Sadržaj ovog pogleda predstavlja okvir koji sadrži formu za unos podataka o novom korisniku. Potrebno je upisati sljedeće podatke: korisničko ime, email, lozinku te ponovljenu lozinku (Slika 8).

Slika 8

Pogled za registraciju



Forma ima validaciju i na klijentskom dijelu aplikacije i na poslužitelju. Validacija na klijentskoj strani provjerava jesu li upisana sva polja. Također, provjerava se da lozinka sadrži barem 8 znakova kako bi se osiguralo da korisnici ne koriste jednostavne i lako provaljive lozinke. Lozinku je potrebno upisati dvaput kako bi se spriječio slučajan unos pogrešne lozinke korisnika. U svrhu provjere formata emaila, validira se i polje za unos emaila. Ukoliko su sva polja uspješno validirana na klijentskoj strani, zahtjev se šalje na poslužitelj. Na poslužitelju slijedi drugi korak validacije. Budući da se korisničko ime koristi kao dio URL-a kod dohvaćanja slike, dobiveno polje potrebno je pretvoriti u niz znakova koji je prihvatljiv za URL. Nakon toga se provjerava postoji li već korisnik sa istim korisničkim imenom. Ukoliko je korisničko ime jedinstveno, provjerava se jedinstvenost emaila budući da se email koristi prilikom prijave korisnika u sustav. Tek u ovom trenutku validacija je uspješna te poslužitelj obavlja potrebne akcije kako bi se korisnik kreirao.

Sve gore spomenute validacijske greške, uključujući one na poslužiteljskoj strani, prikazuju se korisniku na dnu forme, te se onemogućuje predaja forme dok sve greške nisu uklonjene.

Po uspješnoj registraciji, korisnik je automatski prijavljen u sustav.

Prijava

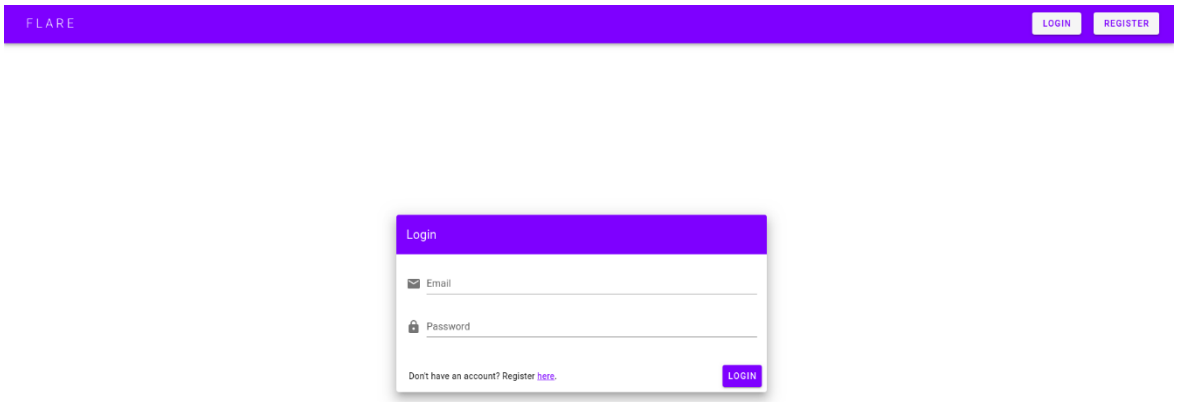
Za pristup svim pogledima, osim pogledu za registraciju, korisnik mora biti autoriziran. Pri pokušaju pristupa jednom takvom pogledu, korisnik će biti preusmjeren na pogled za prijavu (Slika 9). Taj pogled se sastoji od forme koja sadrži polja za unos emaila i lozinke. Ukoliko korisnik sa upisanim emailom nije registriran u

sustavu, ili je lozinka pogrešna, korisniku će se na dnu forme prikazati poruka da podaci za prijavu nisu ispravni.

Po uspješnoj prijavi, korisnik je preusmjeren na pogled za upravljanje sadržajem.

