Razvoj i monetizacija socijalnih mreža u programskom okviru Phalcon PHP

Tomašković, Roman

Master's thesis / Diplomski rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Organization and Informatics / Sveučilište u Zagrebu, Fakultet organizacije i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:211:902172

Rights / Prava: Attribution-NonCommercial 3.0 Unported/Imenovanje-Nekomercijalno 3.0

Download date / Datum preuzimanja: **2024-05-03**

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Repository / Repozitorij:

Faculty of Organization and Informatics - Digital Repository

zir.nsk.hr

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Roman Tomašković

RAZVOJ I MONETIZACIJA SOCIJALNIH MREŽA U PROGRAMSKOM OKVIRU PHALCON PHP

DIPLOMSKI RAD

Varaždin, 2021.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Roman Tomašković

Matični broj: 41675/17–R

Studij: Informacijsko i programsko inženjerstvo

RAZVOJ I MONETIZACIJA SOCIJALNIH MREŽA U PROGRAMSKOM OKVIRU PHALCON PHP

DIPLOMSKI RAD

Mentor :

Prof. dr. sc. Dragutin Kermek

Varaždin, prosinac 2021.

Roman Tomašković

Izjava o izvornosti

Izjavljujem da je moj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor potvrdio prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Cilj diplomskog rada je razviti društvenu mrežu temeljenu na programskom okviru Phalcon PHP i ovisno o tehničkom dizajnu, potencijalno drugim okvirima. Shodno tome, opisati će se pojmovi kao što su društvena mreža i navesti specifičnosti i izazovi koje razvoj takve web aplikacije donosi. Opisati će se strategije monetizacije, razmotriti koje od njih bi bile primjenjive s obzirom na dizajn društvene mreže te naposljetku opisati provedbu monetizacije. Na samom kraju, prikazati će se tehnička dokumentacija projekta, u vidu raznih dijagrama koji opisuju strukturu i dizajn aplikacije, zajedno s objašnjenjima zašto je izabran određeni dizajn dijelova aplikacije. Ta će se aplikacija razložiti na temeljne funkcionalnosti čija će se implementacija i programski kôd pokazati i pojasniti.

Ključne riječi: web;php;phalcon;društvene mreže;monetizacija

Sadržaj

1. Uvod . . . 1

2. Metode i tehnike rada . . . 3
2.1. Angular Material UI . . . 3
2.2. RxJS . . . 3
2.3. Mariadb . . . 4
2.4. PHPStorm . . . 4
2.5. Git . . . 5
2.5.1. GitLab . . . 6
2.6. XAMPP . . . 9
2.7. Mailtrap . . . 10

3. Korišteni jezici i okviri . . . 11
3.1. PHP . . . 11
3.1.1. Povijest . . . 11
3.2. Phalcon PHP . . . 14
3.2.1. Zephir . . . 14
3.2.2. ADR . . . 15
3.2.3. Usporedba sa drugim okvirima . . . 17
3.2.3.1. Tehničke usporedbe . . . 17
3.2.3.2. Zahtjevi u sekundi . . . 18
3.2.3.3. Vrijeme obrade zahtjeva . . . 19
3.2.3.4. Uključene datoteke . . . 20
3.2.3.5. Uporaba memorije . . . 21
3.2.3.6. Općenita usporedba . . . 22
3.2.3.7. Popularnost . . . 24
3.2.3.8. Ekosustav . . . 24
3.2.3.9. Alati za naredbeni redak . . . 25
3.2.3.10. Dokumentacija . . . 26
3.2.3.11. Jednostavnost . . . 26
3.2.3.12. Zaključak . . . 27
3.2.4. Instalacija PhalconPHP okvira . . . 27
3.2.5. MVC . . . 29
3.3. Angular . . . 32
3.3.1. Kratka povijest . . . 35
3.3.2. Typescript . . . 36
3.3.2.1. Primjeri . . . 36

4. Društvene mreže . . . 38
4.1. Glavne značajke . . . 38
4.2. Povijesni pregled razvoja društvenih mreža . . . 38
4.3. Community Memory . . . 39
4.3.1. PLATO . . . 39
4.3.1.1. Talkomatic i Term-Talk . . . 40
4.3.1.2. PLATO Notes . . . 41
4.3.2. Tekstualne mreže . . . 42
4.3.3. SixDegrees . . . 45
4.3.4. Friendster . . . 45
4.3.5. MySpace . . . 47
4.3.6. Facebook . . . 47
4.3.7. Reddit . . . 49
4.3.8. Youtube . . . 51
4.3.8.1. Alternativni servisi . . . 52
4.3.9. Twitter . . . 52
4.3.10. Instagram . . . 53
4.3.11. TikTok . . . 55

5. Monetizacija . . . 56
5.0.1. Patreon . . . 61

6. Specifikacije i funkcionalni zahtjevi . . . 65
6.1. Funkcionalna raščlamba projekta na visokoj razini . . . 65
6.2. Detaljna raščlamba funkcionalnosti . . . 65
6.2.1. Registracija . . . 65
6.2.2. Prijava . . . 66
6.2.3. Korisnički profil . . . 66
6.2.4. Korisničke grupe . . . 67
6.2.5. Oznake . . . 67
6.2.6. Pregledavanje i pretraživanje sadržaja . . . 67
6.2.7. Chat . . . 68

7. Tehnička implementacija projekta . . . 69
7.1. Arhitektura sustava . . . 69
7.2. Podatkovni model . . . 71
7.3. Dijagram slučajeva korištenja . . . 73
7.4. Dijagrami slijeda . . . 75
7.4.0.1. Prijava . . . 75
7.4.0.2. Objavljivanje . . . 77
7.4.0.3. Naplata . . . 78
7.5. Struktura kôda i dijagrami klasa . . . 80
7.5.1. Poslužiteljska strana . . . 80
7.5.1.1. Kontroleri . . . 83

7.5.1.2. Entiteti . . . 84
7.5.1.3. Modeli . . . 85
7.5.1.4. Repozitoriji . . . 85
7.5.1.5. Resursi . . . 85
7.5.1.6. Transformatori . . . 87
7.5.2. Klijentska strana . . . 88
7.5.3. Navigacija . . . 94
7.6. Zavisnosti . . . 97
7.6.1. Klijentska aplikacija . . . 97
7.6.2. Poslužiteljska aplikacija . . . 98
7.7. Konfiguracija . . . 99
7.8. Detalji implementacije . . . 102
7.8.1. Prijava . . . 102
7.8.2. Registracija . . . 109
7.8.3. Objavljivanje . . . 116
7.8.4. Naplata . . . 123
7.9. REST API . . . 128
7.9.1. Prijava . . . 128
7.9.2. Odjava . . . 129
7.9.3. Zaboravljena lozinka . . . 129
7.9.4. Višefaktorska autentikacija . . . 130
7.9.5. Korisnički profil . . . 133
7.9.6. Registracija . . . 133
7.9.7. Ažuriranje korisnika . . . 134
7.9.8. Dohvaćanje korisničkih objava . . . 135
7.9.9. Brisanje objave . . . 136

8. Zaključak . . . 137

Popis literature . . . 143

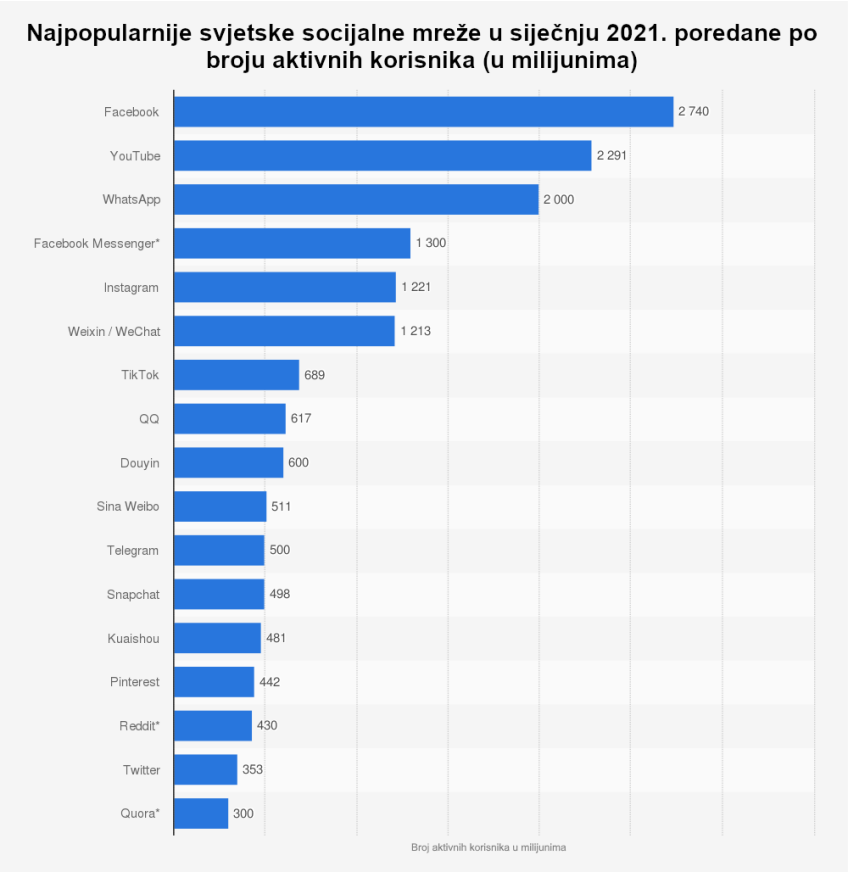
Popis slika . . . 146

Popis tablica . . . 147

Popis isječaka kôda . . . 147

1. Uvod

Društvene mreže termin su za web aplikacije koje dozvoljavaju korisnicima izradu profila kojim se predstavljaju i interakciju sa drugim korisnicima i sadržajem kojeg izrađuju kako vlasnici platforme, tako i njihovi korisnici. Danas su iznimno popularne te se procjenjuje da više od pola stanovništva danas koristi društvene mreže u jednome ili drugome obliku. Izuzmu li se aplikacije za dopisivanje (kao što su WhatsApp i Messenger), među najpopularnijim društvenim mrežama današnjice su Facebook, Instagram, Twitter, Youtube, Tiktok, Reddit i Telegram (iako je Telegram primarno platforma za dopisivanje, sadrži mnoge elemente društvenih mreža te ga se može smatrati svojevrsnim hibridom).



Slika 1: Popularnost društvenih mreža po broju korisnika

Izvor: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/

Društvene mreže su od posebnog interesa korporacijama, budući da velika korisnička baza koja generira sadržaj predstavlja veliki monetizacijski potencijal i samim time izvor prihoda. Radi se naravno o značajnoj temi za diskusije, budući da postoje kako zagovornici toga, tako i protivnici te takve poslovne prakse ipak povlače za sobom određene moralne i legalne implikacije i podižu pitanja.

Iako su društvene mreže danas postigle određenu razinu zrelosti, mogu se očekivati ipak daljnje promjene i novosti u budućnosti, tako da one predstavljaju još uvijek interesantnu i

opširnu tematiku. Stoga postoje raznoliki razlozi i motivacije za odabir teme ovog rada.

Izazov Izrada društvene mreže veliki je poduhvat i predstavlja znatan izazov osobi koja poduzima takav projekt, budući da se radi o arhitekturalno kompleksnom sustavu na čije oblikovanje utječe mnogo faktora.

Motivacija Budući da se radi o kompleksnom projektu, sljedeće se postavlja pitanje motivacije pojedinca za takav projekt. Prije svega, sam izazov je motivacija autoru rada, budući da predstavlja priliku za značajan razvoj vještina i znanja.

U sklopu rada biti će prikazani korišteni alati i metodika rada, pojašnjeni koncepti društvenih mreža i njihova povijest te će biti prikazana izrada društvene mreže i njene strukture na tehničkoj razini.

2. Metode i tehnike rada

Kako se ne radi o jednostavnome projektu te se za izradu istoga koriste različiti programski jezici i tehnološka rješenja, potrebno je koristiti različite programske alate kako bi razvoj bio što jednostavniji. Tako su se u svrhu izrade projekta koristila sljedeća programska rješenja i alati:

- PhalconPHP (detaljnije opisano u 3.1)
- Angular (Typescript, detaljnije opisano u 3.3)
  - Angular Material UI
  - RxJS
- Mariadb
- PHPStorm
- Git
  - GitLab
- XAMPP
- Mailtrap

Dotični alati bit će redom opisani u sljedećim potpoglavljima.

2.1. Angular Material UI

Kako je opisano u odjeljku 3.3, Angular je idealan za razvoj kompleksnih sučelja sastavljenih od jednostavnijih individualnih podkomponenti. Kao takav, vrlo je podoban za implementaciju sustava dizajna koji su svojom filozofijom temeljeni na takvom komponentom načinu sastavljana sučelja.

Jedan takav primjer, koji je korišten za razvoj sučelja projekta, jest Google-ov tzv. materijalni dizajn (engl. *Material Design*). Angular Material službeni je paket Angular razvojnog tima koji implementira filozofiju materijalnog dizajna te tako nudi gotove komponente za izgradnju korisničkih sučelja. Uz gotove komponente, nudi i alate za izradu daljnjih namjenskih komponenti temeljenih na materijalnom dizajnu kako bi omogućilo izradu dodatnih i proširenje postojećih funkcionalnosti (Angular Components Team 2021.).

2.2. RxJS

RxJS je biblioteka za izradu asinkronih i na događajima temeljenih programa koristeći takozvane *posmotrive sekvence* (engl. observable sequences). RxJS uvodi koncept "reaktiv-

nog" programiranja te kombinira *Observer* i *Iterator* uzorke dizajna sa funkcijskim programiranjem i kolekcijama kako bi ponudio elegantan sustav za rukovanje nizovima događaja (RxJS 2021.).

2.3. Mariadb

Društvene mreže pohranjuju potencijalno velike količine različitih podataka, kao što su podaci o korisniku, sadržaj koji korisnici učitavaju i pohranjuju na mreži te meta-podaci potrebni za raznolike funkcionalnosti unutar aplikacije. Uobičajeno je da se takvi podaci pohranjuju u tzv. *bazi podataka*. Postoji više vrsta baza podataka, neke od kojih su relacijske (primjerice Oracle, MySQL, Postgres i MSSQL), NoSQL (MongoDB, CouchDB, itd.), memorijske (Redis) i tako dalje. Danas najpoznatiji modeli baza podataka koji su u uporabi su relacijski model, NoSQL model i memorijske baze podataka (Harrison 2015.).

Za svrhe ovog projekta korištena je MariaDB, ogranak poznate MySQL baze podataka kojeg su stvorili originalni razvojni programeri MySQL-a nakon što ih je kupio Sun Microsystems (u produžetku Oracle, budući da je Oracle kupio Sun). MariaDB nastoji zadržati visoku kompatibilnost sa MySQL bazom te se one tako mogu uglavnom koristiti naizmjenično. Ukratko, MariaDB je relacijska baza podataka otvorenoga-kôda koja podržava višedretvenost rad te je jedna od njenih najvećih prednosti što je besplatna za uporabu (kao i sam MySQL), ali usprkos tomu što je besplatna ipak nudi pouzdanost, brzinu i značajke koje ju čine ozbiljnim odabirom kod izrade aplikacija (Dyer 2015.).

2.4. PHPStorm

PHPStorm je integrirano razvojno okruženje (engl. integrated development environment) specijalizirano za razvoj programa u PHP jeziku. Razvijeno u Java programskome jeziku od strane tvrtke JetBrains i izdan prvi puta 2009. godine. Zadnja inačica je 2021.3. PHPStorm nudi kompletno razvojno sučelje za web programiranje sa ugrađenom podrškom za PHP 8, JavaScript, HTML i CSS.

Sav PHP alat Uz to, podržava mnoge poznate razvojne okvire kao što su Symfony, Laravel, WordPress, Zend, Magento i tako dalje (JetBrains 2021.b).

Klijentske tehnologije uključene Podržava JavaScript, HTML i CSS koji su osnovne web tehnologije ali i njihove dopune i proširenja kao što su CoffeeScript, TypeScript, Sass, Less i Emmet (JetBrains 2021.b).

Pomoć u razvoju Kao alat specijaliziran za PHP razvoj, PHPStorm nudi dublje razumijevanje PHP kôda i strukture sa izvrsnim automatskim prijedlozima i popunjavanjem te tako ubrzava i olakšava razvoj (JetBrains 2021.b).

Dodatni ugrađeni alati Pored razvoja kôda, također ima ugrađene alate za pristupanje različitim izvorima podataka kao što su PostgreSQL i Redis, ugrađenu podršku za različite sustave za kontrolu kôda kao što su Git, Subversion i Mercurial, generiranje UML dijagrama na temelju kôda, udaljeni pristup putem SSH, FTP i SFTP (JetBrains 2021.b).

Brzo refaktoriranje Izmjena kôda na visokoj razini također je jednostavna uz ugrađene alate za refaktoriranje, preslagivanje, optimizaciju uvezenih paketa i biblioteka i analizu kôda (JetBrains 2021.b).

Otklanjanje greška i testiranje Ugrađena podrška za Xdebug i Zend Debugger alate bez potrebe za posebnim podešavanjem kao i za testiranje kôda putem PHPUnita ili Behata (JetBrains 2021.b).

2.5. Git

Git je distribuirani sustav za kontrolu verzija pohranjenog sadržaja u digitalnom obliku, najčešće programskog kôda i tekstualnih datoteka. Radi se o alatu koji svojom funkcionalnošću, performansama, sigurnošću i fleksibilnosti zadovoljava potrebe velike većine individualnih programera i razvojnih timova, uključujući i vrlo velike timove (Ahmed 2020.).

Izrađen je od strane Linusa Torvaldsa 2005. godine kao važan alat u razvoju Linuxa (Ahmed 2020.);

Otvoreni programski kôd Objavljen je pod GPL licencom te je potpuno besplatan, javno dostupan i dozvoljene su preinake u skladu s potrebama korisnika (Ahmed 2020.).

Distribuirani sustav za kontrolu verzija Distribuirani sustavi za kontrolu verzija se ne moraju oslanjati na središnji server za pohranu svih verzija projektnih datoteka. Svaki korisnik ima lokalnu kopiju svih podataka u glavnom spremištu. Lokalni podaci se usklađuju s glavnom verzijom putem operacije povlačenja razlike u podacima između glavnog spremišta i lokalne kopije (engl. *"Pull"*). Lokalne promjene se mogu prenijeti na glavnu verziju putem operacije učitavanja lokalne razlike podataka u glavno spremište (engl. *"Push"*) (Ahmed 2020.).

Brzina rada Git ne zahtijeva povezivanje na mrežu ili neki server pa je sve operacije moguće izvršavati lokalno na korisničkom računalu. To omogućava izmjerene brzine rada najmanje za red veličine brže od drugih sustava za kontrolu verzija, a kod operacija vezanih za povijest promjena i do dva reda veličine (Ahmed 2020.).

Pouzdanost Kako svaki korisnik ima kopiju cijelog projekta na svojem lokalnom računalu, u slučaju ispada glavnog poslužitelja, oporavak podataka je jednostavan putem operacije učitavanja (engl. *"Push"*) s bilo kojeg korisničkog računala. Dakle, sigurnosna kopija podataka uvijek postoji (Ahmed 2020.).

Skalabilnost Sustav lako podržava rast broja korisnika. Podaci na korisničkim računalima se nalaze u sažetim datotekama tako da razmjena podataka s glavnim spremištem nije zahtjevna (Ahmed 2020.).

Sigurnost Za imenovanje i identifikaciju datoteka Git koristi SHA1 (engl. Secure Hash Function) funkciju za stvaranje kriptografskih sažetaka. Kod spremanja i dohvata svake datoteke koristi se njezin "kontrolni zbroj" (engl. *checksum*). Povijest promjena se sprema na takav način da ID svake verzije zavisi od kompletne povijesti promjena. Stoga, nije moguća naknadna izmjena datoteka bez da se to jasno ne vidi (Ahmed 2020.).

Ekonomičnost Kod centraliziranih sustava kontrole verzija, središnji poslužitelj mora biti dovoljno snažan da zadovolji potrebe cijelog tima. Ako broj članova tima raste, performanse servera mogu postati usko grlo projekta. Git, kao distribuirani sustav, nema potrebe za komunikacijom sa središnjim serverom osim pri operacijama usklađivanja podataka ("Pull" i "Push"). Sve zahtjevne operacije se odvijaju na korisničkim računalima te tako nije nužno za središnji server da raspolaže velikim resursima (Ahmed 2020.).

Jednostavno grananje projekta Upravljanje grananjem projekta na Git-u je iznimno jednostavno. Nove grane se stvaraju, brišu ili spajaju u sekundama. To osigurava da se u glavnoj grani projekta nalazi isključivo produkcijski kôd, a drugi razvoj i sve nove funkcionalnosti se razvijaju u odvojenim granama (Ahmed 2020.).

2.5.1. GitLab

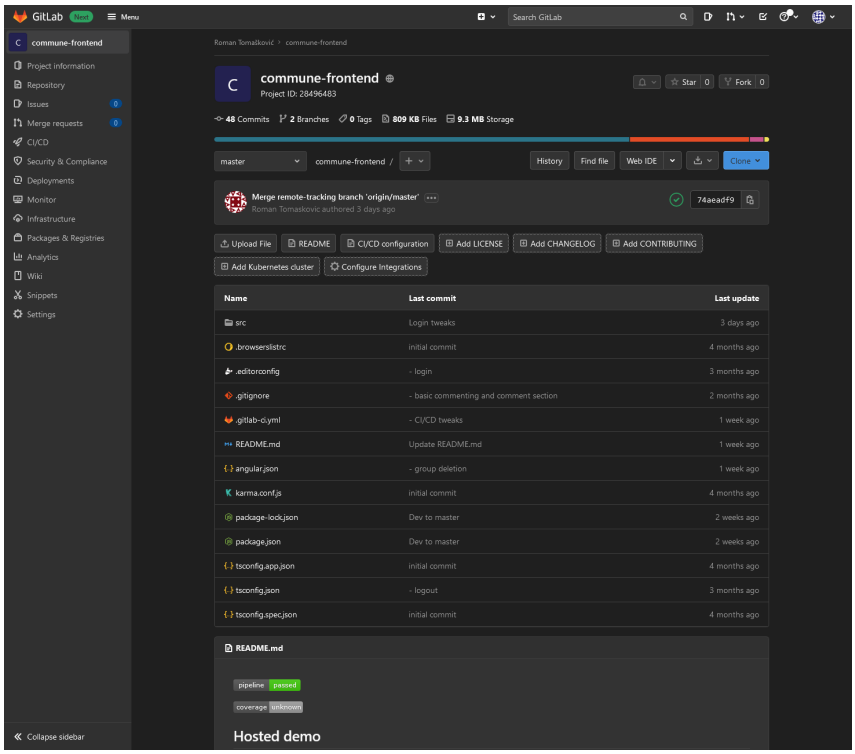
GitLab je online DevOps platforma za razvoj i distribuciju programa te unutar sebe ugrađuje spomenuti Git sustav za kontrolu verzija. Sam pojam *DevOps* skraćenica je dobivena iz spoja engleskih pojmova *software development* (hrv. razvoj softvera) i *IT operations* (hrv. informatička podrška) te koncept DevOpsa spaja filozofije, alate i prakse obje strane i za cilj ima ubrzati razvoj, poboljšati suradnju i povećati kvalitetu softvera koji se isporučuje (Atlassian 2021.). Uz GitLab postoje još i druga rješenja kao što su GitHub i Bitbucket. GitLab dolazi u dvije verzije:

- SaaS - Inačica GitLaba koja se poslužuje na njihovim poslužiteljima. Najjednostavnija za uporabu no nije ju moguće konfigurirati po volji.
- Self-Managed - GitLab poslužen na vlastitoj infrastrukturi. Nudi veću kontrolu (GitLab 2021.b).

GitLab je besplatan za uporabu no podršku nudi isključivo korisnicima koji plaćaju.

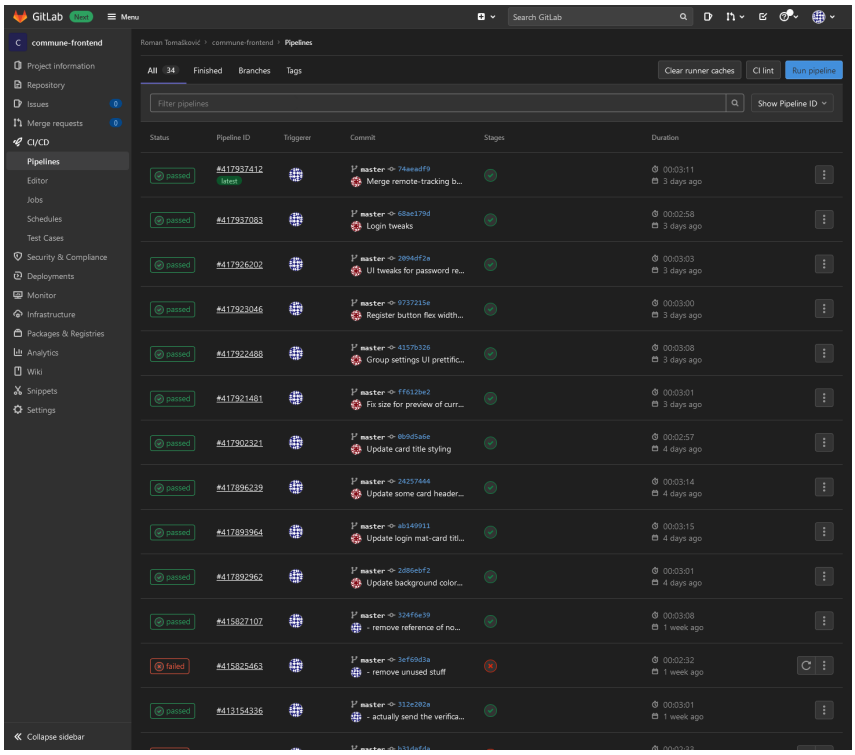
GitLab stavlja korisniku na raspolaganje mnogo mogućnosti i alata kako bi upravljao svojim kôdom te kako bi ga analizirao i distribuirao. Nudi alate za organizaciju agilnog razvoja kroz kreiranje zadataka, ciljeva, detaljno opisivanje, pa alate za razvoj i integraciju sa drugim programima poput Jire. Jedna od najzanimljivijih značajki su njihovi alati za kontinuiranu

integraciju i distribuiranje. (engl. CI/CD) putem kojih se omogućava razvoj automatiziranih procedura za testiranje i verifikaciju kôda i njegovo sigurnosno testiranje te njegovo pakiranje i distribuiranje sa podrškom za mnoge registre i repozitorije programa kao što su Maven, npm i Composer (GitLab 2021.a).



Slika 2: Glavna stranica repozitorija na GitLab-u

Uz čuvanje i kontrolu verzija kôda, GitLab je bio korišten i kako bi se uspostavilo automatizirano distribuiranje novih verzija kôda za klijentsku aplikaciju.



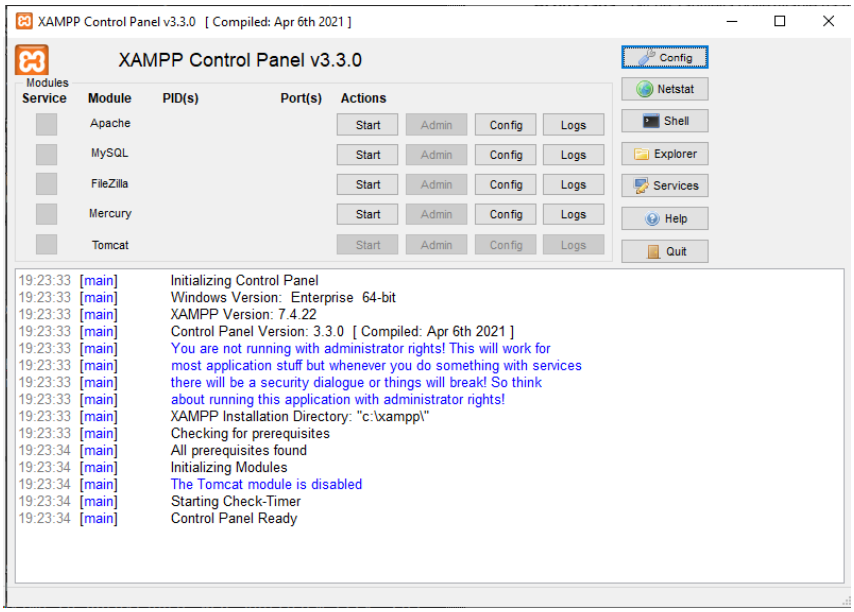
Slika 3: GitLab-ovo sučelje za pregled kontinuiranje integracije

Kontinuirana integracija se podešava kreiranjem posebne datoteke *.gitlab-ci.yml* koja sadrži potrebnu konfiguraciju i korake koje GitLab treba odraditi kada se distribuira nova verzija kôda. Primjerak takve datoteke za potrebe klijentske aplikacije prikazan je u isječku 2.1.

```
1  image: node:14.17.0
2
3  pages:
4      cache:
5          paths:
6              - node_modules/
7
8      stage: deploy
9      script:
10         - npm install -g @angular/cli@12
11         - npm install
12         - ng build --output-path public --base-href=/commune-frontend/
13         - cp public/index.html public/404.html
14         - cp public/index.html public/about.html
15     artifacts:
16         paths:
17             - public
18     only:
19     - master
```

Izvorni kôd 2.1: CI/CD konfiguracijska datoteka za GitLab

Jednostavno rečeno, projektne zavisnosti koje se nalaze u *node_modules* direktoriju se pohranjuju u predmemoriju i u koraku distribucije definirana je skripta koja odrađuje potrebne



Slika 4: Kontrolna ploča XAMPP programa

radnje kako bi se izgradila klijentska aplikacija na GitLabovom poslužitelju i premjestila u poseban direktorij *public* kojeg na razini repozitorija GitLab poslužuje i na kojem klijentska aplikacija zapravo jest dostupna (tzv. *GitLab pages* funkcionalnost).