Slika 9

Pogled za prijavu

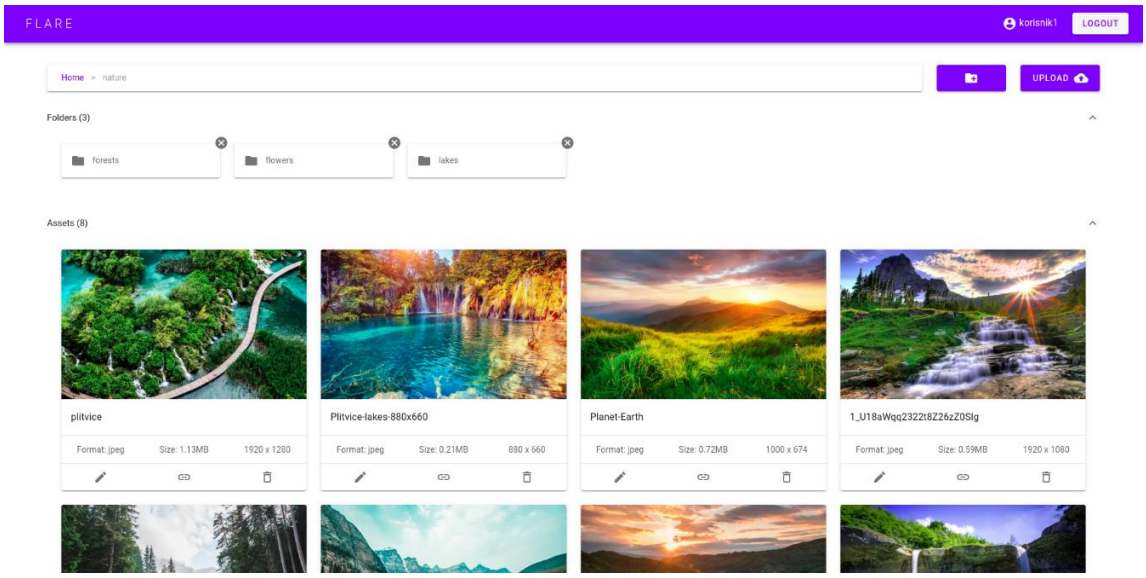


Upravljanje sadržajem

Glavni pogled ove aplikacije je pogled za upravljanje sadržajem (Slika 10). To je pogled na koji je korisnik preusmjeren prilikom prijave u aplikaciju. Korisnik se u ovom pogledu uvijek nalazi u nekom direktoriju. Polazni direktorij je korijenski direktorij. Korisnik u svakom trenutku vidi svoju lokaciju u strukturi direktorija u obliku, tzv. krušnih mrvica (engl. *breadcrumbs*) pri vrhu pogleda.

Slika 10

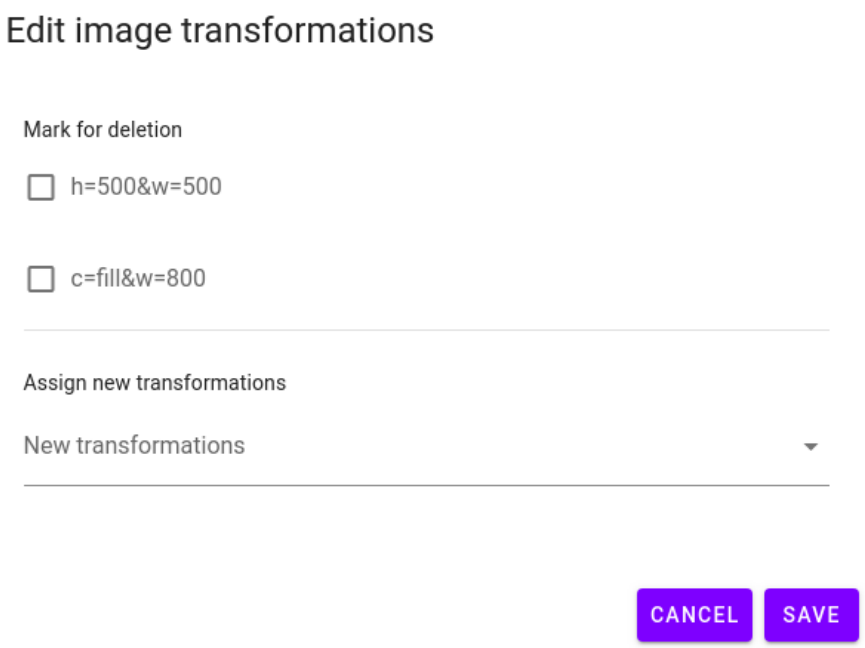
Pogled za upravljanje sadržajem



Sadržaj trenutnog direktorija prikazan je u dvije sekcije. Prva sekcija sadrži direktorije u obliku kartica. Pritiskom na gumb u gornjem desnom kutu kartice korisnik može obrisati direktorij. Prije same akcije brisanja prikazuje se dijalog za potvrdu. Nakon potvrde korisnika rekurzivno se brišu svi direktoriji i slike unutar tog direktorija. U drugoj sekciji prikazane su slike. Ispod svake slike nalaze se osnovne informacije o slici: ime slike, dimenzije slike, format i veličina. Ispod informacija o slici postoje tri gumba koja predstavljaju tri akcije: uređivanje verzija, kopiranje URL-a slike i brisanje slike. Prilikom pritiska na gumb za uređivanje verzija prikazuje se okvir s formom gdje se postojeće verzije mogu označiti za brisanje kao i dodati nove (Slika 11).