2.6. XAMPP

XAMPP je, po tvrdnjama svojih autora, najpopularnija gotova okolina za razvoj PHP aplikacija.

XAMPP gotova je razvojna okolina za PHP koja unutar sebe sadrži Apache poslužitelj, MariaDB bazu podataka, PHP jezik i Perl jezik. Naziv XAMPP dolazi od skraćenice *cross-platform apache + MariaDB + PHP + Perl*. Glavna svrha takvog programskog paketa je maksimalno pojednostavljenje pripreme razvojne okoline za razvoj koja se može zakomplicirati kada je potrebno instalirati i pripremiti nekolicinu zasebnih programskih paketa (Apache Friends 2021.).

Dodatne prednosti su također i dostupnost dodatnih popularnih aplikacija koje se sa lakoćom mogu instalirati u sklopu XAMPP sustava kao što su Wordpress i Joomla!. Dostupan je za operacijske sustave Windows, Linux i OS X te tako pokriva široku bazu korisnika. U trenutku pisanja (2.12.2021.) zadnja dostupna verzija je 8.0.13. (u skladu sa podržanom verzijom PHP jezika).

XAMPP je bio korišten kako bi se omogućila brza i jednostavna razvojna okolina za projekt iako su bile razmatrane i druge opcije kao što su Devilbox (virtualna razvojna okolina temeljena na Dockeru).

2.7. Mailtrap

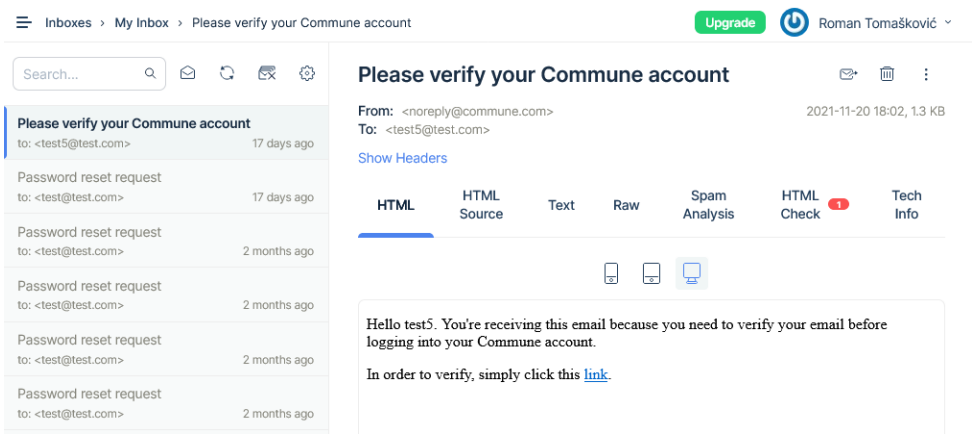
Mailtrap je platforma koja nudi alata za jednostavan rad s SMTP prometom, odnosno s e-mail porukama. Nudi nekoliko usluga, koje su redom:

- Automatsko testiranje e-mailova (jesu li poslani, da li su ispravna zaglavlja i tako dalje) (Mailtrap 2021.e)
- Provjera ispravnosti HTML-a i CSS-a za različite klijente i preglednike (Mailtrap 2021.c)
- Lažni SMTP poslužitelj (Mailtrap 2021.b)
- Mailtrap API (Mailtrap 2021.a)

Od navedenih usluga, korišten je lažni SMTP poslužitelj za jednostavno testiranje slanja e-mail poruka. Iako XAMPP dolazi sa svojim ugrađenim poslužiteljem, njegovo podešavanje je kompleksnije dok Mailtrap nudi vrlo jednostavnu i pristupačnu okolinu za rad s e-mail porukama.

Mailtrap je plaćena usluga no nudi besplatnu upotrebu sa nekim ograničenjima kao što je maksimalni dopušteni broj poruka na mjesec, ograničenja na broj poruka u sekundi, količinu sandučića i poruka unutar sandučića i tako dalje (Mailtrap 2021.d).

Na slici 5 prikazan je testni sandučić.



Slika 5: Mailtrap sandučić

Prilikom pregledavanja poruka mogu se vidjeti zaglavlja, verifikacija HTML i CSS sadržaja kao i način na koji bude poruka prikazana ovisno o tipu uređaja (stolno računalo, tablet, mobitel).

3. Korišteni jezici i okviri

3.1. PHP

PHP (engl. *Hypertext Preprocessor*) skriptni je interpreterski jezik čija je svrha programiranje web aplikacija na poslužiteljskoj strani. PHP se može kako ugraditi direktno u HTML kôd i izvršavati u vlastitim skriptnim datotekama, tako i koristiti u kombinaciji sa sustavima predložaka kao što su Dreamweaver, Flash, sustavima za upravljanje web sadržajem (tzv. CMS sustavi, engl. *Content management system*), ali i u sklopu različitih web razvojnih okvira (npr. Laravel, Symfony, Phalcon).

3.1.1. Povijest

PHP je 1994. godine osmislio Rasmus Lerdorf, isprva u obliku nekolicine CGI (engl. *Common Gateway Interface*) programa napisanih u jeziku C, koje je koristio za održavanje svoje osobne web stranice (koje su u to doba interneta bile popularne). Prije svega, svrha mu je bila praćenje pregleda svog online životopisa te ju je zvao "Personal Home Page Tools", odnosno "PHP Tools" (PHP Grupa 2021.b).

FI/PHP Construction Kit Krajem 1995. godine, Rasmus proširuje jezik sa podrškom za varijable u stilu Perl-a, automatsku interpretaciju podataka iz obrazaca i mogućnost direktne ugradnje PHP sintakse u HTML dokumente. Također, (privremeno) izbacuje riječ "PHP" iz naziva te tako ostaje samo naziv "FI". Mjesec dana kasnije, objavljuje se cjelokupno prepisivanje jezika sa novim, privremenim, imenom "Personal Home Page Construction Kit" (PHP Grupa 2021.b).

PHP/FI, PHP2 No, kôd je ponovno dobio još jedno prepisivanje i u travnju 1996. objavljen je "PHP/FI" (Personal Home Page Tools/Forms Interpreter, kombinacija prijašnjih naziva). Sa tom inačicom, PHP postaje više od samo kolekcije alata i pretvara se u vlastiti programski jezik, sa podrškom za sustave baza podataka (mSQL i Postgres95), kolačićima (engl. *cookies*), korisničkim funkcijama i mnogo toga dalje. U lipnju iste godine, PHP/FI-u se daje oznaka 2.0 (PHP Grupa 2021.b).

PHP 3 Iako je PHP/FI 2.0 bio veliko unaprijeđenje prvotnih alata i mogućnosti, ipak su ga mučili problemi poput ne-efikasnosti i nedostatka nekakvih "*enterprise*" funkcionalnosti. Budući da je cjelokupni PHP bio otvorenog kôda, to je omogućilo rad na njemu od strane više suradnika. Dvojica studenata, Andi Gutmans i Zeev Suraski sa Tel Avivskog sveučilišta započeli su tako još jednom cjelokupnu ponovnu izradu programa za sintaksnu analizu (engl. *parser*) za potrebe razvijanja aplikacije za online trgovanje (engl. *e-Commerce*) koje su razvijali za svoj fakultet 1997. godine. Također su kontaktirali samog Rasmusa online kako bi porazgovarali o raznim stranama trenutne implementacije PHP-a i o svojim naporima da ga se prepiše. Rezultat tih razgovora je bio zajednički napor u razvoju novog programskog jezika i okvira (PHP Grupa 2021.b).

U ovoj inačici nalazi se začetak PHP jezika kakav je danas poznat. Također se sa ovom inačicom mijenja ime kako bi se napustila stara konotacija osobne uporabe (Personal Home Page), i naziva se "PHP: Hypertext Preprocessor". Jedna od najvažnijih stvari koje PHP 3.0 donosi je snažna podrška za proširivanje jezika sa vlastitim korisnički razvijenim modulima. U ovoj inačici također se uvodi podrška objektno orijentiranog razvoja (PHP Grupa 2021.b). Prema (Krawczyk 2021.), smatra se da je oko 10% svih web servera u 1998. koristilo PHP 3.0.

U ovo vrijeme PHP također dobiva svoju maskotu, poznatoga slonića zvanog ElePHPant, koju je dizajnirao Vincent Pontier (JetBrains 2021.a).

PHP 4 PHP 3 uveo je mnoge nove funkcionalnosti koje podržavaju razvoj daleko kompleksnijih aplikacija nego što su one prije bile moguće (kao što različite baze podataka, programska sučelja). Kako je PHP bio razvijan poprilično, prema riječima svog autora Lerdorfa, "organički", odnosno bez nekog konkretnog dugoročnog planiranja i predumišljanja, moglo bi se reći da osnova, odnosno jezgra, jezika nije više bila adekvatna da bi zadovoljavajuće podržavala nove mogućnosti (PHP Grupa 2021.b).

Zbog toga je ubrzo nakon izdavanja PHP verzije 3 započet rad na ponovnoj izradi okvira. Cilj ovog puta je bio unaprijediti performanse za kompleksne aplikacije te daljnje proširenje modularnosti PHP-ovog kôda. Rezultat tih napora novi je "pogon" (engl. *engine*) zvan 'Zend Engine', po imenima autora Zeev i Andi. Novi pogon je službeno izdan u svibnju 2000. godine, donoseći sa sobom značajna unaprijeđenja performansi te mnoge nove značajke kao što je podrška za širi raspon web poslužitelja, HTTP sjednice, *output buffering*, sigurnija obrada korisničkog unosa i tako dalje (PHP Grupa 2021.b).

PHP 5 Sredinom 2004. godine izlazi sljedeća inačica PHP jezika koja sa sobom donosi verziju 2 tzv. *Zend engine*-a. Neke od važnih značajki ove verzije su poboljšana podrška za objektno orijentirano programiranje, novo proširenje zvano *PHP Data Objects* (PDO) koje služi kao jednostavno sučelje za razne baze podataka, proširenja za rad sa SOAP protokolom, unaprijeđena inačica MySQL sučelja te poboljšanja performansi kao i druge stvari (JetBrains 2021.a).

Kako bi se potakao prelazak sa PHP 4 na PHP 5 pokrenuta je GoPHP5 inicijativa, koja je do 2008. u potpunosti istisnula PHP 4 iz uporabe.

PHP 5 vjerojatno je najdugovječnija verzija PHP-a koja je uživala najdužu podršku i razvojne cikluse, od svojeg objavljivanja 2004. godine sve do zadnje podržane verzije (ako se gleda raspon datuma na popisu izdanja verzije 5 (PHP Grupa 2021.c). Djelomični razlog tomu je i neuspjeh PHP-a 6, iz čijeg razvoja je naknadno preuzeta te integrirana većina značajki. Kroz svoje glavne razvojne cikluse, PHP 5 preradio je način na koji barata sa datumima i vremenskim zonama, uvodi JSON i ZIP podršku, novi rukovoditelj memorije, imeničke prostore (engl. *namespace*), anonimne funkcije, tzv. *trait*-ovi (način dijeljenja zajedničkog ponašanja među klasama), ugrađeni web server za lakše testiranje, generatore, korutine, ugrađeno proširenje za predmemoriju (engl. *caching*) kao i naravno stalno unaprjeđivanje performansi, razrješavanje grešaka i tako dalje (JetBrains 2021.a).

Izgubljeni PHP 6 Kako je PHP postajao zreo programski jezik i nalazio primjenu u sve širim područjima, određeni nedostaci su postajali sve očitiji. Jedan od glavnih bio je nedostatak podrške za *Unicode* znakove. 2005. započinje rad na implementaciji Unicode podrške, čiji je krajnji cilj ugradnja *International Components for Unicode* biblioteke u jezik kako bi se tekst interno predstavljao po UTF-16 standardu. Takva promjena zahtijevala je značajni reinženjering jezika i njegove implementacija te su odlučili promjene vezane uz to izdati pod novom verzijom PHP 6. No, zbog težine implementacije, nedostatka programera sa potrebnim razumijevanjem kao i poteškoće sa performansama prilikom konverzije iz i u UTF-16 format i visokom potrošnjom memorije odugovlačile su razvoj projekta (Zmievski 2011.).

Sve to konačno je rezultiralo 11. ožujka 2010. godine napuštanjem razvoja PHP 6. Kroz godine, većina glavnih značajki razvijanih u sklopu PHP 6 su bile integrirane u PHP 5 te se tako dalje nastavlja na razvijanju PHP 5 (Lerdorf 2010.).

PHP 6 je bilo visoko iščekivano izdanje o kojem se naširoko pričalo, kako na internetu tako i konferencijama. Tako je u pripremi bila napisana i stavljena u prodaju znatna količina knjiga o PHP 6.

PHP 7 Iako su neuspjesi PHP-a 6 bili razlog dugovječnosti PHP-a 5, prije ili kasnije je trebao doći trenutak kada će se izdati cjelovita nova verzija jezika. Tako je samim krajem 2015. godine izdan PHP 7, najveće izdanje PHP-a u mnogo godina, koje je sa sobom donijelo mnoga unaprjeđenja i novosti koje su PHP učinile zrelijim jezikom. Prije svega treba spomenuti modernizirano rukovanje greškama i iznimkama te uvođenje skalarnih tipova podataka i mogućnost navođenja tipa podatka varijable. Također su uvedeni novi operatori, među kojima jedan zanimljivog naziva *spaceship operator*, te koji se navodi kao '<=>'. U osnovi, taj "svemirski" operator implementira trosmjernu usporedbu kao što se koristi u C jeziku sa funkcijom *strcmp*. Drugi koristan operator koji PHP 7 uvodi je tzv. *null coalesce* operator, koji se navodi kao dva upitnika '??' te je u osnovi kratica za specifičnu uporabu ternarnoga operatora koji vraća vrijednost ukoliko postoji, a u protivnome vraća *null* (PHP Grupa 2021.a) i (Popović i Tomašković 2016.).

No svakako najvažnija i najimpresivnija značajka nove verzije PHP-a je značajno unaprijeđenje performansi jezika uslijed novog izdanja Zend engine-a 3. Po nekim mjerenjima nosi i do dvostruko veću brzinu obrade, što nije nikako za odmetnuti (Popović i Tomašković 2016.).

PHP 7 do danas uživa daljnju podršku te je trenutno (od 11.12.2021.) najnovija inačica 7.4.26.

PHP 8 PHP 8 najnovija je inačica koja je izašla krajem 2020. godine te ponovno donosi mnoge promjene i stoga nije unatrag kompatibilna sa prijašnjim verzijama. Među glavnim novostima je podrška za *JIT* kompilaciju koja u određenim situacijama pruža konkretno ubrzanje rada no uglavnom ne za web stranice kao takve. Ostale istaknute novosti su imenovani parametri za funkcije, "zbirni" (engl. *union*) tipovi podataka (dakle mogućnost navođenja više tipova podataka za jednu varijablu), tzv. *nullsafe* operator (`?->`) koji se služi sigurnom pristupu ugnijezđenim atributima objekata i poboljšano rukovanje greškama (PHP Grupa 2021.d).

3.2. Phalcon PHP

U naslovu rada spominje se izrada društvene mreže u Phalcon PHP-u, no što je točno PhalconPHP? Po nazivu se može zaključiti da je nekako povezan sa PHP jezikom. Radi se o razvojnom okviru za PHP kao što su Laravel i Symfony. No ono po čemu se razlikuje od drugih okvira jest to što je Phalcon pisan u C jeziku kao proširenje za PHP jezik. Kao takvo, radi se o predkompiliranome programu. Prednost takvoga pristupa je u velikoj brzini izvođenja i niskoj potrošnji memorije naspram klasično razvijenih okvira za PHP te sami autori isto navode kao glavnu prednost uporabe. Osim toga navode nekolicinu standardnih značajki koje valja očekivati od svakog zrelijeg okvira, kao što su sigurnost (kriptografska biblioteka, ACL (pristupne liste, engl .*access control list*) funkcionalnosti), vlastiti sustav za predloške (engl. *templating engine*), agnostička podrška za više baza podataka, ORM sloj (objektno-relacijsko preslikavanje, engl. *object-relational mapping*), transakcije, predmemorija (engl. *cache*), internacionalizaciju, ubrizgavanje zavisnosti (engl. *dependency injection*), podršku pisanja REST servisa, itd (Phalcon Tim 2021.a).

PhalconPHP razvija se kao projekt otvorenog kôda, koji je dostupan na repozitoriju `https://github.com/phalcon/cphalcon`.

Iako isprva pisan u C jeziku, kasnije zbog jednostavnosti i veće produktivnosti osmišljavaju novi jezik zvan Zephir, u kojemu nastavljaju razvoj. O Zehpiru se malo više može pročitati u odjeljku 3.2.1.

Trenutno podržane verzije PhalconPHP-a su verzije 3.4 i 4 te se također radi na sljedećoj inačici 5. Naspram verzije 4, verzija 5 će biti izvedena kao čista PHP implementacija (zbog problema sa razvojem Zephira), što također znači da će okvir biti moguće instalirati kao *Composer* (sustav za rukovođenje zavisnostima u PHP jeziku) paket i možda najvažnije od svega, uvodi tzv. ADR (Action Domain Responder) uzorak kao nadomjestak poznatom MVC uzorku. O istome će biti malo više riječi u odjeljku 3.2.2.

U svakom slučaju može se zaključiti da se radi o interesantnom okviru za razvoj, ali koji zbog okolnosti gubi svoju glavnu značajku po kojoj se razlikuje od drugih razvojnih okvira. Povodom istog, kreće se ukratko na brzu usporedbu PhalconPHP okvira sa drugim okvirima na tržištu.

3.2.1. Zephir

Inspiriran C-om i otvorenog je kôda sa repozitorijem na adresi `https://github.com/zephir-lang/zephir`, Zephir je izrađen sa specifičnom svrhom da olakša razvoj PHP proširenja, usredotočivši se na podatkovnu i memorijsku sigurnost (engl. *type safety* i *memory safety*). Nudi mogućnosti dinamičkog i statičkog tipiziranja podataka (engl. *dynamic typing* i *static typing*), što daje potrebnu fleksibilnost programerima. Također se hvale mogućnošću prevođenja ispred vremena (engl. *ahead-of-time compilation*) i ugrađenom funkcijom sakupljanja smeća (engl. *garbage collection*) (Zephir Language Team 2021.b).

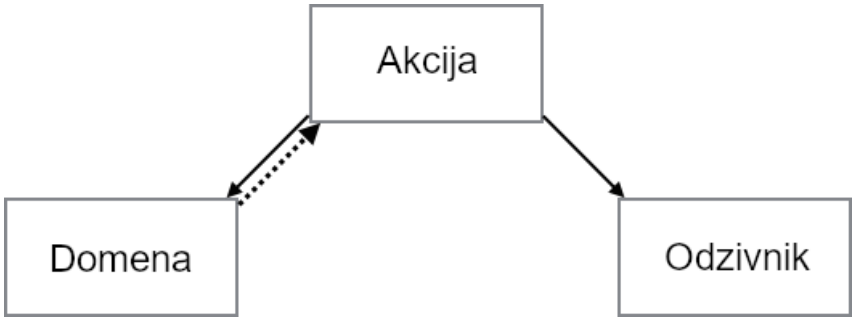
Odgovor na pitanje zašto bi netko trebao koristiti Zephir, ako već postoje jezici poput

C/C++ i PHP-a je upravo u tome da Zephir nastoji objediniti njihove prednosti u jedan jezik. U usporedbi sa PHP jezikom, Zephir je stroži i stoga manje fleksibilan te je tako razvoj u njemu malo sporiji nego u PHP-u. No u usporedbi sa C jezikom, Zephir je siguran (u smislu da ne nudi mogućnost direktnog pristupa memoriji). Program pisan u Zephiru prevodi se u C jezik, koji se onda naknadno prevodi u strojni jezik preko jednog od standardnih C prevoditelja (gcc, clang, vc++). Uz brzinu, to prevođenje pruža dodatnu prednost povećane sigurnost izvornog kôda, budući da ga je potrebno prvo dekompilirati, a onda i analizirati kako bi se shvatilo što on radi (Zephir Language Team 2021.a).

U kolovozu 2020. posljednji razvojni inženjer na Zephir projektu odlazi, što taj projekt ostavlja na mrtvoj točci. To predstavlja problem za daljnji razvoj PhalconPHP-a te se razvojni tim usuglašava da će verzija 6 biti izrađena kao izvorni PHP okvir koji će tako biti moguće instalirati bez tehničkih problema. No, to ipak povlači sa sobom neke nedostatke, prije svega činjenicu da je jedna od glavnih "prodajnih" točaka PhalconPHP-a bila upravo implementacija u obliku PHP proširenja pisanoga u C/Zephir jeziku, što će sa inačicom 6 biti izgubljeno. Korisnici su zabrinuti da će to dovesti do velikih gubitaka performansi, no razvojni tim uvjerava da to nije u cijelosti točno i da se u većini slučajeva upotrebe neće vidjeti primjetna razlika između izvorne verzije i one pakirane kao proširenje (Phalcon Tim 2020.).

3.2.2. ADR

ADR Action-Domain-Responder je softverski arhitekturalni razvojni uzorak kojeg je prvotno predložio Paul M. Jones 2014. godine. Glavna motivacija iza osmišljavanja koncepta stajala je, kako objašnjava, u činjenici da MVC nije uzorak prikladan za web razvoj te dodatno pojašnjava kako MVC nije arhitekturalni uzorak, već uzorak za izradu korisničkog sučelja. Stoga predlaže novi uzorak zvan ADR koji je specifično osmišljen za potrebe web razvoja i HTTP komunikacije (Jones 2021.a).



Slika 6: Dijagram ADR uzorka

Izvor: https://paul-m-jones.com/adr/

Action Dok je u MVC modelu **kontroler** zadužen za programsku logiku koja povezuje podatke modela i pogleda, u ADR modelu je **akcija** (engl. *action*) zadužena za logiku spajanja domene i odzivnika. Dok je **kontroler** obično odgovoran za nekoliko funkcionalnosti vezano za neku domenu, za svaku od kojih implementira zasebnu metodu koja predstavlja nekakvu rad-

nju i obradu, **akcija** predstavlja jednu odvojenu obradu koja živi u svojoj klasi koja implementira jednu metodu (slično nekakvome servisu) (Jones 2020.).

Slično kao što **kontroler** radi sa **modelom**, tako **akcija** radi sa **domenom**, no razlika je da **akcija** ne radi ništa sa **pogledom** niti sa sustavom za predloške (engl. *templating engine*) (Jones 2020.).

Prednost takvog ustroja jest u tome što se time postiže dobro razdvajanje odgovornosti (što je jedno od principa SOLID dizajna). Ukratko, **akcija** je odgovorna samo za sljedeće:

- sabire ulazne podatke sa HTTP zahtijeva
- poziva **domenu** kojoj prosljeđuje ulazne podatke i sabire povratni rezultat
- poziva **odzivnika** (engl. *responder*), proslijeđujući sve potrebne podatke kako bi **odzivnik** izgradio HTTP odgovor i poslao ga natrag (Jones 2020.)

Domain **Domena** je praktički identična po svojoj koncepciji kao i **model** u MVC-u. Glavna razlika nalazi se eventualno u činjenici da **odzivnik** ne manipulira sa **domenom**, osim čitanja podataka u svrhe prikazivanja. Iako su **domena** i **model** načelno isti, razlika u nazivlju motivirana je željom da se konceptualno i asocijativno poveže ADR uzorak sa nekim od uzoraka za *enterprise* programska rješenja (Jones 2020.).

Responder **Odzivnik** Budući da je ADR uzorak za web aplikacije, prikaz korisniku ne sastoji se jednostavno od tijela HTTP odgovora, već čitavi HTTP odgovor predstavlja prezentaciju korisniku (drugim riječima, zaglavlja odgovora, statusni kôd, pa tijelo i sve ostalo). Stoga bi bila loša razdioba odgovornosti kada bi **kontroler** (odnosno **akcija**) bila odgovorna za izgradnju povratnog odgovora korisniku (Jones 2020.).

Stoga svaka **akcija** poziva **odzivnika** koji je odgovoran za izgradnju specifičnog HTTP odgovora, uključujući njegova zaglavlja, kolačiće, status itd. **Odzivnik** se naravno može služiti nekakvim sustavom za preglede (engl. *views*) (Jones 2020.).

Hodogram Ukratko, radni tok od zahtjeva do odgovora u ADR uzorku ide na sljedeći način:

1. Rukovatelj web zahtjeva zaprima HTTP zahtjeva i prosljeđuje ga **akciji**
2. **Akcija** poziva **domenu**, koja sabire sve ulazne podatke iz HTTP zahtjeva
3. **Akcija** poziva **odzivnika**, sa podacima koji su mu potrebni za izgradnju HTTP odgovora (obično to bude HTTP zahtjeva i rezultati **domene**)
4. **Odzivnik** izgrađuje HTTP odgovor sa podacima koje mu je proslijedila **akcija**
5. **Akcija** dobiva odgovor od **odzivnika**, kojeg onda prosljeđuje web rukovatelju koji šalje HTTP odgovor (Jones 2021.b)

3.2.3. Usporedba sa drugim okvirima

Opsežnu i značajnu usporedbu okvira teško je napraviti, niti je takva usporedba cilj ovog rada. Važno će biti samo prikazati okvirno razlike između odabranog okvira (PhalconPHP) i drugih većih i popularnijih okvira koji su danas u uporabi. U te svrhe, bit će prikazano par jednostavnih tehničkih usporedbi performansi okvira. Uz tehničke usporedbe, bit će prikazane i neke općenitije usporedbe po pitanjima kao što su popularnost i rasprostranjenost uporabe.

3.2.3.1. Tehničke usporedbe

Prokofyeva i Boltunova sa tehničkog sveučilišta u Rigi provele su kratko istraživanje koje uspoređuje performanse PhalconPHP-a sa Symfony okvirom. Treba napomenuti, budući da rad potječe iz kraja 2016. godine, da je korištena verzija 3 PhalconPHP okvira te kao takva ne može predstavlja u potpunosti stanje t. Usporedba performansi je provedena na temelju četiri jednostavne metrike pomoću alata *Apache Benchmark* (Prokofyeva i Boltunova 2017.).

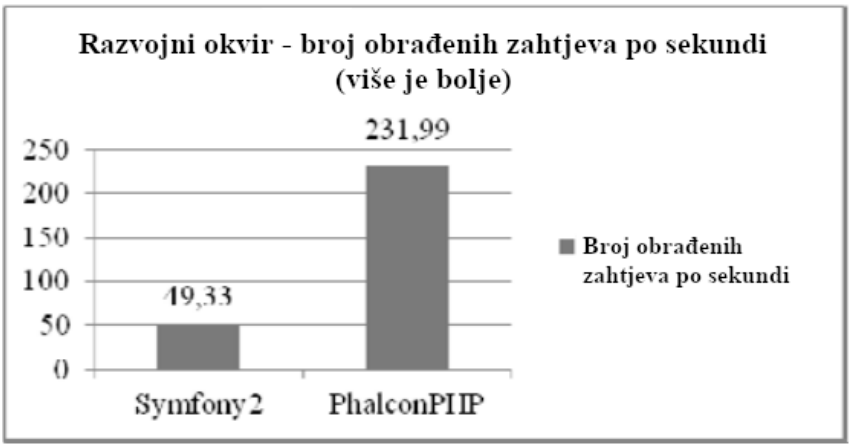
Tim koji stoji iza PhalconPHP-a je također odradio pokoju usporedbu performansi njihovog okvira sa drugim popularnim PHP okvirima. Nažalost, radi se o usporedbi za verziju 3 okvira te je tako ona također malo zastarjela (Phalcon Tim 2017.).

Sljedeće metrike su mjerene i prikazane u tim usporedbama:

- Zahtjevi u sekundi
- Vrijeme po zahtjevu
- Količina
- Uporaba memorije

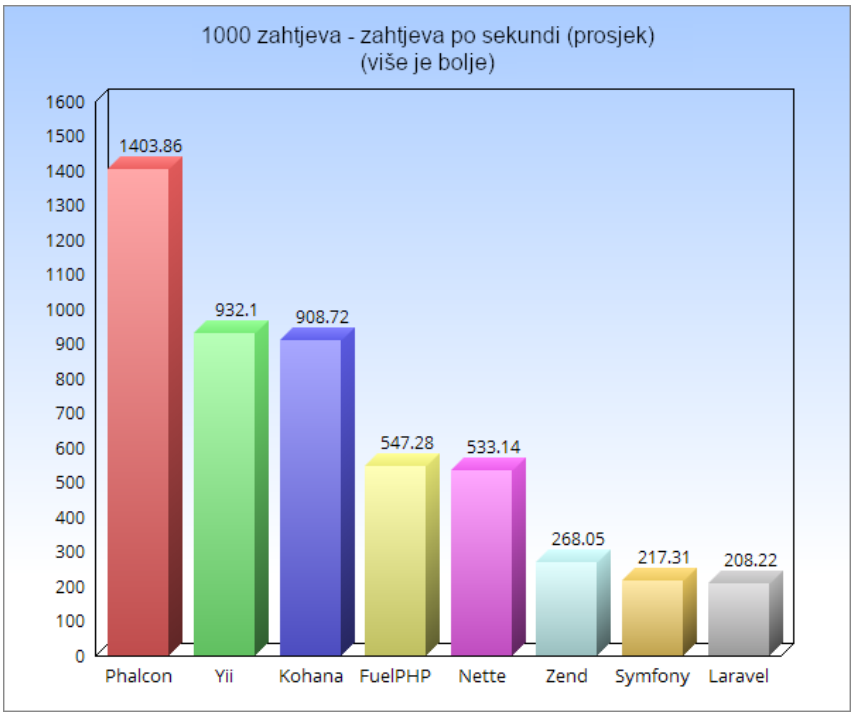
3.2.3.2. Zahtjevi u sekundi

Što se više zahtjeva može obraditi u jedinici vremena, to bolje. Pogotovo je važno kod aplikacija sa velikim brojem istovremenih korisnika, kako bi im se zahtjevi na vrijeme i bez zastoja obradili.



Slika 7: Broj zahtjeva koje okvir u prosjeku obrađuje u sekundi

Izvor: (Prokofyeva i Boltunova 2017.)



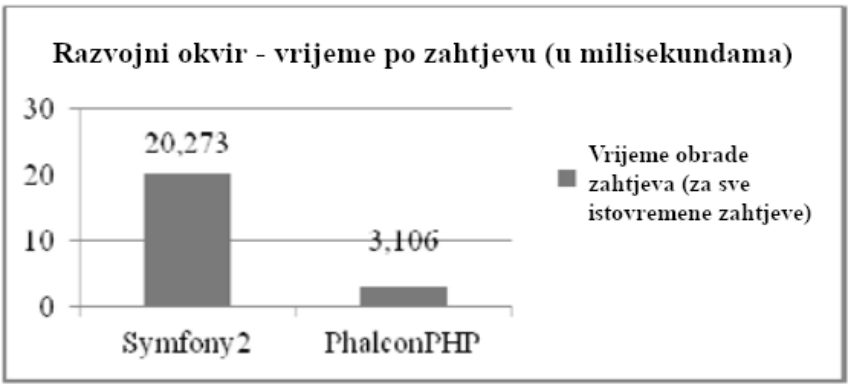
Slika 8: Broj zahtjeva koje okvir u prosjeku obrađuje u sekundi

Izvor: (Phalcon Tim 2017.)

Iz slika se vidi da PhalconPHP može obraditi značajno veću količinu zahtjeva u jedinici vremena (do 7 puta više).

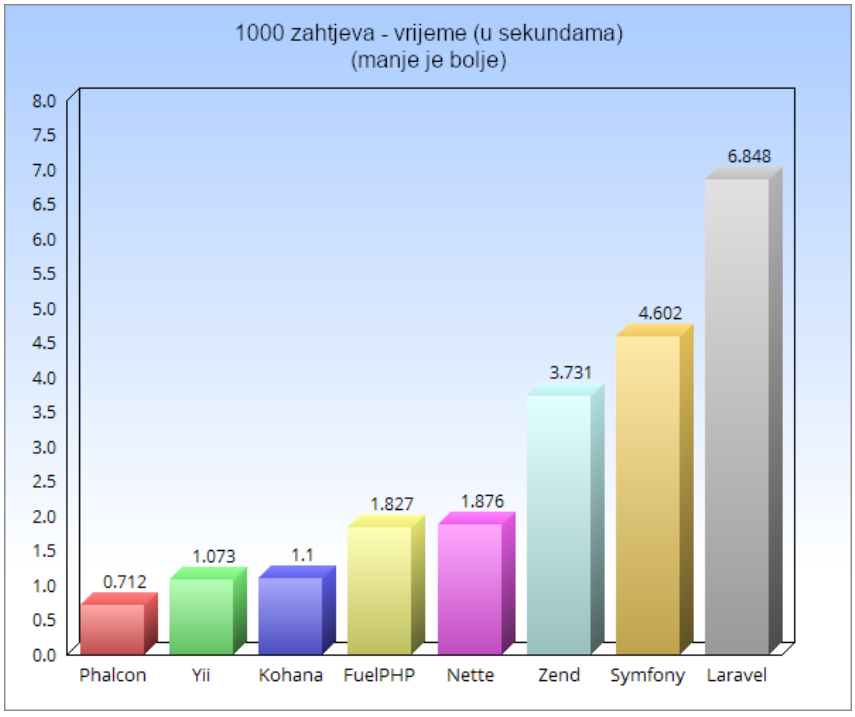
3.2.3.3. Vrijeme obrade zahtjeva

Što se više zahtjeva može obraditi u jedinici vremena, to je kraće vrijeme potrebno za obradu jednog zahtjeva. Metrike su inverzno povezane i ugrubo predstavljaju istu stvar. No autori rada su ipak odlučili prikazati ovu metriku.



Slika 9: Prosječno potrebno vrijeme za obradu jednoga zahtjeva

Izvor: (Prokofyeva i Boltunova 2017.)



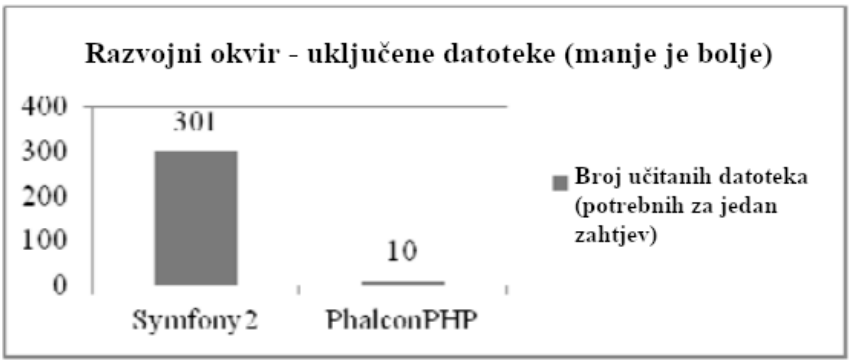
Slika 10: Prosječno potrebno vrijeme za obradu jednoga zahtjeva

Izvor: (Phalcon Tim 2017.)

Naravno, budući da je poznato koliko zahtjeva u sekundi može obraditi PhalconPHP, a koliko drugi okviri, nimalo ne čudi iznad prikazani rezultat.

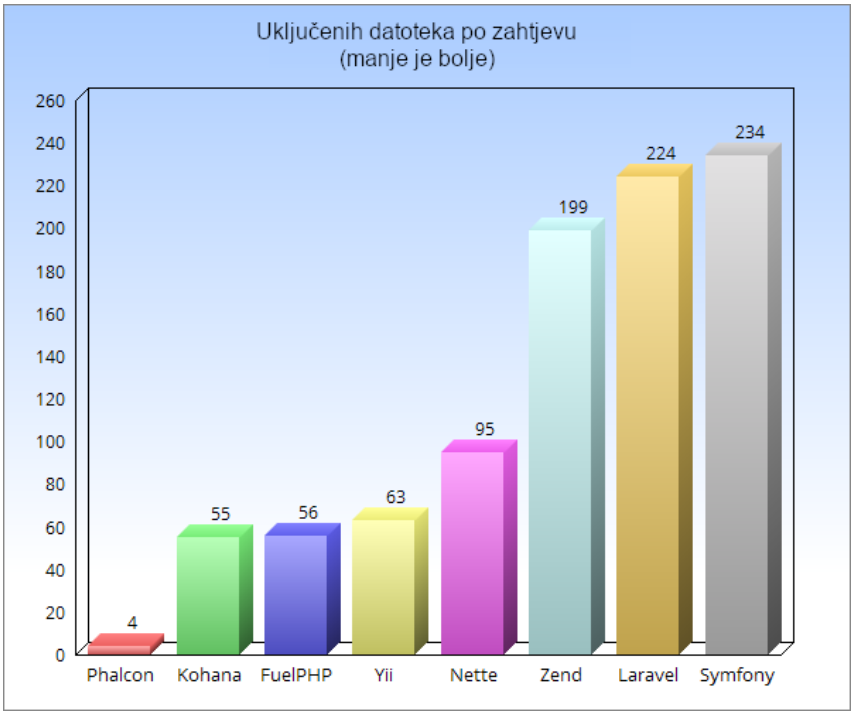
3.2.3.4. Uključene datoteke

Kroz obradu korisničkog zahtjeva, poslužiteljski kôd će uključiti različitu količinu datoteka. Broj datoteka koje će biti uključene ovisi jednim dijelom o razvojnom okviru, ali također i o razvojnom programeru koji implementira potrebni kôd. No, ako se radi o jednostavnom zahtjevu koji ne vrši nikakvu posebnu obradu i služi isključivo provjeri preostalih uključenih datoteka, može se to uzeti u obzir kao metriku.



Slika 11: Broj uključenih datoteka kroz obradu zahtjeva

Izvor: (Prokofyeva i Boltunova 2017.)



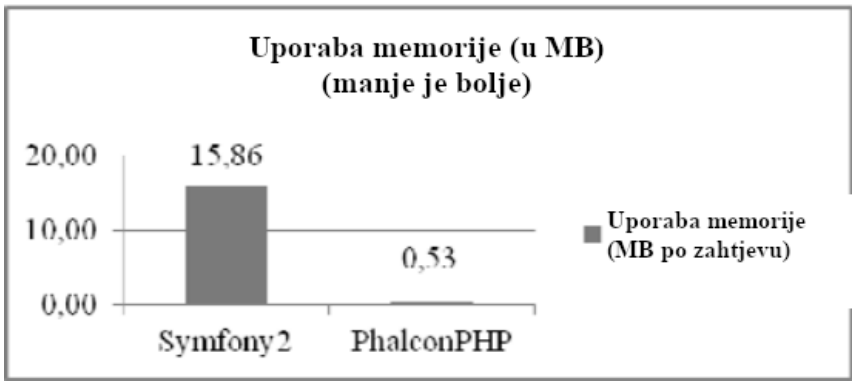
Slika 12: Broj uključenih datoteka kroz obradu zahtjeva

Izvor: (Phalcon Tim 2017.)

Ponovno se da iščitati da za obradu jednostavnog zahtjeva PhalconPHP uključuje znatno manje datoteka nego Symfony. No to ponovno ne čudi, budući da je Symfony opsežan i "težak" razvojni okvir.

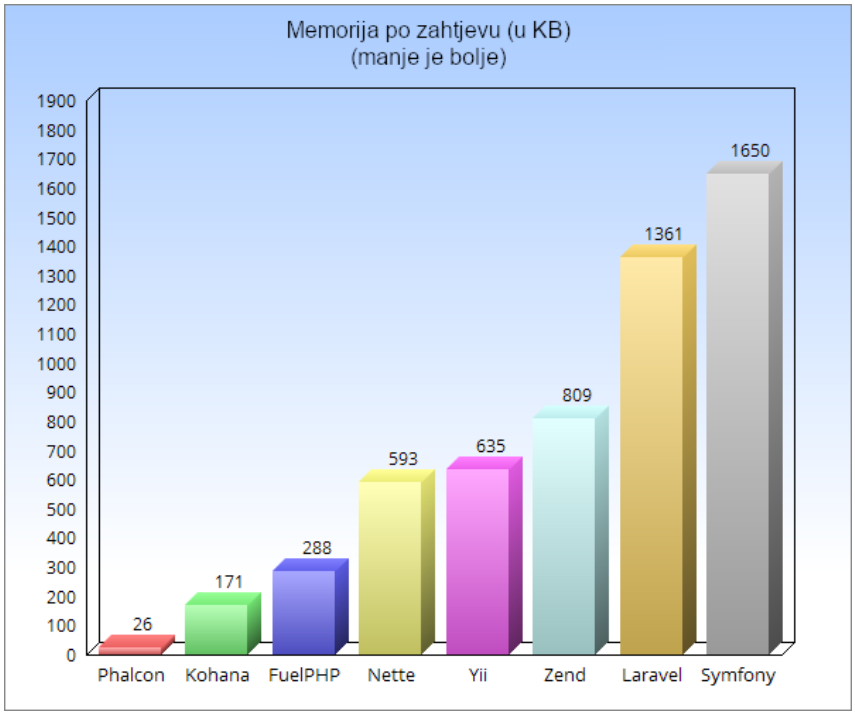
3.2.3.5. Uporaba memorije

Uporaba memorije je uz broj obrađenih zahtjeva u jedinici vremena vjerojatno najvažnija metrika. Što manje računalne memorije zahtjev zauzima, to će se više usporednih zahtjeva moći obrađivati, a da se procesi ili čak čitavi poslužitelj ne sruše.



Slika 13: Prosječna uporaba radne memorije

Izvor: (Prokofyeva i Boltunova 2017.)



Slika 14: Prosječna uporaba radne memorije

Izvor: (Phalcon Tim 2017.)

S obzirom da je PhalconPHP pisan u C jeziku, može se očekivati da će njegov utisak na memoriju poslužitelja biti vrlo nizak. To potvrđuju rezultati istraživanja prikazani u gornjem grafikonu. PhalconPHP koristi čak 30 puta manje poslužiteljske memorije od Symfony-a, što je značajna razlika.

3.2.3.6. Općenita usporedba

No, odabir razvojnog okvira ne ovisi isključivo o njegovim tehničkim karakteristikama, iako su one važne. Vjerojatno su od veće važnosti implementacijske karakteristike okvira, odnosno drugim riječima, što okvir donosi na stol u vidu svojih značajki, arhitekture, podrške za različite sustave i na kraju krajeva, samog iskustva uporabe i razvoja. Primjerice, ukoliko će se za odabrani projekt koristiti PostgreSQL baza podataka, nije preporučljivo odabrati okvir koji nema ugrađenu podršku za taj sustav baza podataka (iako je moguće implementirati vlastite prilagodbe, za potrebe brzog prototipiranja i razvoja nema smisla).

	Laravel	Symfony	CakePHP	Codeigniter	PhalconPHP
Zadnja verzija	8.6.0, 17.08.2021.	5.3.6., 29.07.2021.	4.2.8., 17.07.2021.	4.1.3., 06.06.2021.	4.1.2., 25.04.2021.
Alati za naredbeni redak	Artisan	bin/console	Cake	Spark	Phalcon (dev tools)
Podržane baze podataka	MySQL, PostgreSQL, SQLite, MSSQL	MySQL, PostgreSQL, SQL Server, SQLite, Oracle, MongoDB	MySQL, PostgreSQL, SQLite, MSSQL, Oracle	MySQL, PostgreSQL, SQLite, MSSQL	MySQL, PostgreSQL, SQLite
Podrška za dnevnike (engl. logging)	Da	Da	Da	Da	Da
Podrška za međuspremnike	Da	Da	Da	Da	Da
Migracije	Da	Da	Da	Da	Da
Podrška za asinkrone događaje	Da	Da	Da	Da	Da
Sustav za predloške (engl. view engine)	Blade	Twig	PHP	PHP	Volt
ORM	Eloquent (Active record)	Doctrine (Active record)	Active record, Datamapper	Active record	Active record
Query Builder	Da	Da	Da	Da	Da
Ubrizgavanje zavisnosti	Da	Da	Ne (u razvoju)	Ne	Djelomično
Podrška za testiranje	PHPUnit	PHPUnit	PHPUnit	PHPUnit	PHPUnit
PSR sukladnost	PSR-2, PSR-4, PSR-7, PSR-11	PSR-4, PSR-6, PSR-11	PSR-2, PSR-4, PSR-7,	PSR-1, PSR-2, PSR-3, PSR-4, PSR-6, PSR-7	PSR-0/PSR-4, PSR-3, PSR-7, PSR-11, PSR-13, PSR-17
Popularnost (GitHub)	66.3 tisuća zvijezdica	25.7 tisuća zvijezdica	8.4 tisuća zvijezdica	3.6 tisuća zvijezdica	10.5 tisuća zvijezdica

Tablica 1: Usporedba PHP razvojnih okvira

Iako opsežnija, gornja usporedba i dalje ne daje dovoljno informacija kako bi se pravilno razabrale razlike između okvira i pitanja kojeg odabrati. Prije svega, može se vidjeti da su po mnogim pitanjima okviri istovjetni (dakle uglavnom svi zadovoljavaju standardne značajke). Za odabir okvira važno je iskustvo iz prve ruke, odnosno kakvo razvojno iskustvo ono pruža korisniku. Autor rada nažalost nije u mogućnosti dati iscrpnu usporedbu u tome pogledu budući da, izuzev sa PhalconPHP koji je propisan temom, susreo se sa Laravelom (opsežno) i Codeigniterom (umjereno).

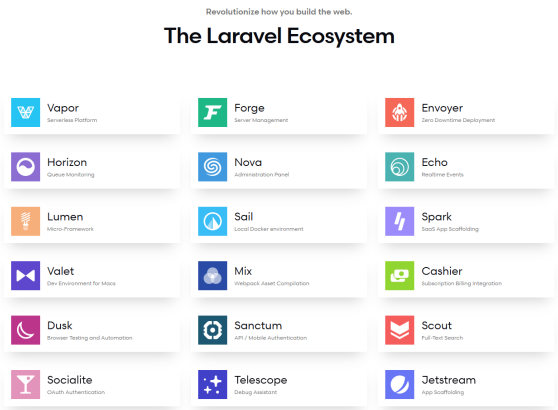
3.2.3.7. Popularnost

Jedan od najvažnijih faktora kod odabira okvira je naravno njegova popularnost. Razlog je jednostavan, popularni okviri imaju veću zajednicu koja može pružati odgovore na pitanja, na uobičajene probleme se nailazilo više puta te za njih postoje često pojašnjenja i rješenja. Nadalje, često korišteni okvir je također "provjereniji", odnosno greške se prije uoče i prije isprave. Po tome pitanju **Laravel** prednjači daleko ispred spomenute konkurencije.

Posljedično prijašnjoj točci, popularni i široko korišteni okviri će imati i bolju podršku *treće strane* (engl. *third-party support*). Različite biblioteke i paketi budu vjerojatnije prilagođeni za rad u tome okviru i vjerojatno će postojati gotovi paketi koji se brinu za kompatibilnost.

3.2.3.8. Ekosustav

"Ekosustav" nekog okvira odnosi se na različite pakete i biblioteke *prve strane* koji pružaju podršku za različite dodatne funkcionalnosti koje nisu nužno dio same jezgre razvojnog okvira. Po tom pitanju ponovno prednjači **Laravel**, čiji razvojni tim je razvio opsežni spisak dodatnih biblioteka za korištenje koje se direktno integriraju u osnovni Laravel okvir.



Slika 15: Laravel-ov ekosustav

Izvor: https://laravel.com/

Većina drugih okvira ne može se hvaliti takvom ponudom. Symfony okvir kao takav se već temeljni na principu modularnosti i razdvajanja komponenti (neke od kojih drugi okviri koriste) te se tako njegove komponente mogu smatrati dijelovima ekosustava. Naspram toga, PhalconPHP, Codeigniter i CakePHP nemaju nekakav opširniji ekosustav koji se gradi oko njih.

3.2.3.9. Alati za naredbeni redak

Kao što se vidi u tablici 1, svi spomenuti okviri podržavaju nekakve alate naredbenog retka, no oni se mogu značajno razlikovati u svojoj uporabi i korisnosti.

Laravel Laravel dolazi s alatom za naredbeni redak zvanim *artisan*. On omogućava jednostavno stvaranje većine različitih komponenti koje se u okviru koriste, kao što su *kontroleri*, *modeli* i *pogledi*, ali i drugih koncepata kojima se Laravel služi kao što su komande za naredbeni redak, komponente za poglede, obavijesti, E-mailovi, slušatelji događaja, resursi, pravila (za validaciju HTTP zahtjeva), HTTP posrednici (engl. *middleware*), pozadinski zadaci, migracije, sjemenje (engl. *seed*, koji služi za predpopunjavanje baze podataka osnovnim zapisima), tvornice (engl. *factory*, što dolazi od istoimenog uzorka dizajna), namjenske iznimke, događaji, zahtjevi, poslužitelji (engl. *provider*) i testove. Uz to sadrži naredbe za rad s međuspremnikom (engl. *cache*), migracijama, događajima i redovima (za izvršavanje pozadinskih zadatka).

CakePHP Alat za naredbeni redak koji dolazi upakiran s CakePHP okvirom maštovito se zove *cake* te nudi jednostavne naredbe za izradu kontrolera, modela, pogleda kao i posrednika, dodatnih komandi, testova, pomoćnika (engl. *helper*), obrazaca, e-mail poruka i umjetnih podataka za testiranje (koji se u ovom slučaju zove na engl. *fixtures*). Također omogućuje generiranje ruta za navigaciju, rad s međuspremnikom, jezičnim prijevodima (odnosno internacionalizacijom) i migracijama.

PhalconPHP PhalconPHP nudi pomoćni razvojni alat za naredbeni redak naziva *Phalcon DevTools*. Naspram prije spomenutih okvira i alata, DevTools je nešto siromašnijih mogućnosti te tako nudi samo opcije za kreiranje kontrolera, modela, modula (ukoliko se radi o višemodulskoj aplikaciji), novoga projekta i nove migracije.

Symfony Symfonyjev alat za naredbeni redak se jednostavno zove symfony (ili *console*, ako se gleda naziv izvršnog programa). Po opsegu sličan Laravelovom (dapače, Laravel se služi mnogim komponentama Symfony okvira, stoga sličnost ne treba čuditi) i tako naravno nudi mogućnost kreiranja kontrolera i modela, ali ne i pogleda. Uz to podržava kreiranje migracija, obrazaca, umjetnih podataka za testiranje (prije spomenutih *fixtures*), komandi za naredbeni redak i validatora za podatke. Također nudi komande za jednostavnu šablonsku izradu registracijskog obrasca (s pogledom, kontrolerom i svime vezanime) kao i za resetiranje lozinke. Dodatno nudi komande za rad s prijevodima, međuspremnikom i s doctrine ORM-om (ORM je skraćenica za *object relational mapper* te predstavlja programski sloj koji reprezentira podatke iz baze podataka).

Codeigniter Za kraj, alat za naredbeni redak koji dolazi upakiran sa Codeigniter okvirom zvan *Spark*, koji je također temeljen na Symfonyjevim bibliotekama, donosi sa sobom mogućnosti za izradu kontrolera i modela, ali ne i pogleda. Uz to nudi komande za rad sa migracijama, za ispis pomoćnih podataka (definirane rute, provjeru imeničkih prostora i lokalno serviranje aplikacije

putem ugrađenog PHP servera), kao i komande za kreiranje novih konfiguracijskih datoteka, komandi, tzv. *filtera* i validacijskih pravila.

3.2.3.10. Dokumentacija

Razvojni okvir je onoliko dobar koliko se osoba njime zna služiti. Svaki okvir nudi vlastitu dokumentaciju u kojoj pojašnjava osnovne principe, prihvaćene konvencije i specifičnosti tog okvira. Nije jednostavno napisati kvalitetnu i čitku dokumentaciju koja pokriva sve glavne točke na intuitivan način bez da opterećuje korisnika nepotrebnim informacijama.

Slijedi, po autorovom osobnom mišljenju, poredani popis programskih okvira po kvaliteti njihove dokumentacije:

- Laravel
- Codeigniter
- PhalconPHP
- CakePHP
- Symfony

Doduše, treba uzeti u obzir da su PhalconPHP i CakePHP po kvaliteti dokumentacije bliski te da autor smatra da je Symfony svakako najlošije dokumentiran iz jednostavnoga razloga što je daleko najmanje intuitivan i jednostavan za navigaciju i pronalazak potrebnih informacija. Dok s druge strane Laravel nudi jednostavnost čitanja i lakoću pretraživanja te pronalaska potrebnih stavki.

3.2.3.11. Jednostavnost

Neki okviri su po svojoj prirodi jednostavniji za usvajanje i uporabu nego drugi. Ovisno o zahtjevima projekta i dostupnom vremenu, možda će korisniku moći od veće važnosti biti brzo usvojiti okvir negoli koje sve mogućnost on njemu nudi. Po tom pitanju svakako prednjači Codeigniter koji se u neku ruku predstavlja više kao set alata (engl. *toolkit*) negoli razvojni okvir te uz svoju jednostavnu i čitku dokumentaciju dozvoljava vrlo brzo usvajanje. Ako je korisniku važno brzo krenuti s projektom, onda je Codeigniter svakako dobar odabir.

S druge strane, Symfony je okvir koji se ne preporučuje neiskusnim programerima zato što nudi napredne i kompleksne razvojne komponente koje zahtijevaju dobro poznavanje kako PHP-a tako i Symfonyja (Brotherton 2021.).

Između Codeignitera i Symfonyja nalaze se po redu jednostavnosti okviri Laravel, koji uz kvalitetnu dokumentaciju i relativnu jednostavnost nudi moćan okvir te tako pogađa jedan balans zbog kojeg, uostalom, i jest jedan od najpopularnijih okvira u uporabi danas. Iza Laravela, slijedi CakePHP te naposljetku PhalconPHP.

3.2.3.12. Zaključak

Prema gore navedenim kriterijima, generalni zaključak postavlja da je Laravel načelno preferirani izbor ukoliko se kreće u izradu novog projekta, zbog svoje popularnosti, kvalitetne dokumentacije, opsežnog ekosustava te dobrog omjera jednostavnosti i moći.

3.2.4. Instalacija PhalconPHP okvira

Preduvjeti PhalconPHP zahtijeva PHP instalaciju minimalne verzije 7.2 te također instalirano PSR proširenje, koje je dostupno na `https://github.com/jbboehr/php-psr`. PSR stoji za *PHP Standards Recommendations* te predstavlja službeno definirane standarde za PHP kôd. Spomenuto proširenje koje je potrebno skinuti kako bi se koristio PhalconPHP je naprosto zbir programskih sučelja svih uvaženih PSR standarda na jednome mjestu. Također će biti potrebno u svojoj **php.ini** konfiguraciji učitati PSR proširenje:

```
extension=psr.so
```

Također, budući da je PhalconPHP okvir agnostičan prema bazama podataka koje se koriste pri izradi aplikacije, potrebno je instalirati i PHP-ovo **PDO** proširenje (php_pdo).

S programske strane, neki od preduvjeta su da sustav koji će posluživati aplikaciju ima instaliranu podršku za sljedeća proširenja i biblioteke:

- curl
- fileinfo
- gettext
- gd2
- imagick
- json
- pcre
- OpenSSL
- Mbstring
- Memcached/Redis

Linux Budući da se PhalconPHP distribuira kao PHP proširenje, njegova instalacija je nešto drugačija od uobičajene *composer install* metode na koju su PHP programeri naviknuti. Za Linux sustave, ukoliko PhalconPHP nije već dostupan, potrebno je dodati repozitorij koji sadrži paket. Za *deb* temeljene pakete (Debian, Ubuntu), potrebno je unijeti sljedeće:

```
curl -s https://packagecloud.io/install/repositories/phalcon/stable/script.deb.s

sudo apt-get update
sudo apt-get install php7.2-phalcon
```

Za pakete temeljene na *rpm*-u (Red hat, Fedora) potrebna je mala preinaka:

```
curl -s https://packagecloud.io/install/repositories/phalcon/stable/script.rpm.s

sudo yum update
sudo yum install php72u-phalcon
```

Te je naravno potrebno navesti:

```
extension=php_phalcon.so
```

Mac OS Za instalaciju Phalcona na Mac sustavima, potrebno je prije toga imati instalirani homebrew upravitelj paketa i jednostavno unijeti sljedeću komandu u naredbeni redak:

```
brew tap phalcon/extension https://github.com/phalcon/homebrew-tap
brew install phalcon
```

Te je naravno potrebno navesti

```
extension=php_phalcon.so
```

Windows Za Windows operacijski sustav instalacija je jednostavnija te je potrebno samo preuzeti predkompiliranu .dll datoteku te ju navesti u php.ini konfiguracijskoj datoteci:

```
extension=php_phalcon.dll
```

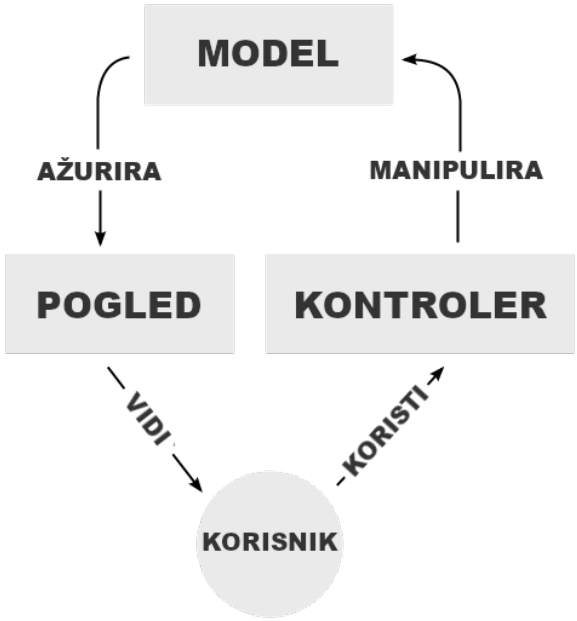
Time je instaliran okvir i spreman je za uporabu. No, prije nego se započne, preporuča se instalacija pomoćnog alata za razvoj u PhalconPHP okviru, tzv. *Phalcon DevTools*. Oni se lako instaliraju preko *composer* alata, unosom:

```
composer global require phalcon/devtools
```

Ako se ne želi imati globalno dostupne alate, nego alate samo za dotičan projekt, onda se zastavica *global* izostavlja.