Slika 11

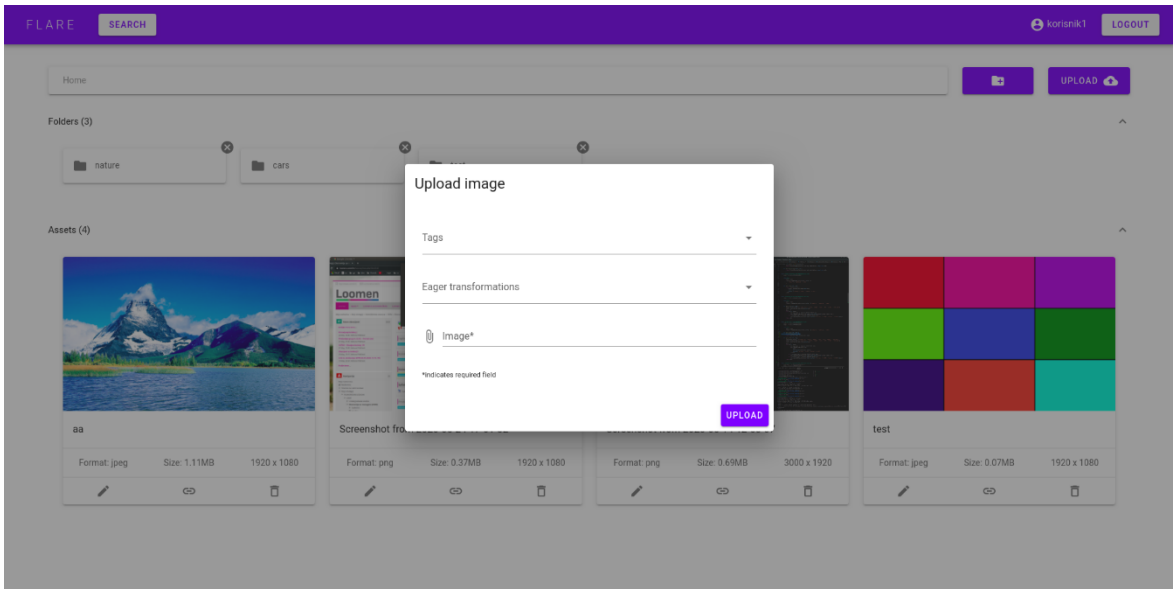
Okvir s formom za uređivanje verzija



Gdjegod da se korisnik nalazi u strukturi direktorija, omogućeno mu je dodavanje novog direktorija i učitavanje slike u trenutni direktorij. Prilikom učitavanja slike korisnik može zadati tagove relevantne za sliku, te verzije slike koje će se asinkrono kreirati na poslužitelju (Slike 12 i 13).