3.2.5. MVC

MVC stoji za *Model-View-Controller* i predstavlja jedan od najcitiranijih i, kao što je (Jones 2020.) spomenuo, najviše krivo shvaćenih uzoraka dizajna u programiranju. Prema (Trygve 1979.) osmišljen je u Xerox Parc-u (*Palo Alto Research Center*) krajem 70-ih od strane Trygve Reenskauga za potrebe Smalltalk-80 jezika. U osnovi, MVC je osmišljen kao općenito rješenje za izazov korisnika koji upravlja velikim i kompleksnim skupovima podataka. Temeljna svrha mu je premostiti jaz između ljudskog mentalnog modela i digitalnog modela unutar računala. Drugim riječima, idealni MVC sustav, u svojoj originalnoj koncepciji, podupire korisničku iluziju direktnog manipuliranja domenskim informacijama u sustavu (Trygve 1979.).



Slika 16: Opisni dijagram MVC uzorka

Izvor:
https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Također, kao što (Model 2014.) navodi, MVC se smatra jednim od prvih uzoraka koji se u svom konceptu bave razdvajanjem odgovornosti unutar programskog kôda. Izvorno osmišljeno, paradigma MVC razdvaja korisnički unos, modeliranje vanjskog svijeta i vizualne povratne informacije u razdvojena i zasebno obrađivana područja. Ta tri područja nose nazive, kao što se može iščitati iz samog imena uzorka, su *Model*, *View* (odnosno *Pogled*) i *Controller* (odnosno *Kontroler*) (Burbeck 1992.).

Model Podatkovna preslika stvarnoga svijeta. Model je nositelj domenskih podataka. Svaka točka koja je korisniku vidljiva u sučelju predstavljena je nekakvim modelom. Model mora biti agnostičan što se tiče prezentacije, odnosno ne smije ni na koji način svojom strukturom diktirati kako će se podaci prezentirati korisniku, jer treba biti moguće za isti model izgraditi više prezentacija (primjerice, korisnika možda zanima iste podatke vidjeti na drugačiji način).

(Späth 2021., Burbeck 1992.).

Neki autori dijele model na dvije podvrste - pasivni i aktivni.

Pasivni Model je isključivo nositelj podataka potrebnih za prikaz korisniku. Odgovornost za obavještavanje o promjenama u podacima snosi kontroler u svojoj komunikaciji s pogledom. Ukoliko kontroler primijeti da su se kroz korisnikov unos podaci ažurirali, kontroler će obavijestiti pogled o potrebi za osvježavanjem (Burbeck 1992.).

Aktivni No, postoje situacije kada model treba snositi više odgovornosti te ne može biti jednostavno pasivan. Ukoliko model ne biva izmijenjen isključivo od strane korisničkog unosa, odnosno u odnosu na kontroler ili model, onda je potrebno da model bude u mogućnosti obavijestiti druge dijelove sustava o svojim promjenama, jer oni inače ne budu svjesni tih promjena. Takve promjene mogu doći od npr. nekog trećeg vanjskog sustava. Ovakva vrsta modela općenito mora implementirati dodatne funkcionalnosti kojima će moći pratiti listu objekata koji ovise o njegovim promjenama te ih mora moći obavijestiti (Burbeck 1992.).

Pogled Odgovoran za prezentaciju podataka korisniku. Uključuje kontrolne elemente (polja za unos, gumbi, kvadratići na križanje, izbornici, i tako dalje). Može postojati više pogleda za iste podatke, primjerice uređeni popis i tablica. U širem arhitekturalnom smislu, pogled mogu biti i različite tehnološke implementacije kao stranica u web pregledniku i aplikacija na mobilnome uređaju (Späth 2021.).

Kontroler Obrađuje korisnički unos i priprema skupove podataka za prikaz korisniku. Sadrži programsku, odnosno poslovnu logiku aplikacije te se sve kalkulacije i transformacije podataka događaju u sklopu kontrolera (Späth 2021.).

Sredinom 90-ih godinama, uporaba MVC uzorka za web aplikacije je eksplodirala u popularnosti, nakon uvođenja NeXT-ovog WebObjects. Nakon toga MVC postaje popularan među Java programerima, koji prebacuju WebObjects na Java platformu. Kasnije se javljaju drugi okviri kao što su Spring koji nastavljaju sa naglaskom na MVC strukturu. Kroz godine se pojavljuju različiti specijalizirani okviri za MVC na različitim jezicima kao što su Django i Pyramid za Python ili Rails za Ruby te ga oni uvode za potrebe projekata manje kompleksnosti.

U kontekstu web aplikacija, par stvari se mijenja prilikom uporabe MVC uzorka. Budući da je HTTP protokol bez ikakvog stanja, identifikacija korisnika se provodi dodatnim mehanizmima kao što su sjednice (engl *session*) ili kolačićima (engl. *cookies*). Uobičajeno je da odabrani MVC okvir podržava nekakav sustav za uzorke koji generira korisnički pogled te se potrebni podaci iz modela samo ubacuju na ključna mjesta unutar uzorka. Nadalje, postavlja se pitanje kako osvježiti pogled ukoliko dođe do promjene u podacima. Jedan pristup je ponovno generirati cijelu stranicu na poslužiteljskoj strani i ponovno ju prikazati. Druga mogućnost je dinamički ažurirati samo one dijelove stranice, odnosno pogleda, koji su izmijenjeni (Späth 2021.).

Kako je MVC uzorak postajao sve popularniji i raširenije uporabe te budući da se mogao po nekim pitanjima slobodnije interpretirati, u toj raznolikoj uporabi razvijale su se i njegove varijante:

- Model-View-Presenter (MVP) - Svrha MVP uzorka je dotaknuti se nekih nedostataka MVC uzorka. Glavni elementi uzorka više nisu organizirani u "trokut" te se izbacuje kontroler u korist tzv. *prezentera* (engl. *presenter*). U ovoj verziji, pogled nije svijestan postojanja modela, već prezenter ima ključnu ulogu prijema korisničkog unosa iz pogleda i mapiranja tih podataka između pogleda i modela, pritom vršeći poslovnu logiku. U ovakvom pristupu, prezenter se najčešće prvi dizajnira (Kouraklis 2016., Almiray 2015.).
- Model-View-ViewModel (MVVM) - Daljnja alternativa MVC i MVP uzorcima, uvedena za Smalltalk-80 pod nazivom *Application model* ili *Presentation model*. Umjesto Kontrolera (ili prezentera) uvodi se PogledModel (engl. *ViewModel*) te se uloge i odgovornosti Pogleda i PogledModela mijenjaju naspram standardnih uzoraka. PogledModel suštinski predstavlja model samog pogleda i tako je odgovoran za njegovo stanje kao i za ponašanje ovisno o korisničkome unosu. PogledModel odgovoran je za komunikaciju između Modela i Pogleda te vrši potrebnu validaciju proslijeđenih podataka. Dodatna novost je postojanje *Veznika* (engl. *Binder*) koji usklađuje stanje podataka između Pogleda i PogledModela, što rezultira automatskim osvježavanjem podataka kada dođe do njihove promjene. Prvenstveno, MVVM uzorak nastao je iz potrebe da se olakša testiranje komponenti (Kouraklis 2016., Almiray 2015.).
- Hierarchichal model-view-controller (HMVC) - Varijanta MVC uzorka koja, suštinski, organizira aplikaciju u više slojeva manjih MVC aplikacija odnosno modula. Takva organizacija može biti korisna pri kompleksnijim aplikacijama kod kojih različiti dijelovi mogu biti prisutni kroz više stranica ili više njih bude prisutno na istoj stranici. HMVC se može smatrati "widget-izacijom" aplikacije. Kao primjer može se uzeti prikaz komentara ili neki izvor/kanal (primjerice RSS ili Twitter) koji se mogu vidjeti na više mjesta u jednoj aplikaciji ili čak kroz više različitih aplikacija (Vance 2011., Cai, Kapila i Pal 2000.).

Spomenutih varijanti postoji još više, no nema smisla zadržavati se na njima niti davati iscrpne popise, budući da to nije tema rada. Opisan je poslužiteljski dio rabljenih tehnologija te je vrijeme premjestiti se na klijentsku stranu, počevši sa Angularom.

3.3. Angular

Angular je programski okvir za razvoj web aplikacija na korisničkoj strani (engl. *frontend*) temeljen na TypeScript jeziku (više o njemu u zasebnom odjeljku 3.3.2), te se vodi principom izrade tzv. *jednostraničnih aplikacija* (engl. *s*ingle page application, skraćeno *SPA*). Danas iznimno popularan i rasprostranjen, nudi dobro i pojednostavljeno iskustvo razvoja web sučelja. Neke od istaknutih značajki Angular-a su:

- Podrška za više platformi

  PWA (progresivne web aplikacije, engl. *progressive web apps*)

  Izvorno mobilno

  Desktop

- Brzina i visoke performanse
- Brza izrada sučelja pomoću predložaka
- Alati za naredbeni redak
- Podrška za mnoge IDE programe
- Ugrađena podrška za testiranje

Prije nego se krene u daljnje razmatranje Angulara, slijedi kratak paragraf koji pojašnjava što su jednostranične aplikacije uopće.

SPA Prema (Skólski 2018.) i (MDN 2021.), SPA je vrsta web aplikacije koja nakon inicijalnog učitavanja izbjegava bilo kakvo osvježavanje preglednika te tako stvaraju dojam jedne cjeline, odnosno jedne stranice kojoj se sadržaj dinamički mijenja. Isto postiže putem JavaScript sučelja kao što su *XMLHttpRequest* ili *Fetch*, koji dohvaćaju potrebne promjene sa pozadinskog poslužitelja. Primjeri takvih aplikacija su poznati iz svakodnevne upotrebe, kao što su: *Gmail*, *Facebook*, ili *GitHub*. Neki popularni JavaScript okviri koji se vode SPA principom su:

- Angular
- Ember.js
- Meteor.js
- Vue.js
- React

Dalje o Angularu Osnovna koncepcija Angulara temelji se na izradi komponenata i njihovim hijerarhijama. Komponenta čini osnovnu gradbenu jedinicu Angular web aplikacije. Svaki dio sučelja, odnosno web aplikacije, izvodi se pomoću komponente. Uz komponente, postoje moduli, servisi i direktive. Slijedi kratko pojašnjenje što svaki od tih elemenata predstavlja.

Modul Kompleksne aplikacije dodiruju se različitih domena te svoj kôd trebaju dijeliti po odgovornosti i zajedničkim funkcionalnostima. Modul predstavlja takvu jednu cjelinu programskog kôda, komponenti i servisa koji se dodiruju određene značajke ili skupa značajki unutar aplikacije, neke specifične domene ili nekakvog tijeka rada unutar aplikacije (Angular Tim 2021.d).

Komponenta Kao što je spomenuto, komponenta je temelja gradbena jedinica aplikacije i njen zadatak je upravljanje nad nekim pogledom. Ona unutar sebe zaokružuje, odnosno enkapsulira, HTML sučelje, CSS dizajn i TypeScript programsku logiku te joj se pridaje jedinstveni identifikator kojim se može kasnije navoditi u HTML-u. U načelu, komponenta bi se trebala baviti samo prikazom podataka, dok za rad sa podacima služe *servisi*. Svaka klasa može postati komponenta ukoliko ju se anotira sa oznakom *@Component* i navedu potrebni metapodaci, koji su *selector* i *template* odnosno *templateUrl* (Angular Tim 2021.c).

```
@Component({
    selector: 'app-primjer',
    templateUrl: './primjer.component.html'
})
export class PrimjerComponent implements OnInit {
    @Input() id: number;

    ime: string;
    prezime: string;
    dob: number;

    constructor(private dohvatiKorisnikaServis: DohvatiKorisnikaServis) {}

    ngOnInit() {
        const osoba = this.dohvatiKorisnikaServis.dohvati(this.id);

        this.ime = osoba.ime;
        this.prezime = osoba.prezime;
        this.dob = osoba.dob;
    }
}
```

Izvorni kôd 3.1: Primjer komponente

Selektor je ime oznake kojom se može unutar HTML predloška navesti komponenta. Pritom je moguće potrebne ulazne podatke proslijediti komponenti. Oni se navode unutar kôda komponente putem @Input anotacije. U kôdu se onda navode kao obični atributi ako je fiksna vrijednost ili unutar uglatih zagrada ako se radi o izrazu ili dinamičkoj vrijednosti (Angular Tim 2021.c).

```
<app-primjer [id]="2+2"></app-primjer>
```

Izvorni kôd 3.2: Uporaba izraza kao proslijeđene vrijednosti

Servis Servis, kao što njegovo ime to implicira, odrađuje određeni zadatak. Najčešće će služiti za dohvat i obradu podataka (primjerice REST API pozivom) te za komunikaciju među komponentama (budući da se mogu inicijalizirati i biti dostupni u globalnom prostoru) (Angular Tim 2021.e).

```
export class DohvatiKorisnikaServis {
    private osobe: Osoba[] = [];

    constructor(
        private backend: BackendServis,
    ) { }

    dohvati() {
        this.backend.dohvatiKorisnike().then((osobe: Osoba[]) => {
            this.osobe.push(...osobe); // fill cache
        });
        return this.osobe;
    }
}
```

Izvorni kôd 3.3: Primjer servisa

Kako bi se koristio servis u nekoj komponenti potrebno ga je putem injektiranja zavisnosti ubaciti u klasu komponente:

```
constructor(
    private backend: BackendServis
) { }
```

Izvorni kôd 3.4: Injektiranje zavisnosti servisa

Direktiva Direktive su posebna vrsta klase čija je svrha direktna manipulacija nad elementima u aplikaciji te se navode kao i drugi HTML atributi. Neki od primjera su Angularove ugrađene direktive za uvjetovani prikaz dijela sučelja ili za ispis elemenata u petlji (*ngIf i *ngFor). Potonji primjeri spadaju pod tzv. *strukturalne* direktive koje, kao što im naziv upućuje, uvjetuju ispisivanje HTML elemenata na stranici. Također, postoje atributne direktive koje modificiraju svojstva elementa na kojem je direktiva navedena. Na taj način se mogu dinamički i uvjetno mijenjati svojstva elemenata (Angular Tim 2021.b, Angular Tim 2021.a, Angular Tim 2021.f).

Ako se uzme za primjer direktivu koja podcrtava sadržaj elementa kojem pripada:

```
@Directive({
    selector: appPodcrtano
})
export class PodcrtanaDirektiva {
    constructor(el: ElementRef) {
        el.nativeElement.style.textDecoration = 'underline';
    }
}
```

Izvorni kôd 3.5: Primjer direktive

U HTML predlošku bismo gornju direktivu navodili ovako:

```
<span appPodcrtano>Neki tekst</span>
```

Izvorni kôd 3.6: Navođenje direktive

3.3.1. Kratka povijest

Kao okvir, Angular svoj život započinje kao osobni projekt Google-ovog zaposlenika Miška Heveryja, kako bi sebi olakšao izradu web aplikacija za interne projekte. Kako su Miško i još nekolicina kolega počeli koristiti taj projekt sve više, 2010. godine ga izdaju kao projekt otvorenog kôda imena AngularJS (prva inačica Angulara se temeljila na Javascriptu, ne Typescriptu). Samo ime *Angular* dolazi od kosih zagrada < i > koje se koriste u HTML oznakama (Gavigan 2018.).

Zajednica ga je odmah prihvatila raširenih ruku te brzo postaje iznimno popularan. Godinama kasnije, zbog ograničenja okvira i razvoja web tehnologija, dolazi do potrebe da se Angular u potpunosti prepiše (Gavigan 2018.).

Jedno takvo prepisivanje sa sobom donosi mnoge posljedice, pogotovo vezane za ekosustav koji je nastao oko biblioteke/okvira. Tako je u prepisivanju napuštena podrška za dotadašnje verzije Bootstrap-a i Angular Material-a. Također, dovodi u pitanje održivost trenutnih aplikacija koje su u produkciji. Ako se "stara" verzija Angular-a ne bude više održavala, kako će tvrtke održavati i podupirati svoje proizvode za nekoliko godina? Posljedica toga je bila da su se mnogi razvojni timovi prerano prebacili u uporabu tada još beta inačica novog Angulara, što je ostavilo negativne posljedice i dojmove. No kroz nekoliko godina i ažuriranja Angular se stabilizirao i postao ponovno prepoznat i popularan razvojni okvir (Gavigan 2018.). A AngularJS još uživa daljnju podršku do kraja 2021. godine (Darwin 2020.).

Unatrag zadnjih dvije godine su nove inačice i ažuriranja postala učestalija i više inkrementalna, sa manjim značajnim promjenama. Danas je aktualna Angular inačica 12 te uživa veliku popularnost i uporabu.

3.3.2. Typescript

Poznato je da je JavaScript dinamički i interpretirani jezik. Dok to sa sobom donosi određenu razinu slobode i fleksibilnosti koja može biti korisna, također ima svoje nedostatke. Kako varijable mogu biti bilo kojeg tipa u bilo koje vrijeme, greške nije moguće utvrditi prije izvođenja programa.

Kako bi riješili taj problem, Google se odlučio za razvoj novog statički tipiziranog jezika za web programiranje, zvanog *Dart* koji je trebao ispraviti probleme JavaScripta i zamijeniti ga. Beta verzija preglednika Chrome trebala je izaći sa virtualnim strojem za Dart putem kojeg bi se izvorno izvodio, a za potrebe unazadne kompatibilnosti bi se Dart prevodio natrag u JavaScript (Hiwarale 2020.).

No, Dart je naišao na velike kritike. Posebice radi fragmentiranja web razvoja kada je JavaScript bio de facto standard. Različiti preglednici već ionako podržavaju različite standarde ili dijelove standarada, tako da bi još jedan klijentski jezik uveo dodatne komplikacije u to područje (Hiwarale 2020.).

Poučeni djelomičnim neuspjehom Dart-a, jedan tim unutar Microsofta odučio se na drugačiji pristup prema razvoju jezika koji bi riješio probleme JavaScript-a. Umjesto osmišljavanja u cijelosti novoga jezika, odlučili su se na dopunjavanje JavaScript-ove sintakse, koja bi bila u cijelosti izborna. Time je nastao **TypeScript** (Hiwarale 2020.).

Jednostavno rečeno, TypeScript je proširenje standardnog JavaScript jezika uvođenjem sustava statičkih tipova podataka. U kojoj mjeri se time netko koristi je u cijelosti na njima. Nadalje, budući da se radi o izbornom "proširenju" JavaScript-a, sav JavaScript kôd jest valjani TypeScript kôd (ali ne i obratno).

Iako je proširenje JavaScripta i u potpunosti kompatibilan sa njime, TypeScript nije jezik koji se izvodi izvorno unutar web preglednika. TypeScript kôd se prevodi natrag u JavaScript (postupak koji se u ovakvim slučajevima nazivima *transpiliranje*, od engl. *transpile* i koji je poznat iz Dart jezika) te se nudi izbor verzije JavaScripta u koji se želi prevesti natrag TypeScript, što daje mogućnost podrške različitih, čak i starijih, verzija web preglednika.

3.3.2.1. Primjeri

```
1 let naziv = 'Test'; // Pretpostavljeni tip podatka je string
```

Izvorni kôd 3.7: Primjer: Pretpostavljeni tip podatka

```
1 let naziv: string; // Zadani tip podatka je string
```

Izvorni kôd 3.8: Primjer: Zadani tip podatka

```
let naziv: string|number; // Zadani tip podatka je string ili broj
```

Izvorni kôd 3.9: Primjer: Više mogućih tipova podataka

```
let naziv?: string; // Zadani tip podatka je string, ali može biti prazan (odnosno null)
```

Izvorni kôd 3.10: Primjer: Tip podatka sa neobaveznom vrijednošću

```
// Definirano je sučelje koje ima obavezne članove \textit{stranaA} i \textit{stranaB}
interface Pravokutnik {
    stranaA: number;
    stranaB: number;
}

// Konstanta tipa Pravokutnik (zapravo anonimni objekt) koja sadrži potrebne članove
const pk: Pravokutnik = {
    stranaA: 2,
    stranaB: 5
}

// Izaziva grešku, jer nedostaje član \textit{stranaB}
const pk2: Pravokutnik = {
    stranaA: 2
}
```

Izvorni kôd 3.11: Primjer: Sučelja kao tipovi podataka

```
class Osoba {
    ime: string;
    prezime: string;
    private OIB: string;

    constructor(ime: string, prezime: string, OIB: string) {
        this.ime = ime;
        this.prezime = prezime;
        this.OIB = OIB;
    }
}

let osoba = new Osoba("Osoba", "Osobić", "12345654321");
```

Izvorni kôd 3.12: Primjer: Klase kao tip podataka

4. Društvene mreže

Kada se priča o društvenim mrežama, može se to raditi u jednom od dva konteksta. Prije svega, društvene mreže kao pojam u razgovornom smislu odnose se na web aplikacije kao što su Facebook i Instagram, gdje korisnici registriraju svoje profile i komuniciraju sa drugim korisnicima, umrežavaju se, učitavaju i dijele vlastiti sadržaj, a što se u današnje doba točnije naziva *socijalnim medijima*.

No, pojam društvene mreže može se također pronaći u području sociologije, gdje ona označava skup aktera i njihove međusobne odnose. Takva društvena mreža ima znanstveni karakter i danas se uvelike koristi unutar područja *analize društvenih mreža*, na čiju tematiku se mogu pronaći mnoga istraživanja i djela (na stranici Google Scholar pretraga pojma *social network analysis* vraća pet milijuna dvjesto tisuća šezdeset rezultata [5.260.000]).

Prema (d. m. b. d. m. i Ellison 2007.a), društvene mreže (socijalni mediji) novije generacije karakterizira nekoliko značajki:

1. jedinstveni profili sastavljeni od sadržaja kojeg pružaju vlasnici profila, drugi korisnici kao i podaci sustava
2. javno prikazane veze sa drugim ljudima koje drugi korisnici mogu gledati i pratiti
3. mogućnost konzumiranja, stvaranja i interakcije sa strujama korisničkog sadržaja kojeg pružaju oni i njihove poveznice na mreži

Detaljnije o elementima društvenih medija, odnosno mreža, biti će u odjeljku 4.1.

4.1. Glavne značajke

4.2. Povijesni pregled razvoja društvenih mreža

Povijesno gledano, društvene mreže su stare kao i čovječanstvo. Od prvih zajednica ljudi, preko trgovinskih sporazuma do tehnološke sadašnjice. Naravno, za svrhe ovog rada, zanimljive su društvene mreže u modernom, informatičkom smislu. Njihova povijest počinje s nastankom interneta, odnosno tadašnjega ARPANET-a, kada su krajem 1969. godine dva računala po prvi puta međusobno komunicirala. Sljedeći važni korak dolazi 1972. godine kada se razvio Email protokol i poslana je prva poruka na taj način, koja je glasila, prema riječima izumitelja Tomlinsona, nešto poput "QWERTYIOP" (pogleda li se tipkovnicu, može se zaključiti da se radi o prvome redu alfabetskih tipki QWERTY tipkovnice) (Malloy 2016.).

U sklopu ARPANET projekta razvijeni su mnogi temeljni protokoli za mrežnu komunikaciju kao što su TCP/IP, Telnet, FTP i drugi.

4.3. Community Memory

Community Memory ime je prvog javno dostupnog medijskog sustava, točnije rečeno tzv. *bulletin board system* (BBS). Razvijen i uspostavljen je u sklopu tzv. *Project One* tehnološke komune u San Franciscu, sustav je postavljen na ulazu diskografskog dućana u Berkleyu, Kaliforniji. Bio je zamišljen kao sustav za dijeljenje informacija i resursa, obično u sklopu kontrakulturnih, obrazovnih i društvenih organizacija koje je povezivao sa javnošću (Malloy 2016.).

Slika 17: Jedan od terminala za Community Memory

Izvor: (Malloy 2016.)

Sustav je funkcionirao na temelju unosa ključnih riječi koje su interesirale osobu te bi potom prikaz ispisao poruke vezane uz tu riječ, od najnovije do najstarije. Pretraživanje sustava bilo je besplatno, no ukoliko je osoba htjela ostaviti poruku, bilo je potrebno ubaciti kovanicu od 25 centi. Unesena poruka mogla je biti odgovor na neku postojeću poruku (čime su se gradili nizovi poruka, odnosno dretve) ili poruka dodana na ključnu riječ, odnosno u takozvani indeks. Unešene poruke mogle su biti anonimne ili potpisane (Malloy 2016.).

Prvi je terminal pušten u funkciju 1973. godine i s obzirom da se nalazio ispred diskografskog dućana, prirodno je da je tematika glazbe sačinjavala najpopularniji pojam, no koristio se i za raznolike teme kao što su povijest, književnost, edukacija i razmjena znanja općenito. Kroz godine dodavani su novi terminali, neki javne, drugi zatvorenije prirode te je cjelokupni projekt okončan, neispunjenog potencijala, 1992. godine. No, za sobom ostavlja veliku ostavštinu i direktno je utjecao na razvoj i uporabu računala (Malloy 2016.).

4.3.1. PLATO

PLATO (skraćenica od engl. *Programmed Logic for Automatic Teaching Operations*) bio je sustav za računalno-potpomognuto obrazovanje, razvijan i korišten u Sveučilištu u Illinoisu. Koristio se u učionicama kao vrsta elektroničkog sustava za učenje (slično današnjem Moodle-

u, Edmodo-u, itd.), te se temeljio na vremenskome dijeljenju resursa. Glavni program nalazio se na *mainframe* računalu te su se korisnici spajali na njega putem terminala. Sustav je bio prepoznatljiv po svojim narančastim prikazima visoke razlučivosti (Malloy 2016.).

Sustav PLATO kroz svojih nekoliko inačica uveo je i prokrčio put mnogim konceptima koji se danas smatraju samo-razumljivima za više-korisničke sustave. Među njima spadaju:

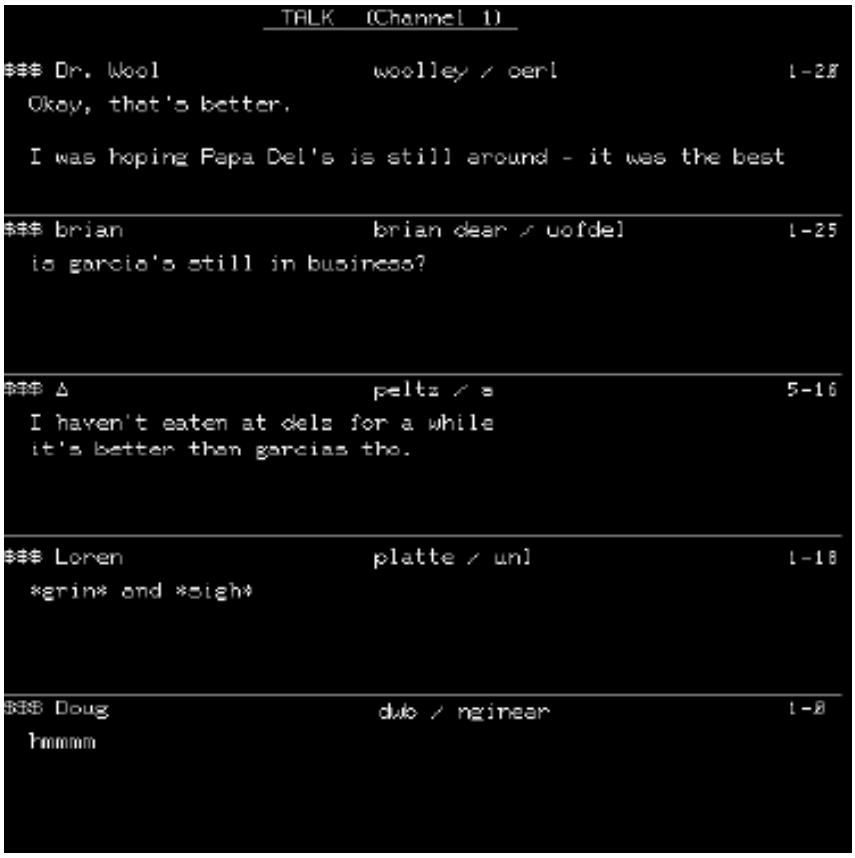
- prikazi na dodir
- emotikoni
- preteča foruma
- dijeljenje bilješki
- chat sobe
- dijeljenje ekrana
- igre za više igrača
- preteče RPG, roguelike i drugih žanrova računalnih igara (Malloy 2016.)

Sustav PLATO osmišljen je da bude maksimalno brzog odaziva na korisnički unos. Stoga, ako bi sustav bio usred iscrtavanja nekog sadržaja, pritiskom tipke za prekid ili promjenu prikaza, odmah bi se prekinulo trenutno iscrtavanje i prešlo na obradu sljedeće potrebne radnje (Malloy 2016.).

4.3.1.1. Talkomatic i Term-Talk

Jedan od najboljih primjera potonjega je program Talkomatic, neslužbeno izveden 1973. godine te koji je doživio veliku popularnost i uspjeh na sustavu PLATO. Radi se o programu za grupno dopisivanje koje podržava do 5 sudionika te neograničen broj promatrača razgovora. Svaki od 5 sudionika dobio bi $\frac{1}{5}$ prikaza za sebe, tako da su svi sudionici bili istovremeno vidljivi u svom virtualnome prozoru. Posebna značajka bila je ta što se sadržaj koji korisnici tipkaju odmah vidio dok su ga pisali, tako da je razgovor između sudionicima bio trenutačan i davao dojam živog razgovora (Malloy 2016.).

Talkomatic se ponašao više kao otvoreni forum odnosno trg negoli kao aplikacija za razgovor među ljudima. Nije postojala mogućnost da osoba direktno kontaktira drugu osobu niti da ih obavijesti da žele razgovor. S obzirom na popularnost programa Talkomatic, osoblje PLATO sustava razvilo je službeni program za razgovor *term-talk*, imenovan po redoslijedu naredbi koje korisnik treba unijeti u sustav kako bi uključio program. Za razliku od talkomatica, term-talk je podržavao isključivo dvije osobe, no bilo je moguće obavijestiti osobu te je prihvaćanjem razgovora se otvorio mali virtualni prozor za razgovor u zadnjim dvjema linijama ekrana. Napredna funkcionalnost za ono doba (Malloy 2016.).