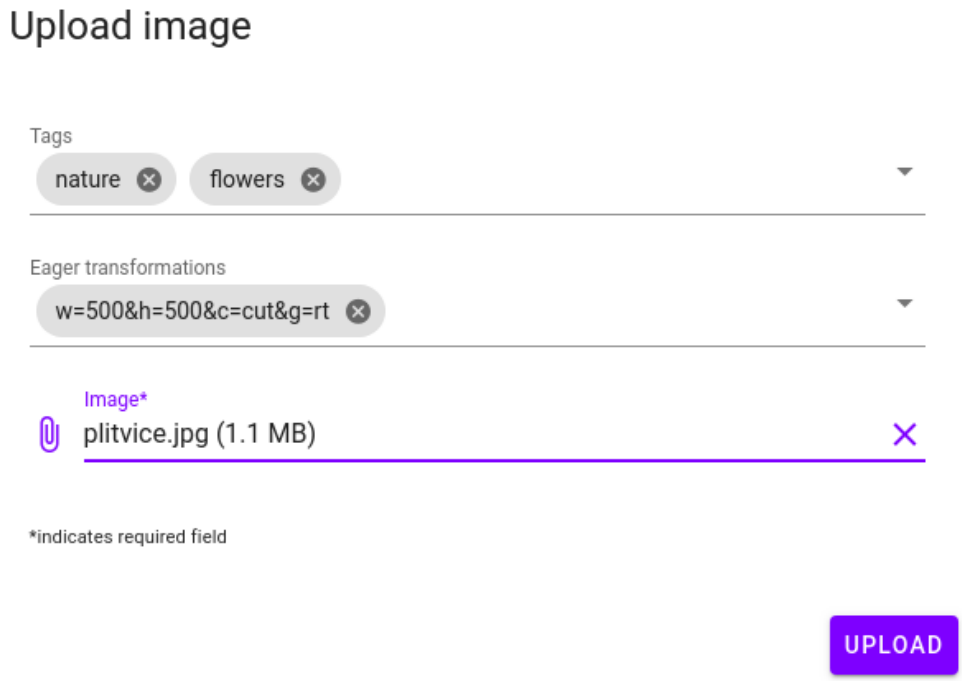
Slika 12

Okvir s formom za učitavanje slike



Slika 13

Ispunjena forma za upload slike



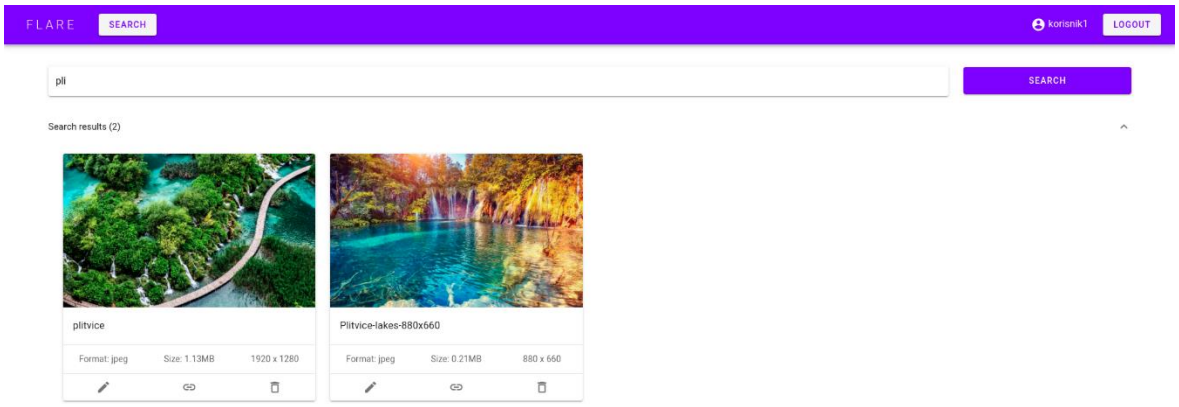
Po uspješnom izvođenju svake akcije korisniku se pokazuje obavijest u obliku pop-upa pri dnu ekrana u trajanju od dvije sekunde.

Pretraživanje slika

Osim u pogledu za upravljanje sadržajem, korisnik slike može pronaći i pretraživanjem. Klikom na odgovarajući gumb u navigacijskoj traci, otvara se pogled za pretraživanje slika (Slika 14). U gornjem dijelu pogleda nalazi se polje za unos teksta. Korisnik može pretraživati slike prema imenu ili prema tagovima zadanim pri učitavanju slike. Pri pretraživanju se ne uzima u obzir jesu li slova velika ili mala. Slika je prikazana kao rezultat pretrage ukoliko počinje sa slovima odnosno riječima zadanim u polju za unos teksta ili ako upisane riječi sadrži kao tagove.

Slika 14

Pogled za pretraživanje slika

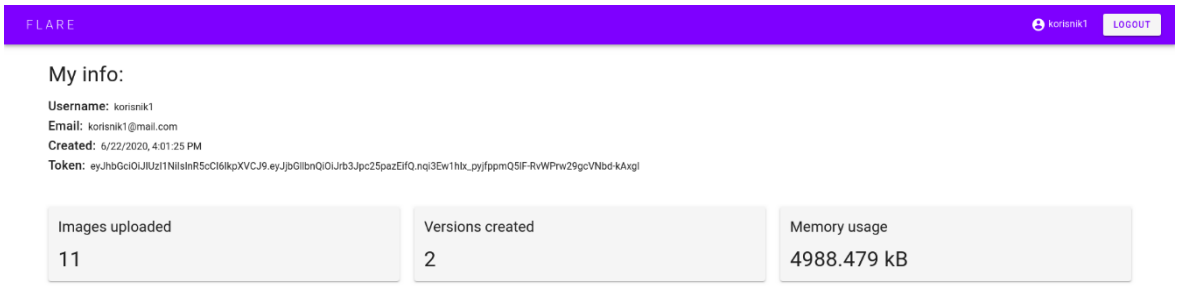


Profil korisnika

Pritiskom na korisničko ime u navigacijskoj traci otvara se pogled s podacima o korisniku (Slika 15). U ovom pogledu korisnik vidi podatke o svom računu: korisničko ime i email s kojim je račun napravljen, vrijeme kreiranja računa te token za pristup API krajnjim točkama zaštićenim autorizacijom. Osim podataka o računu, korisniku su pokazani podaci o korištenju servisa koji uključuju broj učitanih slika, broj kreiranih verzija te ukupno zauzeće memorije.