Slika 18: Sjednica u Talkomatic sustavu

Izvor: (Malloy 2016.)

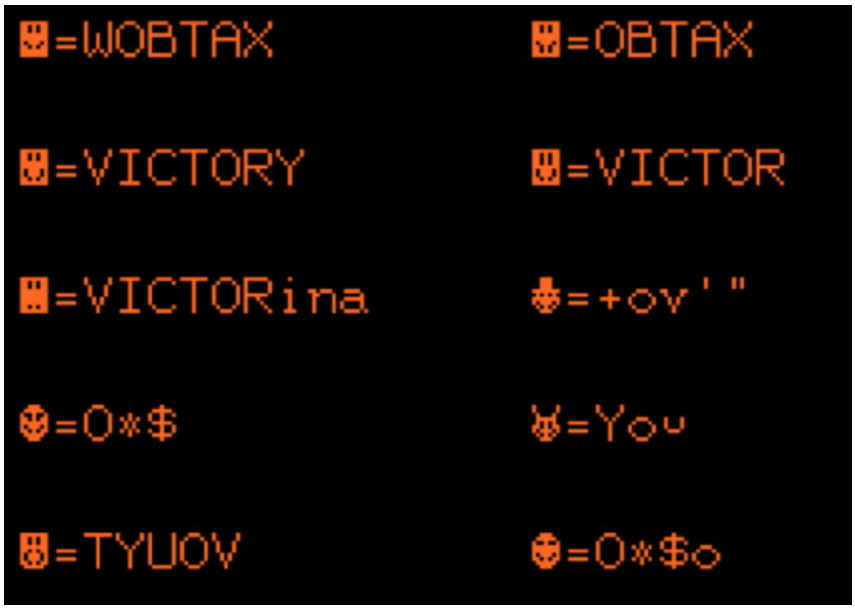
Još jedna, kasnije dodana, napredna funkcionalnost programa term-talk je bila mogućnost dijeljenja svog ekrana sa drugom osobom, kako bi se zajedno radilo na nekome problemu (Malloy 2016.).

No, takvi razgovori se ne mogu nigdje spremiti i jednom završeni, ne mogu se pregledavati. Za trajniju komunikaciju među korisnicima i projektnim skupinama, bilo je potrebno razviti sustav za trajnu pohranu poruka i bilješki (Malloy 2016.).

4.3.1.2. PLATO Notes

Kao sustav za učenje, ima smisla da je podržavao i funkcionalnost pisanja bilješki, kako osobnih, tako i grupnih za suradnju. Kroz nekoliko inačica, zaključno sa *Group notes* 1976., razvijen je sustav sa sljedećim značajkama:

- pisanje i dijeljenje privatnih bilježaka
- mogućnost pregleda cjelokupne povijesti lanca bilješki
- upravljanje pristupom datotekama bilješki
- brisanje bilješki



Slika 19: Emotikoni podržani u sustavu PLATO

Izvor: (Malloy 2016.)

- anonimno objavljivanje (Malloy 2016.)

Sustav bilješki nije se koristio samo za službenu korespondenciju, već i za neslužbene tematske razgovore, kao vrsta ranog i jednostavnog foruma (Malloy 2016.).

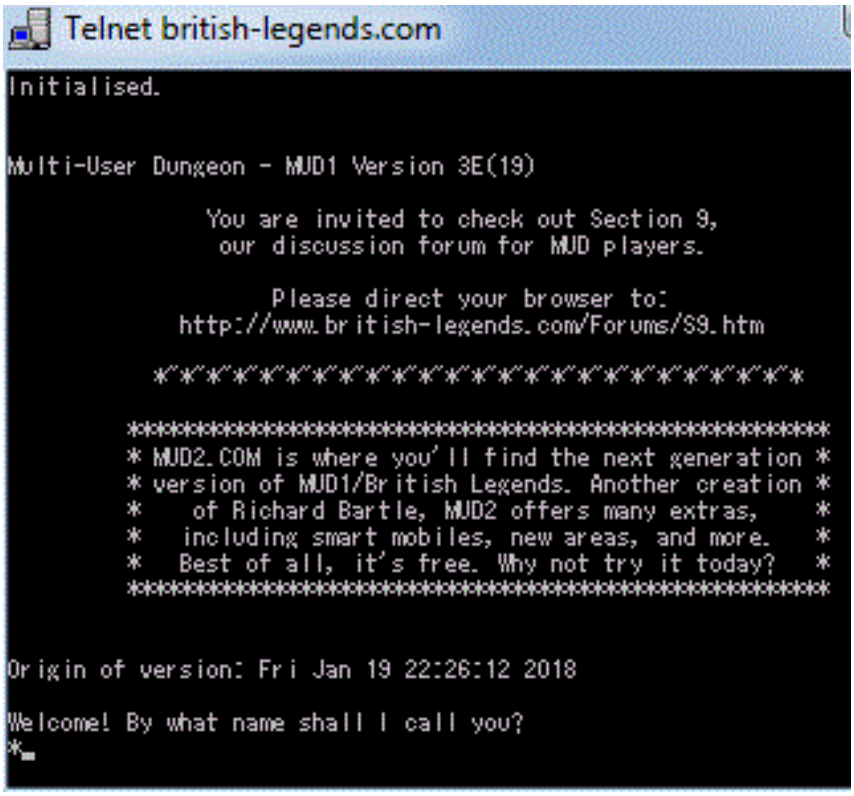
U sklopu talkomatica, term-talka i notes programa počela se stvarati jedna online zajednica unutar koje su postojale poznate ličnosti, interesne skupine, podzajednice, interne šale i jedna zajednička kultura sa svojim obrascima. Dapače, može ju se smatrati prvom pravom online zajednicom u povijesti računalstva i mreža (Malloy 2016.).

4.3.2. Tekstualne mreže

Pod *tekstualne mreže* podrazumijevaju se interaktivne računalne platforme koje su se prikazivale i sa kojima se sučeljavalo tekstualno. Spomenuti će se u nekoliko crtica još poneke povijesno značajne platforme unutar kojih su se razvijale društvene mreže i zajednice.

MUD MUD je skraćenica koja stoji za *Multi User Dungeon*, te po nazivu se može do neke mjere zaključiti o čemu se radi. MUD-ovi su u načelu preteče današnjih masovnih igara za više igrača, odnosno MMO (engl. Massive Multiplayer Online) igara. Dapače, neke od najpoznatijih MMO igara kao što su Everquest i World of Warcraft jasno pokazuju svoje korijene, uglavnom u obliku tekstualnih komandi koje igrači mogu izvršavati. Kao takvi, MUD-ovi se mogu smatrati prvim virtualnim svjetovima u kojima su igrači zajedno boravili i razvijali zajednicu.

Inspiriran igrama kao što su Colossal Cave Adventure i Zork, prvi MUD je izradio Roy Trubshaw 1978. godine za Sveučilište u Essexu te 1980. postaje prva online role-playing igra za više igrača, kada se sveučilišna mreža spaja sa ARPANET-om (Bartle 2016., Bartle 1990.).



Slika 20: Početni prikaz igre prilikom spajanja

Izvor: https://en.wikipedia.org/wiki/MUD

MUD postao je vrlo popularan u 80-ih godinama te je znao imati veliki broj aktivnih igrača. Popularna krilatica pripisana toj igri glasi "You haven't lived until you've died in MUD...", odnosno "Niste živjeli dok niste umrli u MUD-u" (Bartle 1999.).

Inspirirao je mnoge "kopije" i nasljednike kao i vlastiti žanr igara. Iako su danas daleko popularniji moderni MMO naslovi, i dalje postoji zajednica entuzijasta koji igraju i razvijaju nove MUD-ove (http://www.mudconnect.com/).

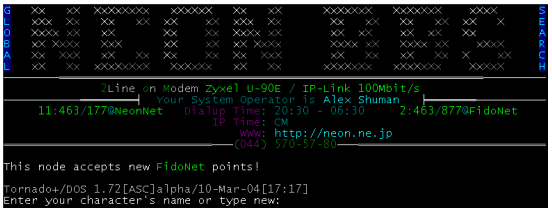
BBS Spomenuti Community Memory može se smatrati pretečom budućih BBS, odnosno *Bulletin Board* (hrv. *elektronička oglasna ploča*) sustava. Glavna razlika nalazi se u arhitekturi, gdje se Community Memory posluživao na *mainframe* računalu na koje su spajali terminalima, dok je pojava i šira dostupnost mikroračunala nešto kasnije učinila praktičnim umrežavanje korisnika tih računala kako bi međusobno komunicirali i razmjenjivali podatke.

Upravo tako 1978. godine Ward Christensen i Randy Seuss stvaraju Chicago Bulletin Board System, CBBS i objavljuju u jednom hobističkom časopisu. Iako su poznati po tekstualnom prikazu i komunikaciji, neka od prvih osobnih računala spojena na BBS nisu imala niti prikaze te se sadržaj očitavao sa ispisa na papir (Malloy 2016.).

BBS su postali popularan način izgradnje malih zajednica te su u početku postojalo tisuće malih BBS zajednica od makar dvije osobe, koje su tako komunicirale. S vremenom nastaju specijalizirani BBS-ovi za specifične zajednice kao što su ruralna područja, LGBT, Američki

Slika 21: Uvodna poruka CWRU BBS sustava

Izvor: https://en.wikipedia.org/wiki/Bulletin_board_system



Slika 22: Primjer BBS-a sa kreativnijim i vizualno naprednijim prikazom

Izvor: https://en.wikipedia.org/wiki/Bulletin_board_system

domoroci, Afro-američke zajednice i naravno po hobističkim interesima (Malloy 2016.).

Pred dolazak Interneta, sredinom 90-ih BBS sustavi su bili iznimno popularni te ih je čak 40.000 bilo umreženo sa milijunima aktivnih korisnika. Za usporedbu, društvena mreža danas ukoliko ima nekoliko milijuna korisnika, nije nešto impresivno (Malloy 2016.).

Usenet Po mnogim pitanjima sličan BBS sustavima, Usenet je zaseban sustav za distribuirane diskusije. Temelji se na *Unix-to-Unix Copy* arhitekturi. Iako sličan BBS, Usenet ima definiranu strukturu u smislu da korisnici čitaju i objavljuju poruke na jednu ili više kategorija, koje se zovu *newsgroup*-ama (Encyclopedia Britannica 2021.).

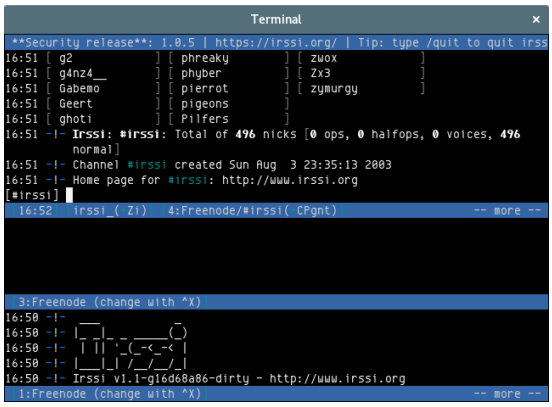
Povijesno gledano od velike važnosti kao preteča modernih online foruma. Mnogi moderni kulturološki koncepti dolaze od Usenet diskusijskih grupa, kao što su termini *FAQ*, *flame* i *spam*.

IRC IRC skraćenica je za Internet Relay Chat, protokol aplikacijskog sloja koji služi prijenosu tekstualne komunikacije među korisnicima na podržanome poslužitelju. Korisnici moraju instalirati klijentski program kako bi se spojili na neki od dostupnih poslužitelja te se pridružuju jednom ili više od ponuđenih kanala na tom poslužitelju. Kanali su najčešće tematski određeni. Uz grupnu tekstualnu komunikaciju, podržava i privatne poruke među korisnicima te prijenos datoteka.

Nastao 1988. godine, kako bi zamijenio program *MultiUser Talk* jednog finskog sveučilišta, prvotni cilj autora Jarkko Oikarinena bio je proširiti BBS sustav koji je administrirao kako bi u njega uveo određene značajke kakve posjeduju Usenet sustavi, kao što su vijesti i diskusije u pravome vremenu (Charalabidis 2000.).

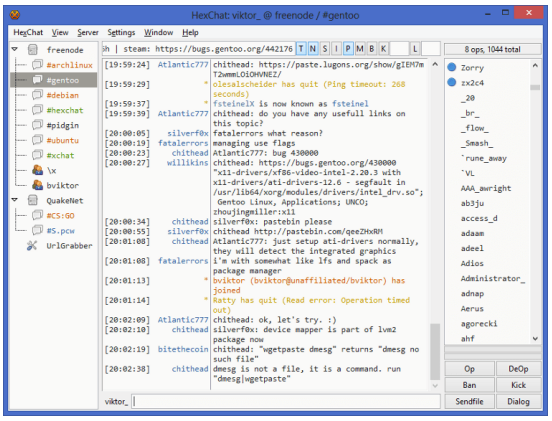
Prvi implementirani dio bila je *chat* funkcionalnost. Prva IRC mreža vrtjela se na tolsun.oulu.fi poslužitelju te se ubrzo diže još nekoliko poslužitelja u sklopu drugih fakulteta na kojima je imao prijatelje. Broj korisnika je rastao, i Jarkko ubrzo shvaća da mu preostale zamišljene funkcionalnosti nisu potrebne. Tako nastaje prva IRC mreža (Stenberg 2021.).

Na svom vrhuncu, oko 2003./2004. godine, IRC mreže skupa brojale su u prosjeku preko 800.000 aktivnih korisnika u isto vrijeme. Danas se prosječni broj korisnika procjenjuje na 380 tisuća, što je daleko manje, no ipak nije zanemariva brojka te se može zaključiti da je IRC i dalje u uporabi (Netsplit 2021.).



Slika 23: Staromodni *textmode* klijent za IRC

Izvor: https://en.wikipedia.org/wiki/Internet_Relay_Chat



Slika 24: Grafički klijent za IRC

Izvor: https://github.com/irssi/irssi

4.3.3. SixDegrees

1997. izdaje se prva društvena mreža u modernome smislu, zvana *SixDegrees* (nazvana po ideji da se bilo koje dvije osobe na svijetu može povezati u najviše 6 skokova). Podržavala je osnovne funkcionalnosti kakve se mogu očekivati kao što su korisnički profili, popis prijatelja i mogućnost pregledavanja tuđih popisa prijatelja, odnosno poveznica. Ovakve značajke su postojale naravno i prije, no ne u jednome paketu. SixDegrees predstavljao se kao alat koji pomaže ljudima u međusobnome povezivanju i izmjeni poruka (d. m. b. d. m. i Ellison 2007.b).

Iako je privukao milijune korisnika u prvim godinama, nije uspio održati stabilni poslovni model te se usluga 2000. godine zatvara (d. m. b. d. m. i Ellison 2007.b).

4.3.4. Friendster

Sljedeća velika društvena mreža koja nastaje zvala se Friendster te se pokreće 2002. godine, paralelno sa nekolicinom drugih mreža čiji su osnivači svi međusobni poznanici - *Ryze*, *Tribe.net* i *LinkedIn*. Nadali su se da će se tako moći međusobno podupirati bez takmičenja. No na kraju Ryze nikada ne stječe popularnost, LinkedIn postaje moćna poslovna mreža, a Tribes.net uspijeva privući samo nišnu korisničku bazu (d. m. b. d. m. i Ellison 2007.b).

Osnovna ideja Friendstera bila je dating platforma. No umjesto da korisnike povezuje sa strancima na temelju nekakvih zajedničkih interesa i karakteristika, kao što je to većina drugih dating stranica tog vremena radilo, Friendster je djelovao na pretpostavci da će poznanici

poznanika, odnosno prijatelji prijatelja, biti bolja osnova za ljubavne interese. Stranica je dobila na zamahu zahvaljujući trima grupama ranih usvojitelja - blogeri, posjetioci Burning Man festivala te homoseksualni muškarci. Stranica je narasla do 300 tisuća korisnika samo na temelju usmene predaje, sve do javnog medijskog pokrića sredinom 2003. godine (d. m. b. d. m. i Ellison 2007.b).

Kako je popularnost platforme rasla, naišla je na mnoge tehničke i društvene poteškoće. Arhitektura sustava i njene baze podataka nisu bile opremljene da podnesu nagli porast prometa kojem je stranica bila podvrgnuta zbog brzog rasta u popularnosti. Zbog toga sustav je često trpio ispade, što je frustriralo korisnike. Sa strane društvenih poteškoća, to se iskazalo u vidu činjenice da je tako veliki broj novih korisnika poremetio "društveni poredak" i kulturu koja se prirodno razvijala. Novi korisnici nisu se više stigli asimilirati u postojeći društveni kontekst te štoviše, kako je postajala sve popularnija, značilo je da neki korisnik sada susreće, uz svoje bliske prijatelje, i svoje kolege sa posla (primjerice šefa) kao i bivše kolege iz razreda i tako dalje (d. m. b. d. m. i Ellison 2007.b).

Zbog određenih poslovnih odluka kao i pojave konkurentnih alternativa, s vremenom popularnost Friendstera na zapadu opada i korisnici prelaze na nove, zanimljivije mreže. Dok je popularnost na zapadu pala, na istoku je tek počinjala te je Friendster od 2007. godine počeo privlačiti veliki broj korisnika u azijskim državama poput Filipina, Singapura, Malezije i Indonezije. 2009. je Friendster brojao 115 milijuna korisnika te ga je otkupila azijska tvrtka MOL Global (d. m. b. d. m. i Ellison 2007.b).

2015. Friendster se gasi, a 2018. godine tvrtka koja stoji iza njega se zatvara.

4.3.5. MySpace

Jedna od općenito poznatijih mreža u njihovoj povijesti zove se MySpace te je osnovana 2003. godine kako bi konkurirala Friendsteru i sličnim mrežama. Između ostaloga, osnivači su htjeli kapitalizirati na korisnicima koje su Friendsterove poslovne prakse otuđile i odvratile, što se pokazalo ispravnim te je MySpace ubrzo doživio veliki polet u broju korisnika. U svojim najboljim danima, MySpace je imao više stotina tisuća novih korisnika na dnevnoj osnovi.

Glavne značajke koje su tada razdvajale MySpace od drugih društvenih mreža su bile mogućnost opširne personalizacije svog profila budući da nisu zabranjivali unos HTML/CSS kôda. Također, MySpace je osluškivao zahtjeve korisnika i njihove navike te je dodavao nove značajke na temelju toga (d. m. b. d. m. i Ellison 2007.b).

Iako nije originalno zamišljena kao društvena stranica za glazbu, spletom okolnosti stranica je bila gurnuta u tome smjeru, budući da su mnogi neovisni glazbeni izvođači prebjegli sa Friendstera (budući da nisu mogli zadovoljiti Friendsterova ograničenja vezana uz korisničke profile) i poticali svoje obožavatelje da se također prebace na MySpace. Tako je nastala osnovna sprega glazbenika i obožavatelja po kojoj je MySpace postao prepoznat. Nadalje, MySpace je od osnivanja YouTube-a nudio mogućnost ugrađivanja videa sa te platforme (d. m. b. d. m. i Ellison 2007.b).

Kako je glazba i zabava bila jedna od glavnih tematika MySpace-a, osnovali su čak i glazbenu izdavačku etiketu preko koje su mnogi danas poznati glazbeni postigli slavu (d. m. b. d. m. i Ellison 2007.b).

Od 2008. nadalje, MySpace počinje gubiti korisnike zbog konkurenata poput Facebook-a i Twitter-a. Aktivno posluje i dan danas, no više nije ni približno popularna kao nekoć.

4.3.6. Facebook

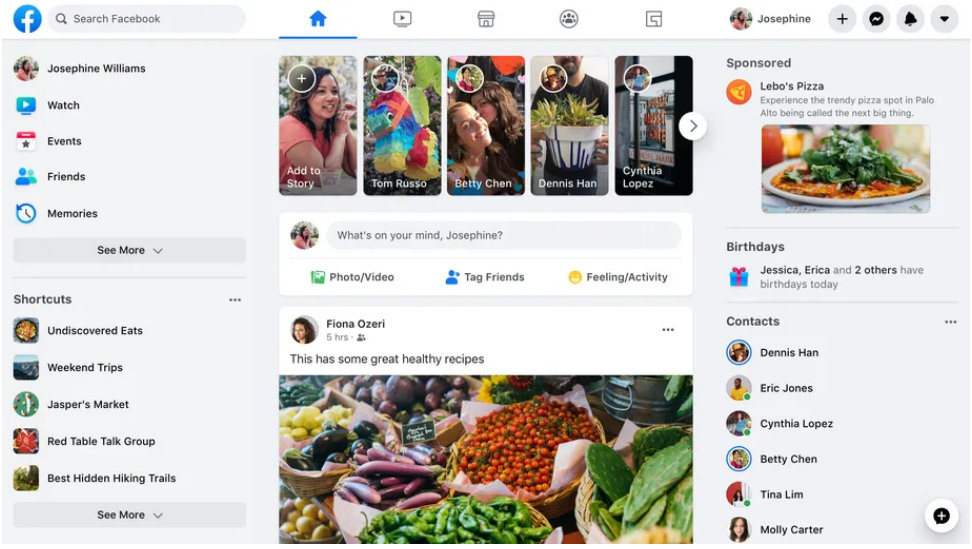
Najpopularnija i najpoznatija društvena mreža svih vremena, Facebook nastaje 2004. godine isprva kao nišna mreža imena *thefacebook* dostupna isključivo studentima Harvardskog sveučilišta, s ciljem da bude jedinstveni i centralni direktorij za sve studente sveučilišta (samo ime *Facebook* potječe od ideje knjige lica, nalik na srednjoškolske maturantske slikovnice). Budući da su korisnici morali posjedovati neku službenu akademsku email adresu, u početnim danima postojalo je ozračje isključivosti i intimne privatne zajednice. Već unutar mjesec dana od svog otvaranja, pola studenata sveučilišta su imali korisnički račun te još nekoliko mjeseci kasnije se Facebook otvara za Yale, Columbia i Stanford. Još koji mjesec kasnije i Facebook je primao financijske investicije od strane Petera Thiela te su preselili sjedište u Palo Alto u Kaliforniji. Krajem iste godine (2004.) Facebook je najavio da je dosegao milijun korisnika (mahom svi studenti različitih sveučilišta i fakulteta koji su imali tada pristup) (McNamee 2019.).

2006. Facebook se otvara svim korisnicima koji imaju najmanje 13 godina starosti i posjeduju valjanju email adresu. Broj korisnika brzo raste i među njima broje se i stotine tisuća korporativnih profila. Već 2008. Facebook nadmašuje svog konkurenta MySpace u broju korisnika (Hall 2021.). Iste godine se Facebook počinje koristiti i u političke svrhe kada su nastale

Slika 25: Osnivač i izumitelj Facebook-a, Mark Zuckerberg

Izvor: https://en.wikipedia.org/wiki/Facebook

stotine grupa u podršci za američke kandidate John McCaina i Barack Obamu (Hall 2021.). Tada izlazi i Facebook-ova funkcionalnost za direktno dopisivanje *Facebook Chat* (danas Facebook Messenger) (Arrington 2008.).



Slika 26: Grafičko sučelje Facebook-a

Izvor: https://www.theverge.com/2020/3/19/21187118/facebook-desktop-redesign-more-available-starting-today

Kao jedan od najboljih i najpoznatijih primjera uspješnosti u Web 2.0 razdoblju, Facebook je privukao mnogo pažnje i samim time i mnogo kritika vezano uz svoje poslovne prakse i povijest. Tako je Facebook-ov poslovni model izložen dugogodišnjim kritikama, primarno zbog načina kako barataju podacima svojih korisnika i ciljanog reklamiranja korisnicima. Od samih početaka, prosti doseg koji je Facebook imao i mogućnost da bilo tko napravi korisnički račun, činio je Facebook atraktivnim korporacijama kako bi stupili direktno u kontakt sa potrošačima i efikasnije se reklamirali.

Dodatne kritike su usmjerene i prema Facebook-ovom pristupu spram korisničke privatnosti. Kritičari ističu da su Facebook-ove postavke za privatnost nepotrebno zamršene i da ih često mijenjaju. Druge kritike su pak usmjerene prema društvenim mrežama općenito i načinu kako vladaju ljudskom pažnjom i usmjeravaju svačije ponašanje. Ne asmije se ni zaboraviti spomenuti *Cambridge Analytica* skandal pred nekoliko godina koji je pokazao koliko

daleko i duboko odlazi prikupljanje korisničkih podataka koje stranice, odnosno tvrtke, poput Facebook-a vrše nad korisnicima (Briggs, Burke i Ytreberg 2020.).

Bilo kako bilo, neupitno je da je Facebook danas i dalje najveća i najutjecajnija društvena mreža sa preko 2 milijarde korisnika. No svakako je valjano upitati se hoće li još dugo vremena uživati takav položaj (Statista 2021.).

4.3.7. Reddit

Prednja stranica Interneta - tako sebe okrunjuje web stranica Reddit.

Iako nije društvena mreža u klasičnom smislu, kroz godine i dodavanjem proširenih funkcionalnosti ipak poprima značajke društvenih mreža. U svojoj osnovi Reddit je *agregator* društvenih novosti i zanimljivosti. Te zanimljivosti se tematski dijele u zajednice, čija je osnova neko područje interesa kao što su npr. sport, znanost ili politika, koji se, maštovito, zovu *podreddit*-ima (engl. *subreddit*). Dok su spomenute zajednice popularne, postoje i male zajednice za vrlo nišne interese. Postoji šala koja kaže da ako nešto postoji, onda se nalazi na Redditu ("There's a sub for that").

Sadržaj na Redditu se ravna na temelju sustava glasovanja. Svaki korisnik može svakoj objavi dati pozitivan odnosno negativan glas, što se na hrvatskome podredditu kolokvijalno naziva *goreglasom* i *doljeglasom*, zbog činjenice da su prikazani kao strelica gore i strelica dolje pored svake objave. Svaki glas pridodaje se tzv. *karmi* objave i posljedično karmi korisnika koji je objavio sadržaj. Na taj način, korisnici moderiraju i razvrstavaju sadržaj koji im je zanimljiv i koji ne.

Reddit osnivaju 2005. godine Steve Huffman, Alexis Ohania. No nije započelo sa Redditom, nego su isprva htjeli kreirati aplikaciju koja će omogućiti naručivanje hrane u restoranima unaprijed, kako osoba nebi trebala čekati. No u to vrijeme mobilna tehnologija nije još bila dovoljno razvijena da bi takvo što bilo moguće na širokoj razini, jer premali broj korisnika je tad posjedovao mobilne telefone, mobilni pristup Internetu je bio još rjeđi. Iz tih razloga je, tada novoosnovani, Y Combinator odbio predloženu ideju. No nije odbio ljude koji stoje iza ideje i njihovu energiju. Osnivač Paul Graham pozvao ih je natrag da porazgovaraju o novoj ideji. Inspiriran Slashdotom i Deliciousom, društvenim stranicama za online oznake koje korisnici mogu dijeliti i komentirati, Graham im predlaže sličnu stranicu, ali za sadržaj koji pripada tome času, odnosno koji tada konzumiraju, jer oznake nam uglavnom služe za "kasnije". Ideja se dopala svima i suradnja je uspostavljena. Ohanian i Huffman se, skupa sa preostalih 8 prihvaćenih timova, preseljavaju u Boston, gdje će 3 mjeseca raditi na svojem projektu (Lagorio-Chafkin 2018.).

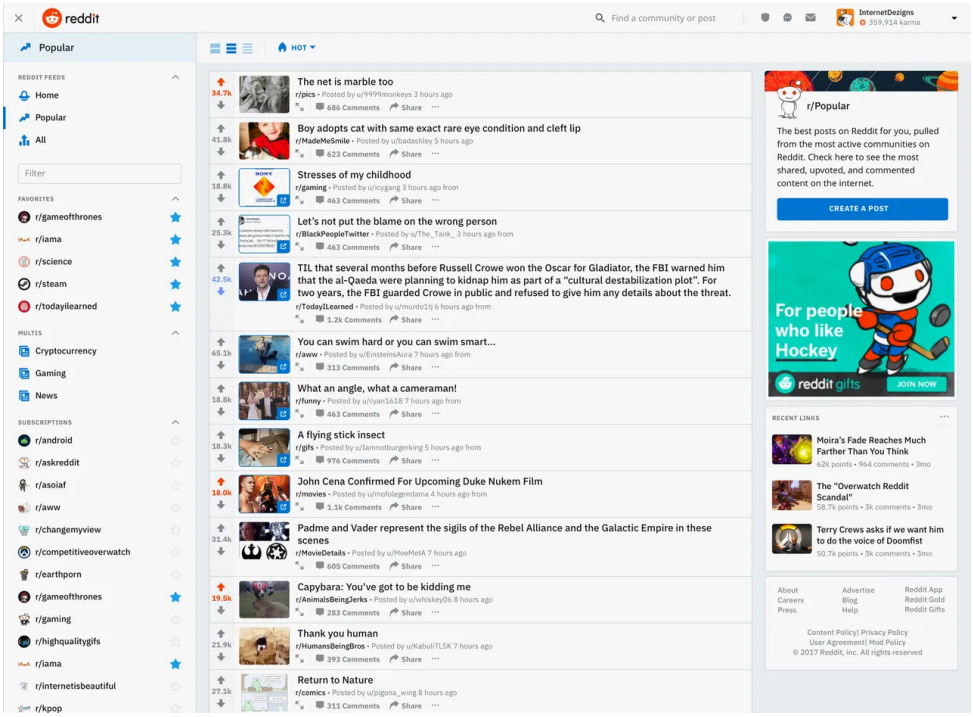
Iz toga se rađa Reddit, koji je isprva izrađen u Lispu i prvi puta objavljen na Internet 23. Lipnja 2005. godine. Prvi korisnici bili su, naravno, sami osnivači, investitori/mentori i kolege iz drugih startup-ova koji su dijelili program sa njima. Dapače, prvih 100 "korisnika" su bili istih nekoliko ljudi koji su objavljivali sadržaj pod različitim imenima kako bi dali dojam da je stranica puno življa nego što zapravo, na svom početku, jest. Mjesec dana kasnije, Reddit je počeo dobivati na korisnicima koji su se mogli brojati u nekoliko stotina, zahvaljujući objavi na blogu

Paula Grahama, koji je direktno podržavao njihov rad (Lagorio-Chafkin 2018.).

Kroz nekoliko mjeseci, promet stranice je porastao do tolike mjere da se jedini poslužitelj, koji je posluživao stranicu, povremeno rušio. U međuvremenu su se timu pridružili Aaron Swartz i Christopher Slowe. Krajem 2005. godine prepisuju cjelokupnu stranicu u Python zbog veće fleksibilnosti i dostupnosti gotovih rješenja za mnoge probleme, koje je Huffman morao sam rješavati dok je implementirao stranicu u Lispu. Tako su dobili proširivu platformu koja postiže bolje performanse (Lagorio-Chafkin 2018.).

Prema kraju 2006. godine, Reddit je bio prodan Conde Nast Publicationsu, vlasniku Wired časopisa za 10 milijuna dolara. Huffman i Ohanian napuštaju Reddit 2009. Erik Martin je nakon toga bio odgovoran za daljnji rast i održavanje stranice pod novim vlasništvom. Uvode se reklame u obliku sponzoriranog sadržaja kao i vlastite reklamne platforme. Kroz godine, Reddit izmjenjuje nekoliko izvršnih direktora koji uvode raznolike promjene kao što su Reddit Gold (premium pretplata) i mogućnost kupovine Reddit Gold-a sa Bitcoinom, kao i uvođenje konkretnih politika protiv online napastovanja i neprimjerenog sadržaja te u sklopu toga zabranio nekolicinu zajednica koje su raspirivale netrpeljiv sadržaj, gajile otrovnu kulturu i ponašanje i objavljivalje osobne podatke drugih osoba protiv njihove volje (primjerice /r/fatepeoplehate i /r/beatingwomen) (Lagorio-Chafkin 2018.).

2014. se Ohanian i Huffman vraćaju vodećim pozicijama unutar Reddit-a te pokreću iOS i Android aplikacije, uređuju mobilnu stranicu i uvode sustav za A/B testiranje budućih značajki (Hempel 2021.). 2018. godine uvode redizajn stranice kako bi bio moderniji (uostalom, neki korisnici su ga nazivali "distopijskim Craiglist-om"), no stari dizajn je i dalje dostupan pod drugom poddomenom, za one koji ga preferiraju (old.reddit.com) (Pardes 2018.).



Slika 27: Novo korisničko sučelje Reddit-a

Izvor: https://www.wired.com/story/reddit-redesign/

Zadnjih godina dodaju chat i livestream funkcionalnosti te prema zadnjim izvještajima, Reddit broji 50 milijuna dnevno aktivnih korisnika, odnosno 400 milijuna aktivnih korisnika mjesečno. (Dean 2021.)

4.3.8. Youtube

YouTube je danas najpopularnija platforma za distribuciju video sadržaja, sa preko 2 milijarde mjesečnih korisnika, što je više od Facebook-a (Statista 2021.). YouTube nudi niz servisa i proizvoda uz samo objavljivanje video sadržaja, kao što su Youtube Music (platforma za streaming glazbe), YouTube TV (televizijski servis), YouTube Movies (usluga za kupovinu digitalnih kopija filmova), YouTube Go (jednostavnija verzija namijenjena za tržišta u državama u razvoju), YouTube Gaming (*live streaming* servis koji konkurira Twitch-u) i YouTube Premium (plaćena pretplata s dodatnim pogodnostima kao npr. odsustvo reklama).

Naravno, YouTube nije oduvijek bio velika multimedijska sila, već ima svoje skromne početke u Web 2.0 eksploziji ranih 2000-ih godina. Prvotna ideja njenih tvoraca, Chad Hurleyja, Steve Chena i Jawed Karima, bila je *dating* platforma, ali s videom umjesto slikama te da su tvorci na Craiglistu nudili 100$ svakoj atraktivnoj ženi koja bi objavila 10 video isječaka na njoj. Niti jedna osoba im se nije javila, pa su se predomislili. Postoji nekoliko verzija priče o postanku YouTube-a, no najvjerojatnija je ta da je nastao iz potrebe da se lakše pronađe i dijeli video sadržaj, kada tvorci nisu mogli pronaći isječak 38. Super Bowl-a niti isječke tsunamija indijskog oceana iz 2004. godine (Burgess i Green 2018.).

U veljači 2005. godine YouTube postao je neslužbeno dostupan na adresi `www.youtube.com` i 23. travnja iste godine Jaed Karim objavljuje prvi video, imena "Me at the zoo". Na početku su video sadržaji bili ograničeni na 100 megabajta veličine. U svibnju je pokrenuta javna beta inačica i u studenome te godine je Nike reklama sa Ronaldinhom bio prvi video koji je dosegao milijun pregleda. 15. prosinca 2005. stranica se službeno objavljuje. U to vrijeme je stranica dobivala i do 8 milijuna pregleda na dan (Burgess i Green 2018.).

Isti tjedan, YouTube je zadobio pozornost mainstream medija kada je skeč *Saturday Night Live*-a bio objavljen stotine puta na platformi što je u veljači 2006. godine izazvalo pravni odgovor NBCUniversal-a koji je zahtijevao uklanjanje svih vezanih videa, koji do tada uspjeli zajedno prikupiti 5 milijuna pregleda. Usput rečeno, taj video može se smatrati jednim od prvih "viralnih" objava na Internetu. Ovakva medijska eksponiranost je uvelike pomogla u porastu popularnosti stranice te je sredinom 2006. godine brojala 100 milijuna dnevnih pregleda (Burgess i Green 2018.).

6. listopada 2006. godine Google kupuje Youtube za 1.65 milijardi dolara vrijednosti u Googleovim dionicama. Kupovina YouTube-a i njena medijska ekspozicija ishodila je još brži rast i već 2010. godine, YouTube je imao tržišni udio od 43%, prema comScoreu (Comscore 2010.). 2013. YouTube je obznanio da se učitavalo na stranicu 100 sati videa svake minute (Welch 2013.). 2017., YouTube je bio odgovoran za 46% svog prometa online servisa za strujanje glazbenog sadržaja (preko YouTube Music servisa) (Music Business Worldwide 2018.).

Danas je YouTube druga najposjećenija web stranica nakon Google tražilice, sa preko

2 milijarde posjetitelja, od kojih 89% dolazi izvan Sjedinjenih Američkih Država.

4.3.8.1. Alternativni servisi

YouTube nije prvi, niti naravno jedini servis za pohranu i dijeljenje videa.

Vimeo Vimeo osnovali su zaposlenici CollegeHumora kako bi dijelili smiješne video objave među sobom i svojim kolegama. Jedno vrijeme se nije ništa pretjerano događalo sa tom stranicom i stavljena je na stranu kako bi se dalje posvetili rastućoj popularnosti CollegeHumora. No, kako je YouTube došao na scenu, postao popularan i kupio ga Google, IAC koja je kupila u međuvremenu Vimeo se odlučila uložiti više razvoja u Vimeo i konkurirati YouTube-u, usredotočenošću na probrani sadržaj, sadržaj visoke kvalitete i kroz godine uvođenje alata za izradu sadržaja i davanje što veće podrške stvarateljima sadržaja. Vimeo je danas uspješna platforma sa više stotina milijuna korisnika (Vimeo 2021.).

DailyMotion Francuski startup i servis za dijeljenje video sadržaja. Nastaje u slično vrijeme kao YouTube i danas je druga najveća video hosting stranica nakon YouTube-a. Uz DailyMotion se veže nekoliko sudskih parnica vezanih za autorska prava. Dailymotion je kao odgovor na to pojačao nadzor sadržaja zaštićenog autorskim pravima (Dailymotion 2021.).

Peertube PeerTube je takozvana *FOSS* (free and open source) platforma za distribuciju videa. Započeo je 2017. kao osobni projekt programera zvanog *Chocobozzz*, ali 2018. godine Framasoft, francuska nezavisna i neprofitna organizacija zapošljava ChocoBozzza i pokreću kampanju za skupno financiranje (engl. *crowdfunding*). Njena svrha je nuditi besplatnu i decentraliziranu alternativu platformama poput YouTube-a, Vimeo-a i Dailymotion-a, za koje drži da previše žrtvuju prava i privatnost svojih korisnika. PeerTube je, naspram drugih platformi, organiziran decentralizirano. Drugim riječima, svatko tko želi može podignuti poslužitelj PeerTube-a i ima potpunu kontrolu nad time kako će ga moderirati, kako će izgledati i sa kojim drugim instancama će ga povezati. PeerTube se temelji na *ActivityPub* protokolu za distribuirane mreže. To znači da korisnik PeerTube-a može zapratiti i biti u interakciji sa korisnicima drugih mreža koji koriste taj protokol (npr. Mastodon). Unatrag zadnjih godinu dana dodali su mogućnost pretraživanja kroz sve instance, unaprijeđene alate za moderiranje i peer-to-peer mogućnost *live streaming*-a (PeerTube 2021., JoinPeerTube 2021.).

4.3.9. Twitter

Twitter je takozvana *microblogging* platforma i u doba svog nastajanja, 2006. godine, uglavnom nitko nije bio upoznat sa tim terminom. Tijekom prve godine postojanja, korisnici su eksperimentirali sa različitim metodama uporabe stranice dok Twitter nije 2007. godine doista dobio na zamahu (Dijck 2013.).

Sam termin *microblogging* odnosi se na format digitalnih medija sličan bloggingu, no značajno smanjenog opsega i širine. Zbog manje veličine sadržaja, češće su neformalnoga

karaktera, no mogu se i dalje doticati specifičnih i dubljih tematika. Uz Twitter, neki dodatni primjeri microblogova su Telegra.ph, Tumblr, micro.blog, Gab i Parler.

Naziv Twitter dolazi od engleske riječi *tweet* što znači *cvrkut* i odnosi se na središnju funkcionalnost platforme - razmjenjivanje i objavljivanje kratkih poruka, obično obavijesti i novosti, ograničenih na 140 znakova (danas povećano na 280). Objavljeni *tweet*-ovi su javno vidljivi svim korisnicima, osim ako ih korisnik ne označi kao *zaštićene*, u kojem slučaju su vidljivi samo pratiteljima korisnika.

Jedna od glavnih snaga Twitter-a je jednostavnost sučelja i uporabe. Danas svi znamo što je SMS poruka i kako ju poslati. Twitter su znali nazivati "SMS-om interneta" zbog svog formata (Dijck 2013.).

Twitter je jako brzo pronašao nišnu poziciju na tržištu društvenih medija, zbog svoje "uskogrudne" prirode. Jednostavno je objavljivati kratak sadržaj i svima je vidljiv. Također, budući da se sadržaj lako i brzo objavi, pogodan je za obavještavanje, praćenje i komentiranje o događajima u stvarnome vremenu. Stoga ne čudi da je Twitter ubrzo pronašao svoju nišu i kod reportera i novinara, bili oni hobisti ili profesionalci. Jedan od glavnih primjera toga dolazi iz 2009. godine kada je uslijed predsjedničkih izbora u Iranu oporba krenula na ulice u prosvjedu protiv rezultata izbora. Novoizabrana vlada je ubrzo uzvratila represivnim mjerama i zabranila javnim medijima izvještavanje o događajima. Tu je Twitter uskočio kao netradicionalni medij kojeg Iranske vlasti nisu mogle kontrolirati i preko kojeg su oporba i njihovi pobornici nastavljali u stvarnome vremenu komunicirati i obavještavati svjetsku zajednicu. O važnosti Twitter-a u tim kriznim vremena govori činjenica da je Američka vlada zatražila Twitter da odgodi zastoj u usluzi zbog održavanja kako se ne bi poremetio tok informacija (Britannica 2021.).

Otada Twitter osnažuje svoju ulogu kao društveni alat za slobodan protok informacija. Nadalje se koristi i kao alat za podupiranje aktivizma diljem svijeta i za koordiniranje napora pomoći u katastrofama. Danas broji preko 300 milijuna aktivnih korisnika mjesečno i jedna je od najvažnijih društvenih mreža diljem svijeta (Statista 2021.).

4.3.10. Instagram

Instagram je jedna od najpopularnijih društvenih mreža danas, sa preko milijardu aktivnih korisnika. Kao i mnoge popularne društvene mreže, njeni počeci su skromni. Instagram izlazi u javnost 6. listopada 2010. godine, no sama ideja nastaje mjesecima prije. Osnivači Kevin Systrom i Mike Krieger htjeli su kreirati aplikaciju koja bi olakšala kreiranje dobrih fotografija na pametnim telefonima onoga vremena (koji nisu bili poznati po moći razlučivosti svojih ugrađenih kamera) i dijeljenje tih fotografija na više platformi (Frier 2020.).

Kako bi to što prije i jednostavnije postigli, odlučili su se na maksimalnu širinu slike od 640 piksela (temeljem prikaza tadašnjih iPhone uređaja), sa kvadratastim omjerom slike. Korisnici bi učitali svoju sliku i dobili na izbor nekoliko filtera koje mogu primijeniti kako bi uljepšali sliku. Nakon toga se korisnik može još dodatno odlučiti podijeliti objavljenu sliku na Twitter ili Facebook (što ima kao pozitivnu posljedicu dostizanje još šireg tijela potencijalnih korisnika sa tih platformi) (Frier 2020.).

Uz pomoć Jack Dorseyjeve promidžbe preko Twitter-a, na dan izdanja Instagram je postao najpopularnija aplikacija u kategoriji kamera na Apple-ovom online dućanu. Iako isprva dostupan samo na iPhone uređajima, već prvi dan Instagram je koristilo 25.000 korisnika. Unutar prvog tjedna, taj broj skočio je na 100.000. Godinu dana kasnije, u listopadu 2011. godine Instagram je imao preko 10 milijuna korisnika. Ovakav iznimno brzi rast predstavljao je neprestani izazov za osnivače i nekolicinu zaposlenika koje su putem zaposlili (Frier 2020.).

Systrom i Krieger su probleme i izazove rješavali putem kako su se pojavljivali i u letu ih raspoređivali po prioritetu. Strategija se pokazala relativno uspješnom i kroz godine su uspjeli održati servis stabilnime. Ispadi poslužitelja su bili uobičajena stvar, posebice dok su vrlo poznate i popularne ličnosti objavile nešto na svoj Instagram profil, što je odmah izazvalo opterećenje sustava (Frier 2020.).

Sve u svemu, Instagram se uspio održati na tržištu, no unatoč svojoj popularnosti nije postizao prihode budući da nisu imali poslovni model koji monetizira platformu (slično kao što Twitter isprva nije). Iz tog razloga su bili vrlo ovisni o investitorima. No, bili su nov i poželjan proizvod na tržištu te su se investitori redali. Dapače, postigli su toliku snažnu tržišnu poziciju da su se Twitter i Facebook nadmetali u tome tko će ih kupiti. Twitter ih je mjesecima nagovarao na stjecanje i na kraju ponudio 50 milijuna dolara u investicijama. No, kako je Instagram počeo predstavljati prijetnju Facebooku, Mark Zuckerberg je osobno kontaktirao Systroma kako bi ispregovarali preuzimanje Instagrama u vrijednosti od čak 1 milijardu dolara, što je bila najvrijednija kupnja u to vrijeme. Jedna od ključnih stavki preuzimanja bio je uvjet da se Instagram i dalje vodi kao nezavisno poduzeće (Frier 2020.).

Kada je Facebook preuzeo Instagram, počeo je integrirati svoju reklamnu platformu. Poslovica kaže da kada je proizvod besplatan, korisnik je zapravo proizvod. Društvene mreže danas ponajviše zarađuju kroz prikazivanje reklama svojim korisnicima. Što je reklamiranje preciznije i primjerenije određenoj demografiji, to su profiti veći. Iz tog razloga Facebook je razvio opsežnu platformu i tehnologije za reklamiranje. Time je Facebook samo proširio svoje tržište i mogućnosti zarade (Frier 2020.).

2018. Kevin Systrom odstupio je sa svoje pozicije kao direktor poduzeća, prvenstveno zbog sve većih nesuglasica sa Markom Zuckerbergom i njegovim sve agresivnijim uplitanjem u poslovanje Instagrama. Nakon toga, Instagram postaje više manje Facebookov proizvod i preusmjerava se više na zaradu iz reklama. Dapače, danas Facebook potencijalno više zarađuje na temelju Instagrama negoli vlastite društvene mreže. To se utoliko više vidi i u samoj činjenici da je prošle godine uvedena mogućnost prodaje proizvoda preko Instagrama (Frier 2020.).

Sa Instagramom vjerojatno po prvi puta se javlja jedan od modernih fenomena društvenih mreža u tržišnome sustavu - takozvani *influenceri*. Budući da je Instagram prvenstveno vizualna platforma, preko nje se lako može graditi željena estetika. Korisnici, potencijalni potrošači, mogu *konzumirati* određeni dojam koji se želi izazvati u njima, što olakšava prodaju proizvoda. Također, kao društvena mreža, omogućuje direktnu interakciju sa korisnicima. Profili koji imaju veliki broj pratitelja predstavljaju tako snažnu marketinšku silu i mogu sponzorirati tuđe proizvode. Za to reklamiranje dobiju novac, bilo fiksna količina ili preko poveznice za svaku prodaju proizvoda. Na temelju toga nastalo je veliko tržište i određeni korisnici mogu više nego

ugodno živjeti temeljem toga. Više nisu samo zvijezde i poznate ličnosti koje reklamiraju tuđe proizvode. Procjenjuje se da je *influencerska* industrija danas teška nekoliko milijardi dolara (Brooks 2019.).

4.3.11. TikTok

TikTok je danas jedna od najpopularnijih društvenih mreža među mlađom demografijom (ispod 24 godine), sa oko milijardu korisnika. Radi se o aplikaciji i društvenoj mreži za dijeljenje kratkih video uradaka do 3 minute (nekoć samo 15 sekundi). Kao svoju misiju navode "inspiriranje kreativnosti i donošenje veselja" (TikTok 2021.).

Izvorno je izašao samo na kinesko tržište u rujnu 2016. godine pod nazivom A.me, ali se ubrzo preimenuje u Douyin. Bivajući svijesni činjenice da Kina čini samo oko petinu online korisnika svijeta, vlasnici TikTok-a su se fokusirali na širenje na strana tržišta (Wang i dr. 2020.). Krajem 2017. godine Douyin se lansira za međunarodno tržište pod poznatim imenom TikTok. Ubrzo nakon toga, tvrtka se spaja s drugom poznatom kineskom platformom *Musical.ly* kako bi privukli i prebacili njihovu mlađu demografiju u SAD-u. Time je TikTok postao doista globalna platforma (Lee 2018.).

Način na koji je TikTok stekao svoju popularnost i proboj na domaćem i međunarodnom tržištu posljedica je više čimbenika i strategija. U vrijeme lansiranja originalne kineske verzije, već je postojalo tržište za takve aplikacije te ona nije bila inovativna i nova pojava. Prije svega, TikTok se javlja u dobro vrijeme kada visoki postotak stanovništva koristi pametne telefone i sa mijenom generacija što obećava daljnje održavanje tog tržišta. Nadalje, u to vrijeme raste popularnost telefona sa većim ekranima i napreduje mrežna infrastruktura, što pospješuje konzumaciju i dijeljenje video sadržaja. U ranim fazama izlaženja na tržište, autori su često i redovito ažurirali aplikaciju kako bi bila jednostavnija i atraktivnija za korištenje, optimizirali ju kao i dodavali nove efekte, filtere i drugi sadržaj (Wang i dr. 2020.).

No, pravu pozornost su počeli dobivati tek kada je jedna kineska zvijezda proslijedila video s Douyin vodenim žigom na svom Weibo profilu. Objava je dobila desetine tisuća "lajkova" i tisuće puta bila dalje proslijeđivana. Nakon toga je TikTokov marketinški tim shvatio moć koju zvijezde i poznati imaju u angažiranju svojim obožavatelja te su tako krenuli u agresivniju kampanju putem pomoći različitih kineskih zvijezda kao i putem sponzoriranja različitih TV kanala. Tada je zaista započeo TikTokov rast (Wang i dr. 2020.).

Ulazeći u druga tržišta, prilagođavali su se njihovim karakteristikama i sukladno svojoj strategiji, nudili su globalne proizvode uz lokalan sadržaj. Drugim riječima, nudili bi različite *hashtag*-ove vezane uz sadržaj lokalan tom tržištu, kao što su razni festivali i događaji te bi prevodili sadržaj same aplikacije na njihov jezik. Od 2018. godine, TikTok je imao preko 500 milijuna mjesečno aktivnih korisnika (Wang i dr. 2020.). Danas se može hvaliti sa preko milijardu mjesečno aktivnih korisnika, što plasira TikTok među vrhom najpopularnijih društvenih mreža (Doyle 2021.).

5. Monetizacija

Danas se živi u tzv. *digitalnoj ekonomiji* (također zvanoj i *internet ekonomijom*). Poslovanje se više ne mora vršiti isključivo putem staroga *bricks and mortar*, odnosno fizičkog, pristupa i sve više poslovnica je kroz zadnja dva-tri desetljeća barem stvorilo neko online prisustvo, ako ne i preselilo svoje poslovanje online. U svakom slučaju, digitalne tehnologije se usvajaju diljem svih industrija i gospodarskih grana, što je imalo dalekosežne posljedice za njihovu efikasnost i način vršenja posla. Internet, odnosno digitalna, ekonomija ima veći godišnji rast negoli tradicionalni fizički sektor i taj rast se neće usporiti već samo ubrzati i time produbiti jaz između onih koji iskorištavaju blagodati digitalnih tehnologija za svoje poslovanje (Skilton 2015.).

Uz današnje intenzivno međupovezivanje uređaja i identiteta kroz više platformi, nastaje mrežni učinak koji obogaćuje vrijednost podataka i pruža dodanu vrijednost svim korisnički interakcijama, bile one među sobom ili sa poduzećima i uslugama. U toj interakciji nastaju prilike i uvjeti za razvoj novih poslovnih i društvenih modela monetizacije (Skilton 2015.). Prema (A. Jablonski i M. Jablonski 2021.), budući da je poslovanje postalo puno socijalnije, danas kao poduzetnik vaš prvi korak ne bude nužno zadobivanje mušterija, koliko možda izgradnja zajednice i istraživanje prilika za monetizacijom poslovanja. Dapače, neki smatraju da danas bez prisustva zajednice moderni poslovni modeli ne mogu biti efikasni (A. Jablonski i M. Jablonski 2021.).

Jedan od najimpresivnijih primjera digitalne ekonomije čine upravo društvene mreže. Neke od danas najpoznatijih i najuspješnijih digitalnih tvrtki se okreću oko društvenih mreža, ili malo opširnije rečeno - platformi. Poznato je da kompanije poput Facebook-a, Instagram-a, Twitter-a i TikTok-a zarađuju milijunske, dapače iznose od više milijardi dolara godišnje. To nije naprosto zahvaljujući njihovim poslovnim modelima, nego i monetizacijskim modelima (što se međusobno razlikuje) (A. Jablonski i M. Jablonski 2021.).

Monetizacija ima više definicija, ovisno o osobi koju pitate. Laički shvaćeno, radi se o načinu kako poduzeće određenu imovinu pretvara u monetarnu vrijednost. Razlikuje se od modela prihoda jer on opisuje način na koji novac teče od mušterije do poduzeća (iako mogu postojati preklapanja).

- Proces putem kojeg se nešto pretvara u novac (likvidacija). Pretvaranje imovine kao što su poslovni model ili podaci u novac (A. Jablonski i M. Jablonski 2021.).
- Čin izmjene informacijskih ponuda u zakonska sredstva plaćanja ili nešto drugo prividno jednake vrijednosti (A. Jablonski i M. Jablonski 2021.).
- Sposobnost generiranja financijskog imetka kroz strategije i metode čija je svrha pretvoriti vještine, odnose, znanje, imovinu, proizvode i servise u novčanu vrijednost (Skilton 2015.).

Slijedi tablični popis metoda monetizacije koje je moguće primijeniti nad digitalnim proizvodom (valja napomenuti da popis nije ni po čemu iscrpan).

Ime metode	Opis
Reklamiranje	Najčešći oblik zarade za web stranice. Prikladan za multimedijske stranice, blogove i druge informacijske usluge. Cilj je prikazati reklame na stranici, koje se mogu javljati u više oblika: • Tekstualne reklame • Video reklame • Ciljane reklame (engl. *Targeted advertising*) • Umetnute reklame (engl. *Interstitial ads*) • Natpisne reklame (engl. *Banner ads*) • I tako dalje...
Sponzorstvo	Način reklamiranja specifičnih brendova i poduzeća, koji bivaju predstavljeni najčešće svojim logotipom negdje na korisničkom sučelju. Ti brendovi plaćaju neku svotu.
Plaćeni sadržaj i advertorijali	Reklamni sadržaj napisan u stručnome i naizgled objektivnome i informativnome formatu, sa krajnjim ciljem da se korisnika navede na kupovinu nekog proizvoda.
Udruženi marketing	Posebna vrsta partnerskog reklamiranja gdje se na stranici prikazuju poveznice partnerskih proizvoda, za čiju kupovinu će stranica onda dobiti proviziju. Primjerice, određena marka tipkovnica se reklamira kao preporučena na vašoj stranici i za svaku postignutu kupovinu preko vaše poveznice dobivate proviziju u obliku 15% prodajne vrijednosti.
Donacije	Jedan od najjednostavnijih načina monetiziranja, donacije su dobrovoljnja novčanja davanja od strane samih korisnika neke stranice ili platforme. Obično su motivirana dobrom voljom ili željom da se podrži bilo nekog kreatora/autora, voljenu platformu ili drugo. Jedna od danas najpopularnijih platformi za doniranje je Patreon.
Stjecanje	Direktna kupovina poduzeća ili prava na proizvod ili intelektualno vlasništvo, obično nakon dužeg pregovaranja. Često kod start-upova.

Ime metode	Opis
Inicijalna javna ponuda (IPO)	Privatna (ili državna firma) otvara svoje dionice javnosti i daje ih na ponudu svim interesiranim ulagateljima, kao i zaposlenicima toga društva i drugim partnerima. Na taj način može osigurati novo financiranje ukoliko se iskaže dovoljno veliki interes i što bude veći interes i ponuda za dionice, to će njihova vrijednost više rasti i samim time će poduzeće više vrijediti. Time će ponovno privući dodatnu pažnju što može rezultirati dodatnim financiranjem. Popularno za vrijeme Web2.0 eksplozije (kao što se može vidjeti sa tvrtkama kao što su Facebook, Twitter i Instagram, koji su svi imali IPO). Danas manje popularno.
Investiranje i partnerstvo	Popularno za mlade "startup" tvrtke, koje dobivaju u mahovima financiranje putem tzv. *rizičnog kapitala* (engl. *venture capital*). To su dugoročne investicije kroz više godina sa krajnjim ciljem višestrukog povrata investicije pri kasnijoj IPO ili stjecanju firme. Investitori često postaju partneri tvrtke, vršeći određenu mentorsku ulogu i redovito zahtijevajući povratne informacije o razvoju tvrtke.
Premium članstvo, pretplate, licenciranje	Plaćena verzija inače besplatne usluge s dodatnim pogodnostima ili punim skupom značajki. Primjer je Google Drive koji u besplatnoj inačici nudi 15GB prostora pohrane, dok sa plaćanjem nudi 100GB, 200GB ili 2TB pohrane.
Virtualna (Digitalna) dobra	Slično unutar-aplikacijskim kupovinama, primjereno najviše za masivne online igre za više igrača kao što su Second Life i EVE Online. Radi se o predmetima/proizvodima najčešće kozmetičke prirode koje igrači kupuju stvarnim novcem ne bi li tako stekli viši društveni status unutar same igre. U slučaju EVE Online može se raditi o stotinama tisuća dolara vrijednosti digitalnih dobara stjecanih zajedničkim naporom kroz više godina.