Slika 15

Profil korisnika



Zaključak

U ovom radu prikazan je cjeloviti postupak razvoja web aplikacije za verzioniranje slika. Prva faza sastoji se od razvoja web servisa i logike verzioniranja slika. Za njegovu izradu korišten je programski jezik Go zbog svoje brzine i jednostavnosti pri implementaciji konkurentnih aktivnosti. Web servis je realiziran kao aplikacijsko programsko sučelje slijedeći načela REST arhitekture. Podaci koji se pohranjuju na poslužitelju spremljeni su u nerelacijsku bazu podataka zbog jednostavnosti korištenih modela, brzine pristupa te mogućnosti spremanja objekata s različitom strukturom u istu kolekciju.

Druga faza razvoja sastoji se od izrade klijentske aplikacije u razvojnom okviru Vue.js. Korištenje razvojnog okvira ubrzalo je proces razvoja zbog mogućnosti korištenja postojećih rješenja za često korištene funkcionalnosti. Klijentska aplikacija omogućuje korisniku konzumiranje API-ja preko intuitivnog korisničkog sučelja.

Prethodno spomenute cjeline čine aplikaciju Flare koja korisnicima može olakšati razvoj vlastitih programskih rješenja pružajući im jedinstveno i učinkovito rješenje za upravljanje slikama i njihovim verzijama.

Literatura

Vue (2020). Vue.js overview. https://v1.vuejs.org/guide/overview.html

Omole, O., (2019). Nuxt.js: a Minimalist Framework for Creating Universal Vue.js Apps. Sitepoint. https://www.sitepoint.com/nuxt-js-universal-vue-js/

Vuetify (2020). Why Vuetify. https://vuetifyjs.com/en/introduction/why-vuetify/

Material, Material Design. https://material.io/design

Golang, Frequently asked questions. https://golang.org/doc/faq

Race detector, Data race detector. https://golang.org/doc/articles/race_detector.html

Jones, M., Bradley, J., Sakimura, N. (2020). JSON Web Token. https://tools.ietf.org/html/rfc7519

Josefsson, S. (2020). The Base16, Base32, and Base64 Data Encodings. https://tools.ietf.org/html/rfc4648#section-4

Vue.js components. https://vuejs.org/v2/guide/components.html

REST, What is REST. https://restfulapi.net/

MongoDB, What is MongoDB. https://www.mongodb.com/what-is-mongodb

Markuš, N., Frljak, M., Pandžić, I. S., Ahlberg, J., Forchheimer, R. Object Detection with Pixel Intensity Comparisons Organized in Decision Trees, (2014). https://arxiv.org/pdf/1305.4537.pdf

OpenCV, About OpenCV, https://opencv.org/about/

O autorima

Ivan Zerec je magistar inženjer računarstva, a svoj završni i diplomski rad obranio je na Fakultetu elektrotehnike i računarstva Sveučilišta u Zagrebu. U fokusu njegovih istraživačkih interesa je razvoj web aplikacija i servisa. Trenutno radi u tvrtci Shape na poziciji Backend Developera gdje radi na brojnim projektima u programskim jezicima Python i Go. Autora se može kontaktirati na ivan.zerec@fer.hr.

Marina Bagić Babac docentica je na Zavodu za primijenjeno računarstvo Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu. Na istom Fakultetu stekla je titule magistrice i doktorice znanosti u području elektrotehnike, a na Fakultetu političkih znanosti Sveučilišta u Zagrebu titulu diplomiranog novinara. Trenutno je suradnica na nekoliko međunarodnih projekata vezanih uz lingvističke podatke i strojno učenje. Članica je programskog odbora nekoliko međunarodnih znanstvenih konferencija i recenzentica u brojnim međunarodnim časopisima. Njezin istraživački interes uključuje obradu prirodnog jezika, društvene medije i analizu društvenih mreža te je

objavila preko četrdeset znanstvenih radova. Autorica se može kontaktirati na marina.bagic@fer.hr.