Ime metode	Opis
Prodaja podataka	Danas vruća tema zbog problema privatnosti i anonimnosti podataka. Online ekonomija se uvelike temelji na reklamama, a reklame da bi bile efikasne moraju biti informirane jer se korisnike interneta bombardira reklamama. Svi su upoznati sa pojmom "algoritam". Algoritam je, takoreći, pojeo svačije živote, sveprisutan je i nekada se čini da sve diktira. Sadržaj koji se konzumira na internetu i na društvenim platformama uslužuje uglavnom nekakav algoritam, na temelju onoga što uči iz nečijeg ponašanja i iskazanih interesa, odnosno na temelju toga što se "lajka". Ovisno o sofisticiranosti algoritma, on može mjeriti i koliko se dugo ljudi zadržavaju na nekome sadržaju, a taj sadržaj će imati oznake koje ga opisuju i koje taj algoritam može onda međusobno asocirati i pretpostaviti dalje kakav sadržaj će nekome biti interesantan. Cilj je zadržati nečiju pažnju što dulje jer time se povećava šansa da će osoba vidjeti više reklama. No tu priča ne staje, jer te reklame su također informirane i upogonjene nekim algoritmom. Taj algoritam također poznaje informacije o nečijem ponašanju i interesima te tako servira reklame o proizvodima koji imaju veću vjerojatnost da će uloviti i poklopiti se sa nečijim interesima, čime se povećava vjerojatnost uspješnog angažiranja korisnika i konverzije pogleda reklama u kupovinu. Podaci o korisnicima od iznimne su važnosti za online ekonomiju danas, a društvene mreže su jedno od najplodnijih tla za sakupljanje raznoraznih informacija o korisnicima, jer oni grade svoje profile, pokazuju sa kime su povezani, a te osobe imaju opet svoje informacije, pa je moguće korelirati poveznice, zajedničke interese i tako dalje. Radi se o vrijednim marketinškim podacima i na njihovoj prodaji tvrtke poput Facebook-a i Google-a ostvaruju veliku zaradu.
Freemium	Omogućen je besplatni pristup usluzi, no sa ograničenim značajkama i mogućnostima te se korisnike navodi na izdvajanje novaca kako bi se stekao pristup punom skupu mogućnosti servisa. Što može biti bilo preko jednokratne kupovine pune verzije, mjesečne pretplate ili članarine ili unutaraplikacijskih kupovina.
Unutaraplikacijska kupovina (engl. *In-App purchases*)	Uglavnom primjenjivo za video igre, danas popularan i vrlo efikasan način izvlačenja dodatnog novca iz igrača nuđenjem digitalnih dobra unutar video igara koje je moguće kupiti. Ta dobra mogu vršiti čisto kozmetičku funkciju ili čak davati prednost igračima i ubrzati njihov napredak (praksa na koju se negativno gleda i naziva *pay-to-win*). Najčešće prisutno u popularnim besplatnim društvenim online igrama.

Ime metode	Opis
Skupno financiranje (engl. *crowdfunding*)	Danas iznimno popularan način stjecanja financija za nove projekte (ali i za osobne potrebe kao što su plaćanje zdravstvenih računa ili operacija, npr.) u kojem se traže mali iznosi, ali od velikog broja ljudi (stotine, tisuće). Objavljuju se informacije o projektu, poželjno što detaljnije uključujući ciljeve financiranja i strategije nagrađivanja (najčešće se nude rangovi nagrada ovisno o iznosu donacije). Kampanja obično traje par mjeseci te, ovisno o odabranoj platformi za financiranje, novac bude isplaćen jedino ako postigne cjelokupni cilj financiranja ili će biti isplaćen parcijalno u postignutome iznosu. Nakon uspješne kampanje, kreće se u produkciju i izvršenje, putem informirajući donatore o novim razvojima i događanjima. Ukoliko očekivani proizvod na kraju ne bude isporučen, moguće je tražiti natrag novac, no to zna biti mukotrpno i nažalost nekada novac ne bude nikad vraćen nazad.
Dodatni digitalni sadržaj (engl. *DLC - Downloadable content*)	Primjenjivo za video igre - nekoć se dodatni sadržaj razvijao u obliku proširenja (engl. *expansion pack*) i prodavao po cijeni igre. Danas je rijetkost i zamijenili su ih "dodatni skidivi sadržaji", koji su na lošem glasu. Posebice kada su dostupni na isti dan izbacivanja video igre, jer se onda vjeruje da se radi o sadržaju koji je trebao biti dio standardnog proizvoda, no koji su selektivno izdvojeni ne bi li ga mogli dodatno naplatiti. DLC-jevi mogu biti mini-proširenja ili čak dodatak samo jedne stvari kao primjerice jedna nova "mapa" ili razina.

Tablica 2: Popis metoda monetizacije

Izvor: (A. Jablonski i M. Jablonski 2021.)

Kao što se može vidjeti, postoji mnogo načina kako monetizirati proizvod. Na koje metode će se netko odlučiti ovisi o njihovom poslovnom modelu i očekivanjima za budućnost. Monetizacija je uključena u namjerama poslovnog modela i nije uvijek moguće, niti svrsishodno, efikasno monetizirati u početku. Često je potrebno više godina napora u građenju plodnog tla kako bi se moglo krenuti u monetizaciju (primjerice Twitter prvih godina nije imao nikakav monetizacijski model).

Kako bi korisnici prihvatili monetizacijski model, odnosno kako ih ne bi odvratio od korištenja proizvoda, proizvod mora nuditi adekvatnu korist i dodanu vrijednost korisnicima koja će otkloniti negodovanje (ako ga uopće ima). Naravno, za efikasnu monetizaciju, potrebno je izgraditi veliku zajednicu.

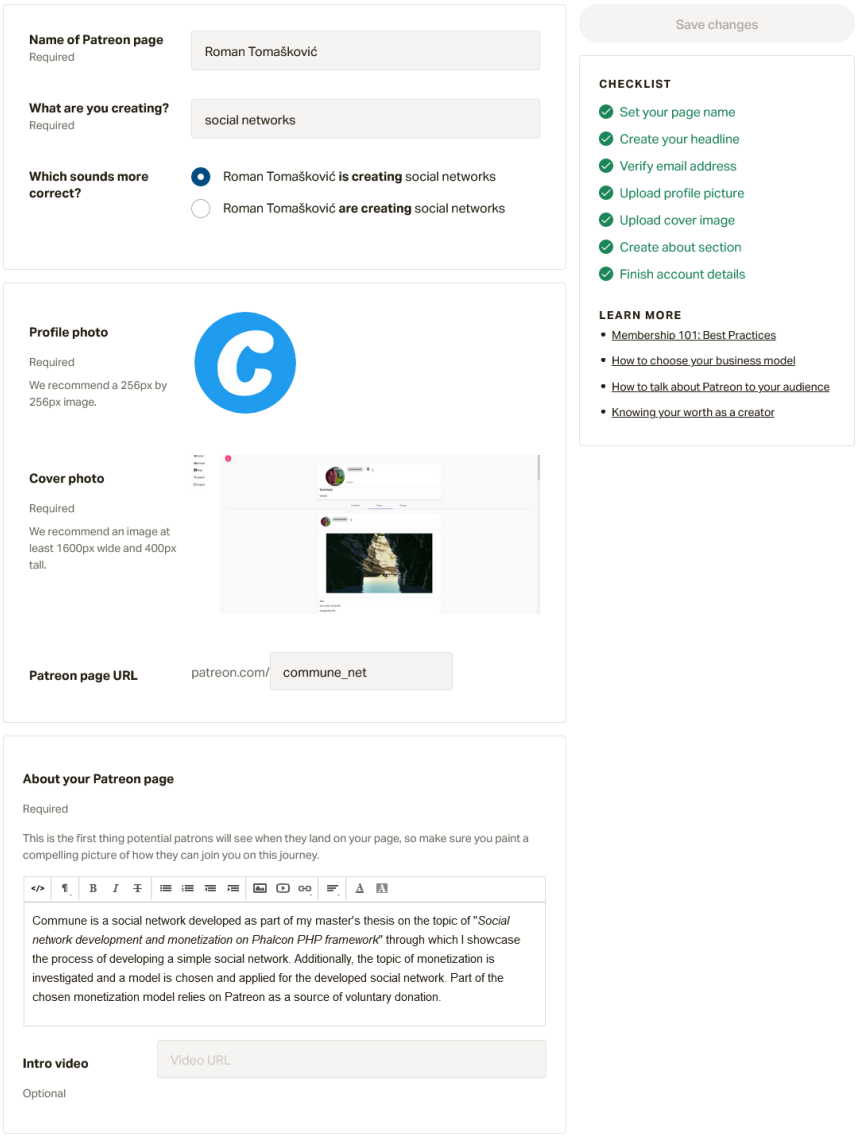
U sklopu projekta ovog rada, s obzirom na osobna uvjerenja, autor se odlučio na monetizacijski model kombiniranja donacija putem Patreona i uvođenja pretplate s dodatnim pogodnostima.

5.0.1. Patreon

Patreon je platforma namijenjena svim kreativnim profesionalcima koji kreiraju sadržaj kako bi jednostavnije monetizirali svoju publiku i pratitelje. Kako navode na svojim stranicama (Patreon 2021.): "*Preko Patreona možete biti plaćeni za stvari koje već stvarate (bile one web stripovi, video uradci, pjesme, itd). Pratitelji plaćaju mjesečno ili po objavi koju izdajete*".

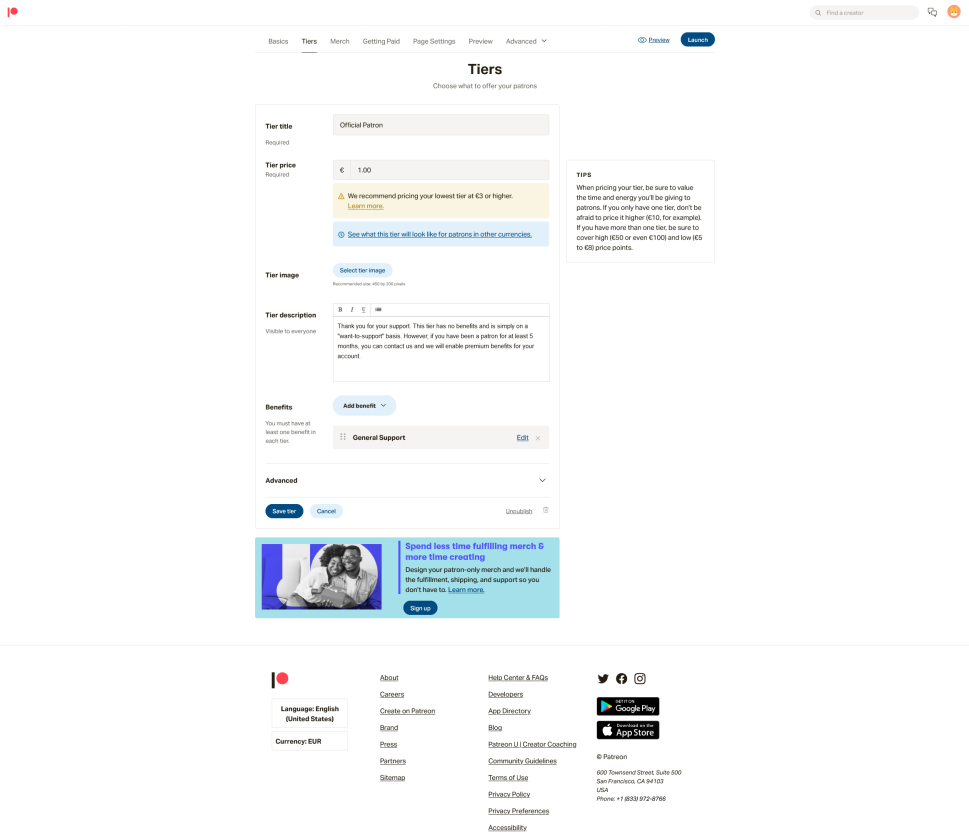
Na Patreonu kreativni profesionalci i drugi kreatori različitog sadržaja stvaraju svoju stranicu sa osnovnim podacima o tome tko su i što rade, na koje načine obožavatelji mogu podupirati kreatora i koje su sve razine pretplata sa pripadnim povlasticama i mogućnostima. Određene razine pretplata mogu biti i ograničene dostupnosti za samo određen broj ljudi. Takve pretplatne razine obično donose posebne pogodnosti kao primjerice 1 na 1 razgovor sa osobom, prihvaćanje prijedloga i tako dalje.

Na slici 28 prikazan je obrazac za podešavanje Patreon stranice. Nudi iscrpni popis opcija kao i spisak stvari koje korisnik mora odraditi prije objavljivanja stranice.



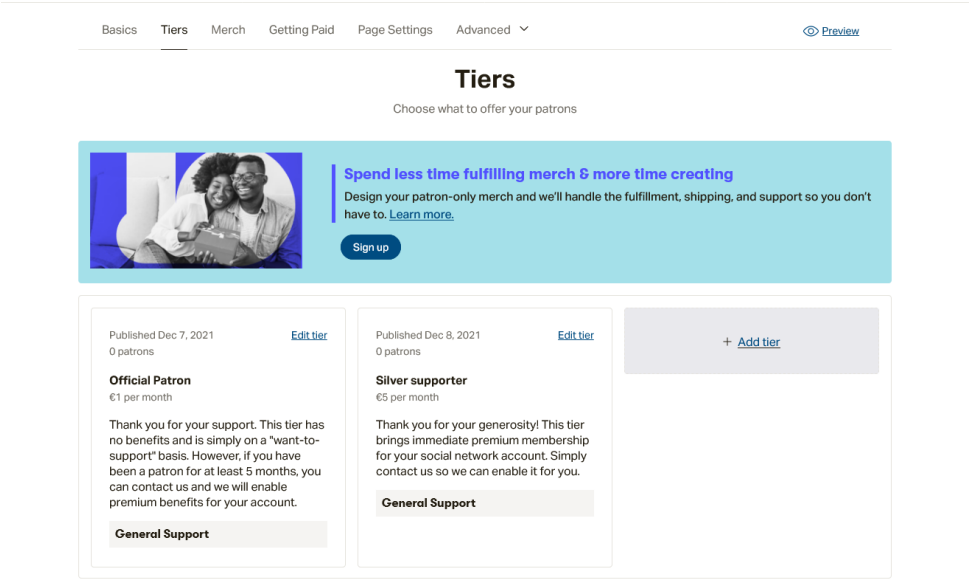
Slika 28: Podešavanje Patreon stranice

Također je potrebno dodati razine potpore koje kreator podržava. Razine se definiraju proizvoljno, no Patreon nudi predloške za različite scenarije i tipove kreatora.



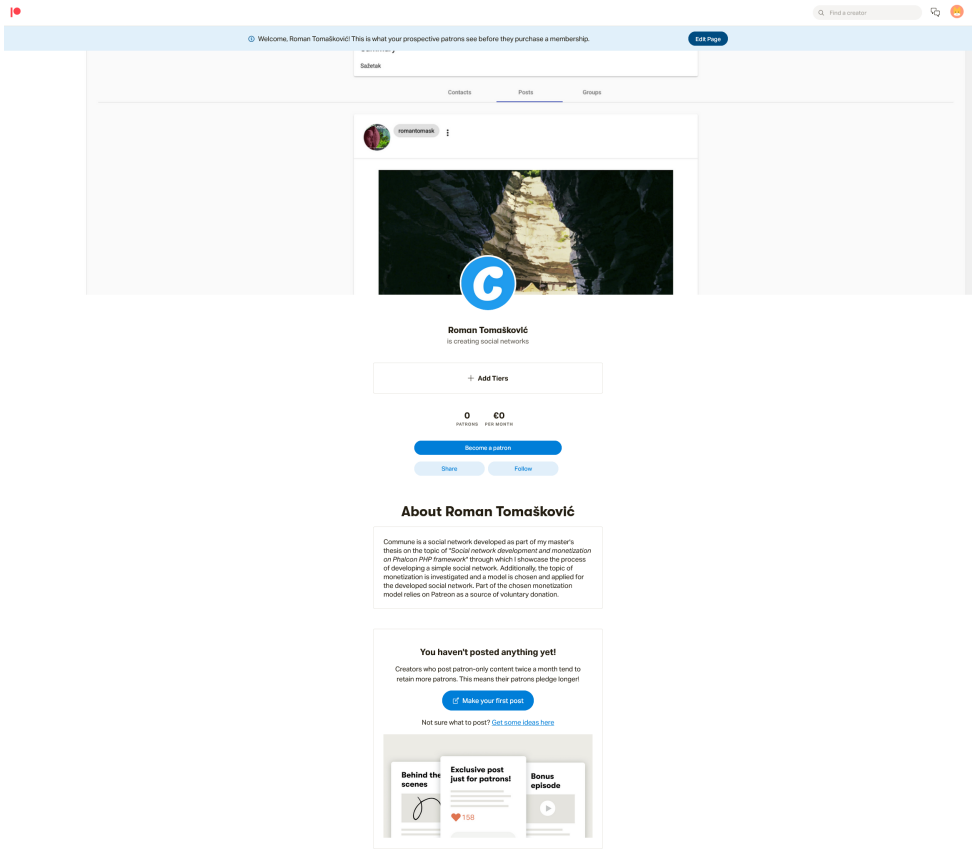
Slika 29: Postavke Patreon razina

Slika 30 prikazuje sve razine potpore koje su definirane za potrebe projekta. Vrlo je jednostavno, dvije su razine. Prva razina košta $1 mjesečno i sa svojom niskom cijenom nudi pristupačan i jednostavan način potpore. Druga razina košta $5, kao i kupovina premium pretplate na samoj društvenoj mreži te nudi odmah i premium pogodnosti.



Slika 30: Razine podrške

Na slici 31 prikazana je gotova Patreon stranica kakvu korisnici mogu posjetiti.



Slika 31: Pretpregled Patreon stranice

6. Specifikacije i funkcionalni zahtjevi

Niti jedan projekt većeg obujma se ne izrađuje na pamet bez barem nekakvog prethodnog razmatranja. U najmanju ruku, valja predodrediti koje će biti minimalne funkcionalnosti i mogućnosti projekta i što se želi izraditi. Stoga slijedi funkcionalna specifikacija koja opisuje mogućnosti društvene mreže koja se u ovome radu opisuje.

6.1. Funkcionalna raščlamba projekta na visokoj razini

- Prijava
  - zaboravljena lozinka
  - 2FA (višefaktorska prijava - prijava koja koristi više vjerodajnica)
- Registracija
- Korisnički profil
  - Popis i pregled kontakata
  - Popis i pregled učlanjenih grupa
  - Popis i pregled objava
  - Postavke
- Korisničke grupe
- Oznake
- Pregledavanje sadržaja i pretraga
- Korisnički chat

6.2. Detaljna raščlamba funkcionalnosti

Slijedi detaljna raščlamba i opisi komponenata funkcionalnosti koje su bile gore navedene.

6.2.1. Registracija

Registracija predstavlja inicijalno učlanjenje korisnika u društvenu mrežu. Prilikom registracije, korisnik unosi ime i prezime, kao i *nickname* (odnosno nadimak), e-mail adresu te svoju lozinku. Na korisničku e-mail adresu stiže e-mail s poveznicom verifikacije korisnika, nakon čega korisnik dobiva pristup društvenoj mreži. Otvara mu se korisnički profil.

6.2.2. Prijava

Korisnici koji su se registrirali putem prijave se vraćaju na društvenu mrežu. Sastoji se od jednostavnog unosa e-mail adrese i korisničke lozinke. Ako je unesena ispravna kombinacija, korisniku se omogućuje pristup mreži.

Zaboravljena lozinka Ako je korisnik zaboravio lozinku koju je odredio prilikom registracije, korisnik može na stranici prijave odabrati opciju "Zaboravljena lozinka". Korisniku se prikazuje novi pogled gdje potvrđuje da mu se pošalje na e-mail adresu poveznica za promjenu lozinke. Alternativno može promijeniti e-mail adresu na koju se ta poveznica bude poslala.

2FA Naravno, u današnje doba informatizacije i dijeljenja podataka, sve je važnije brinuti se za vlastitu sigurnost i privatnost. U sklopu raznih korisničkih postavki bit će, između ostaloga, moguće uključiti višefaktorsku prijavu koja će koristiti pametni telefon korisnika kao taj drugi faktor u prijavi. Prilikom prijave korisnik će biti upitan za privremeni vremenski kôd koji mora unijeti uz postojeće korisničke podatke.

6.2.3. Korisnički profil

Svaki korisnik koji se registrira na mrežu dobiva svoj korisnički profil koji predstavlja tog korisnika i njegovo sudjelovanje u mreži. U svojoj osnovi, korisnički profil se sastoji od nadimka i profilne slike korisnika. Dalje, svačiji profil predstavlja cjelokupnu stranicu te koju drugi korisnici mogu posjećivati.

Objava sadržaja Korisnički profil ujedno predstavlja i samog korisnika te korisnik na svom profilu objavljuje različite sadržaje po svom nahođenju, bili oni tekstualnog, slikovnog, zvučnog ili video formata te će se oni prikazivati u objavi u obliku jednostavne "galerije". Po volji, korisnik može odabrati hoće li objava biti vidljiva svim korisnicima, samo njihovim kontaktima ili samo unutar grupe unutar koje je objavljena (ako je objavljena kao dio grupe).

Kontakti Što su na Facebooku prijatelji ili na Instagramu pratitelji, ovdje su to kontakti. Kontakti predstavljaju sve druge korisnike s kojima se osoba umrežila. Uz kontakte, moguće je vidjeti i popis stranica, odnosno grupa kojima je korisnik pridružen i koje prati.

Grupiranje kontakata i privatnost Kada korisnik želi objaviti neki novi sadržaj, korisnik može birati hoće li on biti vidljiv svim korisnicima društvene mreže (dakle i van kruga kontakata), samo svojim kontaktima ili samo unutar grupe kojoj objava pripada. Korisnici koji su izuzeti iz neke objave istu neće nikada moći vidjeti, osim ako se originalni autor izmijeni postavke privatnost.

Postavke Svaki korisnik, naravno, može podesiti i svoje računske postavke. Pod to spadaju promjena lozinke, primarne e-mail adrese ili nadimka, deaktivacija odnosno brisanje korisnič-

kog računa i slično. Tu također korisnici uključuju ili isključuju višefaktorsku prijavu i mogu provesti naplatu kako bi postali "premium" korisnici s dodatnim pogodnostima.

6.2.4. Korisničke grupe

Naravno, nije društvena mreža ako korisnici nisu u mogućnosti kreirati razne grupe i u njih se učlanjivati.

Svaki korisnik može kreirati grupe bez restrikcija. Kao osnivač grupe, postaje administrator grupe i može pozivati druge korisnike da se učlane u svoju grupu. Korisničke grupe nude prostor za dijeljenje sadržaja, provođenje diskusija i ležerno čavrljanje među članovima. U neku ruku, grupa tako čini istovremeno i malu zajednicu. Grupe mogu biti općenite, ili tematski određene.

Administratori grupe mogu jedini mijenjati njene postavke i prilagođavati ih (ime grupe, profilna slika i naslovna slika grupe).

6.2.5. Oznake

U načelu, sav sadržaj na mreži je, ovisno o postavkama privatnosti, dostupan svim ostalim članovima mreže. Dakle moguće je pretraživati sav sadržaj. Naravno, ukoliko bi postojalo puno članova, postalo bi vrlo teško pretraživati i navigirati kroz veliku količinu sadržaja koji članovi stvaraju. U te svrhe se uvodi sustav označivanja sadržaja. Oznaka je već poznati mehanizam kojeg se viđa na mrežama kao što je Twitter i Instagram te je njegova svrha svojevrsno "kategoriziranje" sadržaja.

Sav sadržaj koji korisnici stavljaju na mrežu je tako moguće "označiti". Postoji više vrsta oznaka. Temeljna je klasična tekstualna oznaka formata #oznaka.

Postoje neka ograničenja što se tiče označivanja - za tekstualne oznake postoji ograničenje od maksimalno 5 oznaka. Razlog takvom ograničenju je izbjegavanje prakse nekih korisnika drugih mreža gdje se na objavu doda 20+ različitih oznaka, koje više niti nemaju veze sa samom objavom, a sve u svrhu iskorištavanja "algoritma" te mreže kako bi objava imala što više šanse prikazati se drugim korisnicima. Svaki sadržaj bi trebalo biti moguće adekvatno opisati s 5 tekstualnih oznaka.

6.2.6. Pregledavanje i pretraživanje sadržaja

Svaka društvena mreža se puni sa sadržajem kojeg korisnici generiraju. Osnovna stranica koju svaki korisnik vidi kada se prijavi u mrežu je "tok" sadržaja kojeg drugi korisnici kreiraju. Sadržaj je prikazan kronološki od najnovijeg prema najstarijem. Ovisno o postavkama privatnosti korisnika, neće sav sadržaj uvijek biti vidljiv. Sadržaj koji se prikazuje može se rastaviti na nekoliko kategorija:

- Korisnici

- Oznake
- Grupe

Svaki korisnik je u mogućnosti pretraživati svaku od tih "kategorija" sadržaja.

6.2.7. Chat

Naravno, društvena mreža nudi mogućnost tekstualne komunikacije među njenim korisnicima. Chat je moguće inicirati s bilo kojim korisnikom mreže, ali jedino ako ste međusobno dodani kao kontakti.

Chat omogućuje tekstualnu komunikaciju među korisnicima. Svaki chat ima svoju povijest koju korisnik može pregledavati.

7. Tehnička implementacija projekta

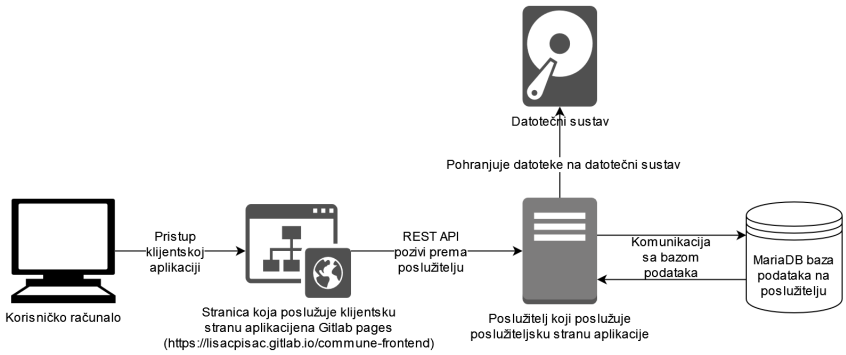
U ovome poglavlju slijedi opis konkretne implementacije rješenja društvene mreže kao praktičnog dijela rada. Biti će prikazani arhitektura rješenja, dijagrami klasa, podatkovni model (ERA dijagram) baze podataka te će se pojedinačne funkcionalnosti aplikacije opisivati tekstualno, vizualno i sa popratnim isječcima programskog kôda.

Kako se radi o "full-stack" aplikaciji, izvorni kôd dostupan je na dva git repozitorija:

- Klijentska strana - https://gitlab.com/LisacPisac/commune-frontend
- Poslužiteljska strana - https://gitlab.com/LisacPisac/commune-backend

7.1. Arhitektura sustava

Kao što je spomenuto, društvena mreža realizirana je kao "full-stack" web aplikacija, odnosno sadrži klijentski i poslužiteljski dio. Klijentski dio je implementiran pomoću Angular razvojnog okvira te se poslužuje pomoću GitLab pages funkcionalnosti koja omogućuje posluživanje statičnoga web sadržaja na temelju sadržaja samog git repozitorija. Korisnici pristupaju aplikaciji putem web preglednika. Aplikacija komunicira sa REST API-jem i tako ostvaruje raznolike funkcionalnosti. Poslužiteljska strana koja poslužuje REST API zauzvrat za većinu poziva komunicira dalje sa bazom podataka (MariaDB) kako bi trajno pohranjivala podatke koji opisuju sadržaj na društvenoj mreži. Kako su korisnici u mogućnosti učitavati datoteke u društvenu mrežu, potrebno je i trajno pohranjivati te datoteke negdje. Te datoteke se trenutno pohranjuju na datotečni sustav samog poslužitelja.



Slika 32: Arhitektura sustava implementirane društvene mreže

Razmatranja za budućnost Iskazana arhitektura je relativno jednostavna te dopušta samo ograničeni rast društvene mreže. Kako se s vremenom povećava količina sadržaja na mreži, učitavanje objava i datoteka postajati će sve sporije, što zahtijeva ažuriranje arhitekture.

Kako bi se unaprijedila, u budućnosti se arhitektura može proširiti dodavanjem predmemorijskog sloja za pohranjivanje koristeći memorijsku bazu podataka Redis (ili alternativno Memcached), koji može, ali i ne mora biti posluživan na istome poslužitelju gdje se nalazi REST

API. Takav predmemorijski sloj dozvoljavao bi brzo dohvaćanje novijeg sadržaja (većina korisnika vjerojatno ne bi pregledavala stariji sadržaj), dok bi se stariji sadržaj dohvaćao sa sporije pohrane.

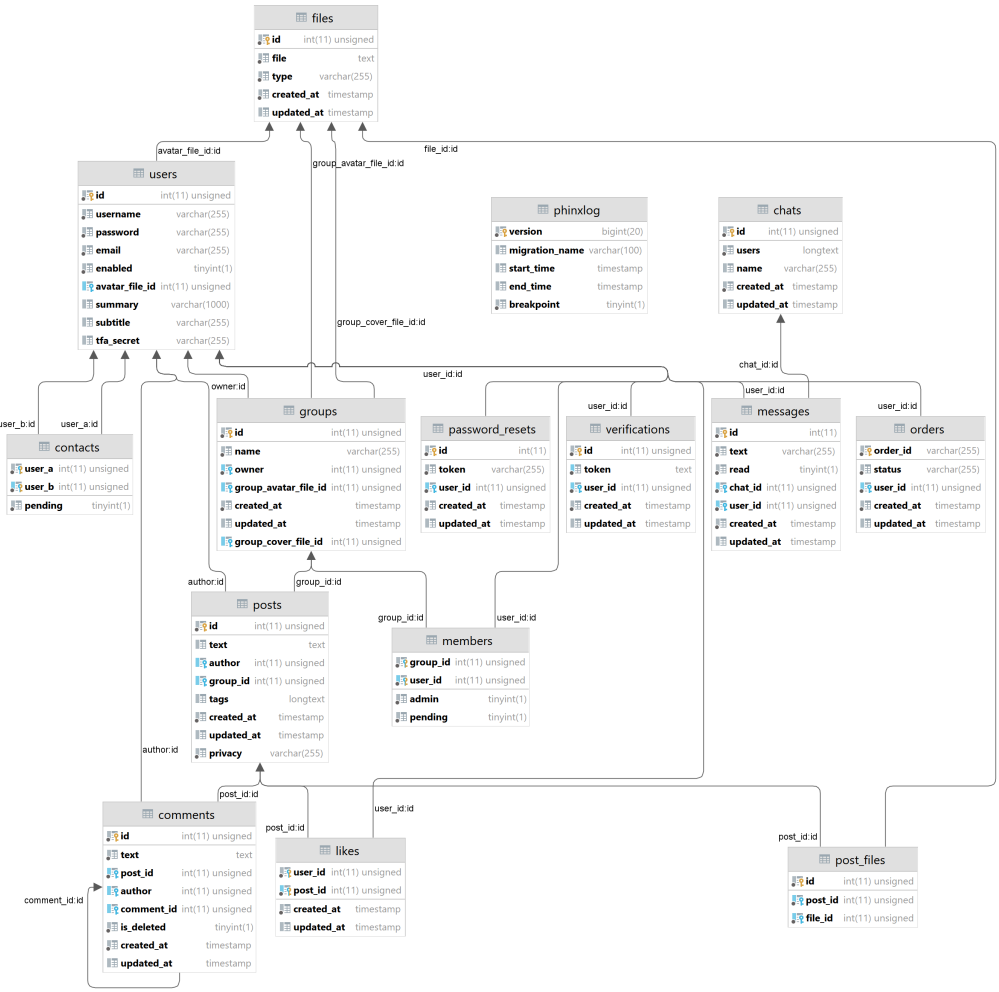
Daljnje važno razmatranje je replikacija baza podataka kroz više podatkovnih poslužitelja, što bi omogućilo rasterećenje poslužitelja i ubrzavanje zahtjeva. Dodatna prednost replikacije je i mogućnost jednostavnog držanja sigurnosnih kopija podataka u slučaju ispada sustava, kompromitiranja podataka ili oštećenja podataka.

Također, pohranjivanje datoteka na podatkovni sustav poslužitelja nije dobro dugoročno rješenje jer će dovesti do usporavanja poslužitelja. Pohrana podataka se treba premjestiti na predani servis za pohranjivanje kao što su *Google Cloud Storage* ili *Amazon S3*.

Konačno, zapisivanje aplikacijskih dnevnika moglo bi se premjestiti ili u svakom slučaju komplementirati uvođenjem servisa za zapisivanje dnevnika u online servis kao što je primjerice Rollbar. Tako bi tok događaja koji se zapisuju unutar aplikacije mogao biti dostupan preko centraliziranog servisa, pogotovo ukoliko bi više ljudi surađivalo na izradi projekta.

7.2. Podatkovni model

Društvene mreže po svojoj prirodi moraju biti interaktivne i pohranjivati raznovrsne podatke kako bi omogućile svoje funkcionalnosti.



Slika 33: ERA dijagram društvene mreže

Izvor: automatski generirani dijagram putem PHPStorm aplikacije

Kao što je vidljivo na dijagramu 33, središnji objekt je sam korisnik. Za korisnika se veže većina podataka kao što su objava, komentar, vlasništvo nad grupom, autorstvo poruka, pripadništvo grupama, odnosi s drugim korisnicima (kontakt ili ne), verifikacija, resetiranje lozinki i "lajkanje" objava. S obzirom na funkcionalnosti aplikacije, podatkovni model nije velik niti zamršen.

Korisniku se tako pripisuju informacije o njegovoj e-mail adresi, korisničkom imenu, lozinci, tituli i sažetom samoopisu. Nadalje, ukoliko je korisnik uključio opciju višefaktorske prijave, zapisuje se tajni podatak koji se koristi za verifikaciju privremenih kôdova za višefaktorsku prijavu (koje generira aplikacija neke treće strane, najčešće instalirane na korisnikov mobitel). Također, korisnici mogu učitati i ažurirati svoju korisničku sliku. Korisničke slike, ali i datoteke

objava te "profilne" slike grupa se pohranjuju u vlastitoj tablici *files*. No, kako je bilo rečeno da se datoteke pohranjuju na datotečnom sustavu (čija je lokacija poznata aplikaciji), u toj tablici se pohranjuju samo nazivi datoteka i njihov tip, dok tablice koje pamte neku datoteku samo referenciraju vanjski ključ na tablicu datoteka.

Sljedeći entitet koji je važno spomenuti je entitet objava (engl. *posts*) koji pohranjuje podatke o svakoj pojedinačnoj objavi na mreži. Objava ima svog autora, oznake (podatkovni tip JSON [ili longtext, ovisno o interpretaciji], odnosno pohranjuju se kao jednodimenzionalna polja), pripadajuću grupu unutar koje je objavljena (ukoliko uopće jest) te postavku za privatnost. Kako objave mogu sadržavati više datoteka, uvodi se slabi entitet *post_files* koji vodi računa o pripadnosti datoteka objavama te se na taj način mogu graditi galerije unutar objava.

Entiteti *password_resets* i *verifications* redom služe za pohranjivanje zahtjeva za ponovno postavljanje lozinke i za verifikaciju e-mail adrese novo registriranog korisnika te se za njihove jedinstvene identifikacije koristi nasumično generirani token koji se veže uz korisnika. Budući da se za oba zahtjeva korisniku šalje e-mail poruka sa poveznicom koja sadrži taj token, osigurava se da samo taj korisnik može imati pristup zahtjevu.

Entitet *orders* bilježi informacije o stanju korisničke pretplate za *premium* značajke aplikacije. Bilježi identifikator narudžbe kakvoga ga vraća sustav za naplatu te se bilježi stanje zahtjeva, odnosno da li je naplata tek zatražena ili je izvršena. Ukoliko je izvršena, korisnik je ostvario napredne značajke.

Svaki razgovor među korisnicima se vodi kao zaseban zapis unutar entiteta *chats*. Korisnici koji su članovi razgovora bilježe se pod podatkom *users*.

Entitet koji iznimno odudara je *phinxlog* - radi se o tablici meta-podataka koju koristi biblioteka za *migriranje* baza podataka o kojoj će biti više riječi kasnije.

7.3. Dijagram slučajeva korištenja

Dijagrami korištenja prikazuju na visokoj razini interakciju korisnika s nekim sustavom te odnose između korisnika, dijelova sustava i slučajeva uporabe tog sustava. Svaki slučaj korištenja određuje očekivano ponašanje i mogućnosti sustava, ali ne opisuju detalje implementacije tih ishoda. Tako dijagram 34 prikazuje funkcionalnosti društvene mreže i koje vrste korisnika mogu ući u interakciju s kojim dijelovima mreže.

Mogu se izdvojiti 4 tipa (potencijalnih) korisnika:

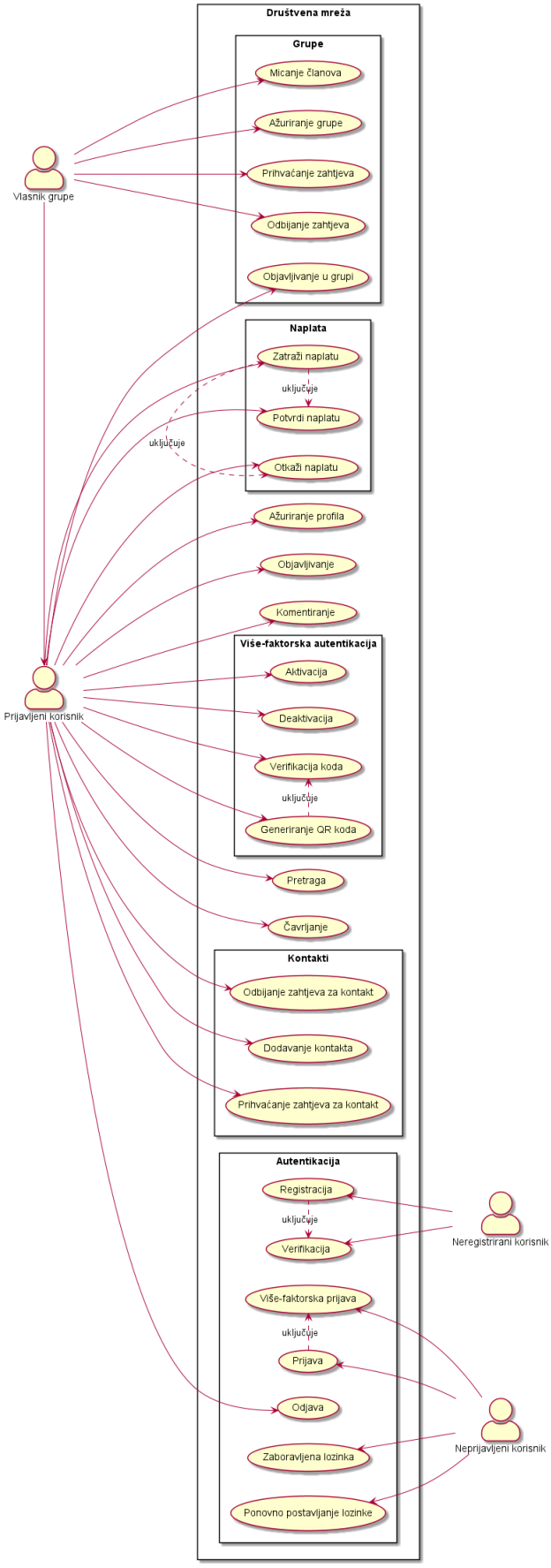
- Neregistrirani korisnik
- Neprijavljeni korisnik
- Prijavljeni korisnik
- Vlasnik grupe

Neregistrirani korisnik Neregistrirani korisnik naravno nije u mogućnosti biti u interakciji s bilo kojim dijelom društvene mreže dok se ne registrira i verificira svoju e-mail adresu.

Neprijavljeni korisnik Korisnik koji se registrirao i verificirao sada je u mogućnosti prijaviti se. Prijava može biti klasična ili, ako je korisnik podesio postavke, višefaktorska (engl. *two-factor authentication*, *multi-factor authentication*). Također, ukoliko je korisnik zaboravio svoju lozinku, u mogućnosti je zatražiti ponovno postavljanje lozinke.

Prijavljeni korisnik Prijavljeni korisnici imaju pristup cijeloj aplikaciji i njenim mogućnostima u koje spada objavljivanje sadržaja, podešavanje korisničkoga profila, pregled i pristupanje grupama, pretraga sadržaja, slanje zahtjeva za prijateljstvom (kontakt), komentiranje, označavanje da mu se objava sviđa i tako dalje.

Vlasnik grupe Vlasnik grupe je svaki prijavljeni korisnik koji je stvorio novu grupu te nad njom ima posebne administratorske ovlasti. U mogućnosti je ažurirati postavke grupe, prihvaćati i odbijati zahtjeve za članstvom te također naknadno izbacivati nepoželjne članove.



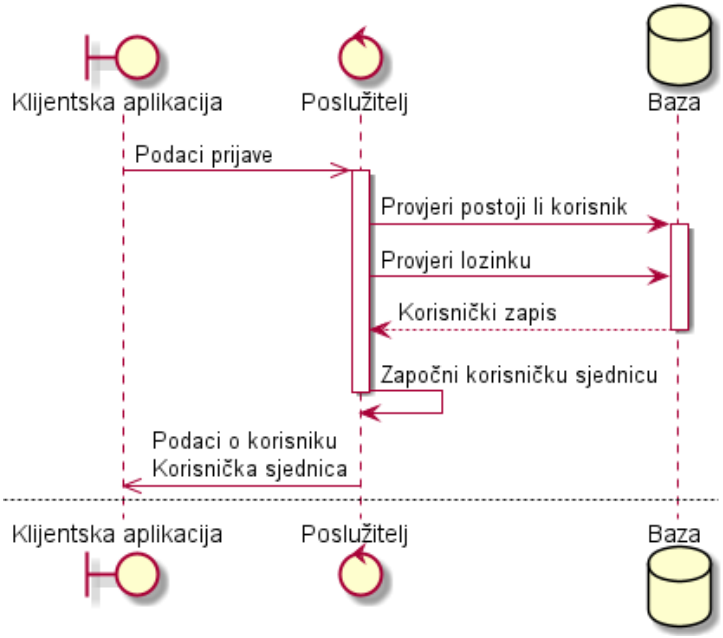
Slika 34: Dijagram korištenja društvene mreže

7.4. Dijagrami slijeda

Dok dijagrami slučajeva korištenja daju pregled nad uporabom sustava na visokoj razini, dijagrami slijeda predstavljaju detalje interakcije pojedinih slučajeva korištenja. U dijagramu slijeda prikazane su interakcije između različitih objekata (sudionika) u redoslijedu u kojem se one odigravaju.

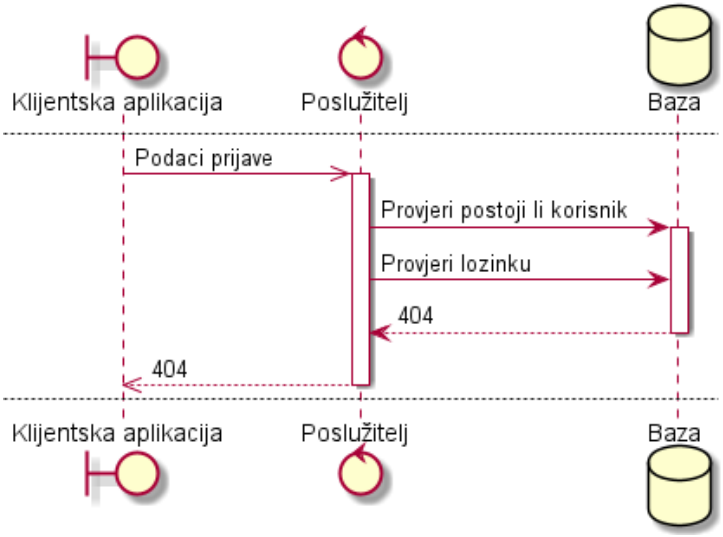
7.4.0.1. Prijava

Putem prijave korisnik započinje korisničku sjednicu preko koje obavlja aktivnosti na društvenoj mreži. Korisnik kroz klijentsku aplikaciju unosi svoje korisničke podatke koji se šalju preko REST API-ja na poslužiteljsku stranu. Tamo podaci bivaju validirani i provjerava se postoji li uopće u bazi podataka zapis o korisniku s tim podacima.



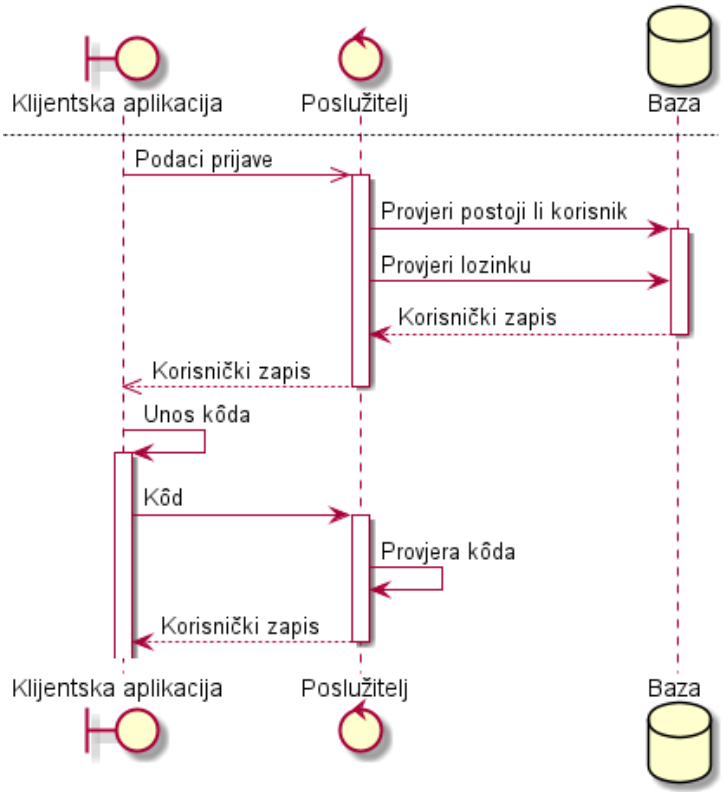
Slika 35: Dijagram slijeda uspješne prijave

U slučaju nepostojećeg korisnika REST API vratiti će informaciju o pogrešci 404 i korisnik će biti obaviješten da korisnik s unesenim podacima ne postoji.



Slika 36: Dijagram slijeda neuspješne prijave

Ukoliko je korisnik uključio opciju višefaktorske autentikacije onda nakon uspješne provjere postojanja korisničkog zapisa isti biva vraćen klijentskoj aplikaciji te se od korisnika traži da unese jednokratnu lozinku iz svoje autentikacijske aplikacije (npr. Google Authenticator). Kôd se šalje poslužitelju koji provjerava njegovu valjanost i ukoliko je ispravan, korisnika se prijavljuje, započinje se korisnička sjednica i vraća se zapis na klijentsku aplikaciju.

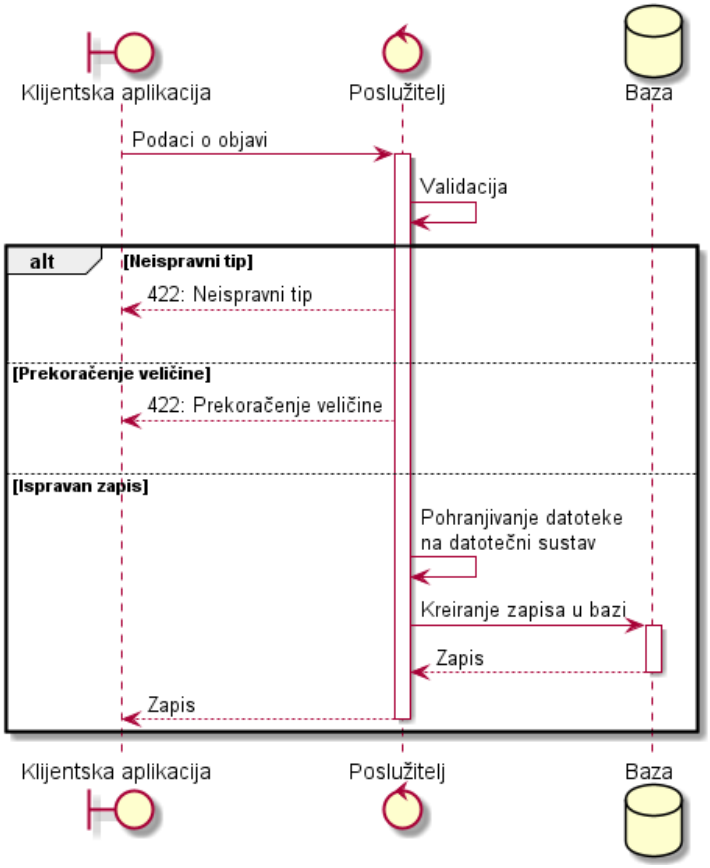


Slika 37: Dijagram slijeda višefaktorske prijave

7.4.0.2. Objavljivanje

Objavljivanjem korisnici učitavaju sadržaj na društvenu mrežu te onda drugi korisnici mogu taj sadržaj pregledavati. Podaci o objavi se šalju poslužitelju koji izvršava nekoliko validacijskih provjera. Prije svega, sadržaj mora biti podržanog tipa (slike, snimke, itd). Ukoliko nije, informacija se vraća korisniku. Isto vrijedi i za provjeru veličine datoteke koja mora biti ograničena.

Ukoliko je sve u ispravno, učitane datoteke se zapisuju na datotečni sustav i u bazi podataka se kreiraju potrebni zapisi koji sve povezuju. Na klijentsku stranu se vraća zapis o objavi i ažurira se prikaz.



Slika 38: Dijagram slijeda objavljivanja

7.4.0.3. Naplata

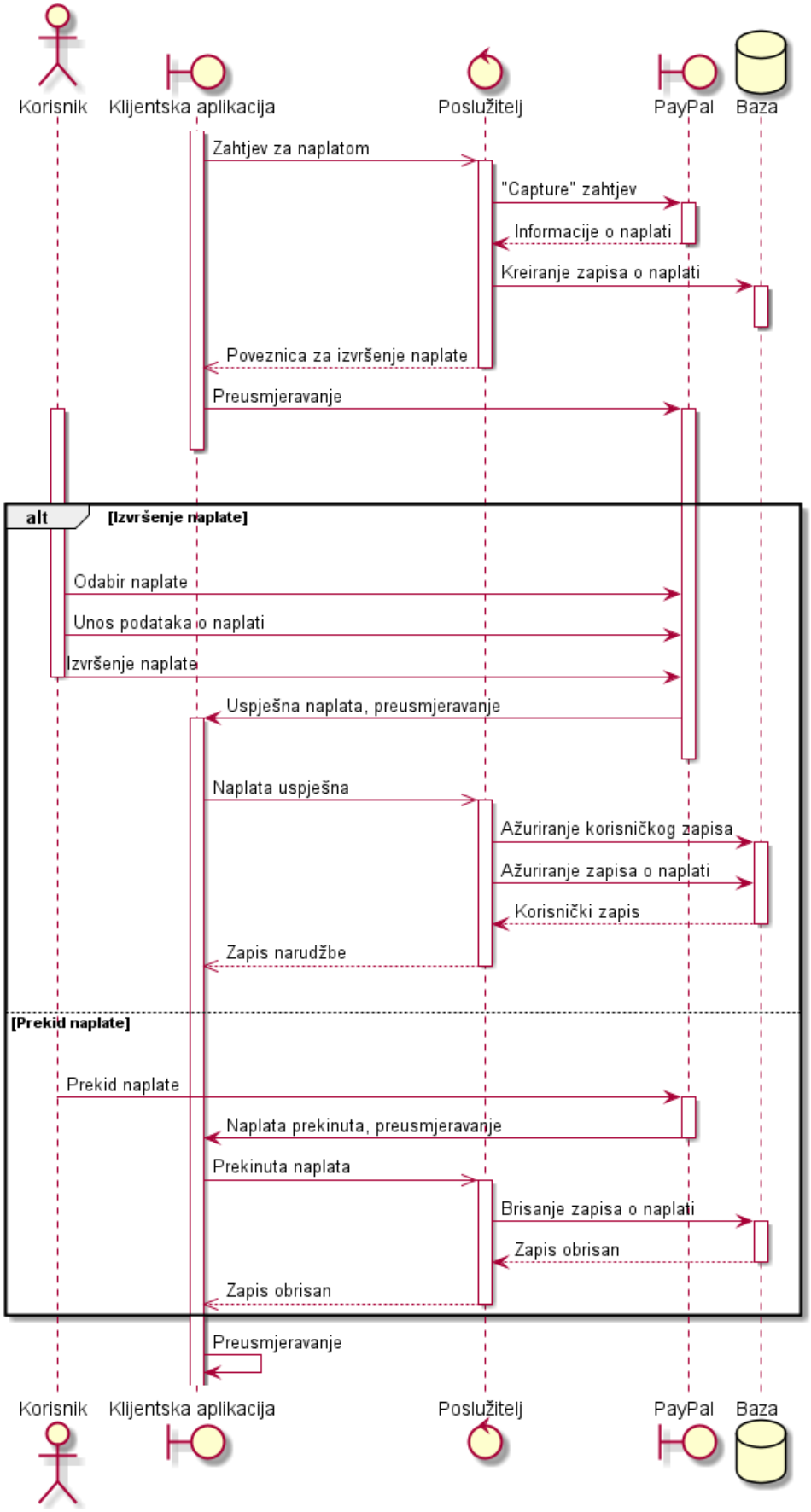
Naplata se izvršava kako bi korisnik ostvario dodatne pogodnosti unutar aplikacije. Za izvršenje naplata odgovorna je integracija sa PayPal-om. Slika 39 prikazuje slijed radnji za obavljanje naplate.

Korisnik šalje zahtjev za naplatom putem klijentske aplikacije i poslužitelj onda kreira i šalje poziv prema PayPal-u za stvaranje naplate. PayPal vraća informacije o naplati, koje uključuje poveznicu na koju treba preusmjeriti korisnika. Također se u bazi podataka stvara zapis o traženoj naplati. Korisnika se preusmjerava na PayPal-ovu stranicu za naplatu gdje onda postoje dva moguća ishoda.

Korisnik bira metodu naplate i unosi podatke za naplatu. Naplata se izvršava i ukoliko je sve prošlo bez greške, PayPal preusmjerava korisnika natrag na stranicu klijentske aplikacije zadužene za obrađivanje uspješne naplate. Ona poziva poslužitelj da ažurira korisnički zapis i zapis o naplati. Vraća se informacija o izvršenoj akciji.

Korisnik ima također opciju predomisliti se te stranica za naplatu nudi poveznicu za prekid naplate koja vodi na posebnu stranicu na klijentskoj aplikaciji koja onda informira poslužitelja o prekidu naplate. Tada se briše zapis naplati iz baze podataka.

Neovisno o ishodu naplate, izvršava se preusmjeravanje na glavnu stranicu mreže.



Slika 39: Dijagram slijeda naplate

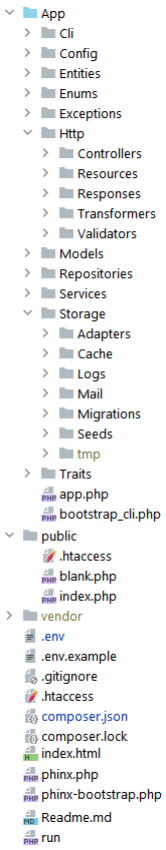
7.5. Struktura kôda i dijagrami klasa

Programski kôd može za isti projekt biti strukturiran na velik i raznolik broj načina, pogotovo ako se razvoju pristupa od "nule". Za potrebe ovog projekta korišteni su razvojni okviri PhalconPHP i Angular, koji svaki dolaze s određenim pretpostavkama što se tiče strukture kôda zbog prirode njihove implementacije (što se naziva konvencija).

7.5.1. Poslužiteljska strana

Poslužiteljska strana aplikacije servira REST API kojeg konzumira klijentska strana aplikacije. Spomenuti REST API implementiran je u jeziku PHP pomoću PhalconPHP razvojnog okvira. PhalconPHP nudi više vrsta "aplikacija" (kako ih nazivaju) koje služe različitim svrhama, odnosno unaprijed pripremaju različiti skup servisa (full-stack aplikacija, micro aplikacija, Cli aplikacija [naredbeni redak]). Budući da se servira REST API, za te potrebe se koristila Phalconova tzv. *Micro* aplikacija te ona učitava minimalnu količinu servisa i koja služi najčešće izradi API-ja, prototipova i slično (poput Laravelovog Lumen okvira) (Phalcon Tim 2021.b). Budući da je naša klijentska strana implementirana putem Angulara, Phalcon-ova Micro aplikacija predstavlja dobar odabir.

PhalconPHP ne nameće mnogo konvencija na razvojnog programera, stoga se struktura projekta može prilagođavati. Na slici 40 prikazana je struktura kôda za REST API.



Slika 40: Struktura kôda za poslužiteljsku stranu aplikacije

Slijedi opis direktorija (odnosno imeničkih prostora [engl. *namespace*]) te što predstavljaju unutar strukture kôda.

Cli Sadrži namjenske komande naredbenog retka pisane u PHP-u

Config Sadrži konfiguracijsku datoteku te priprema ostatak aplikacije učitavanjem potrebnih imeničkih prostora i postavljanjem zajedničkih servisa

Entities PhalconPHP-ovi modeli entiteta koji se nalaze u bazi podataka

Enum Enumeracije

Exceptions Namjenske klase programskih iznimaka

Http Sadrži klase i kôd vezan uz obradu HTTP zahtjeva

- **Controllers** Sadrži sve kontrolere
- **Resources** Sadrži sve resurse koje kontroleri vraćaju klijentskoj aplikaciji
- **Responses** Namjenske klase HTTP odgovora
- **Transformers** Pretvaračke klase koje koriste resursi kako bi oblikovali svoje podatke
- **Validators** Klase za validaciju pristiglih podataka

Models Podatkovni modeli koji nisu vezani uz entitete baze podataka

Repositories Sadrži repozitorijske klase koje su odgovorne za baratanje podacima

Services Svi ponovno iskoristivi servisi koji enkapsuliraju određenu radnju se nalaze u ovom direktoriju

Storage Stvari vezane uz pohranu podataka

- **Adapters** Podatkovni prilagodnici
- **Cache** Privremena pohrana, prvenstveno za Phalcon-ove metapodatke o entitetima
- **Logs** Dnevnici aplikacije i događaja, tu se zapisuju greške i iznimke
- **Mail** HTML predlošci za e-poštu
- **Migrations** Migracije za bazu podataka
- **Seeds** Inicijalizacijski podaci za punjenje baze podataka
- **tmp** Privremene datoteke, tu se zapisuju korisničke sjednice

Traits Zajednički kôd koji se dijeli među više klasa

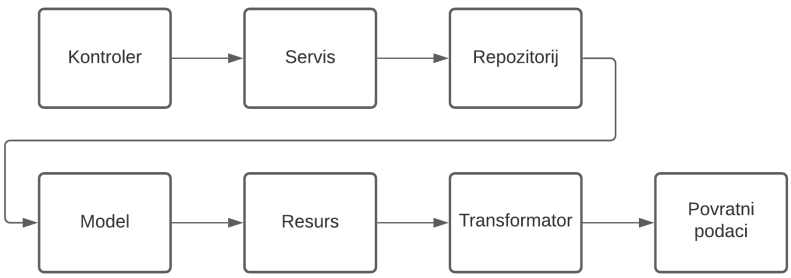
Slika 41 prikazuje životni tijek API zahtjeva kada ga poslužitelj zaprimi od strane klijentske aplikacije. Svaki kontroler je odgovoran za određeni entitet ili logički skup funkcionalnosti. Metode unutar kontrolera obrađuju pojedinačne krajnje točke API-ja. Svaka krajnja točka dozvoljava samo određenu metodu dohvata (GET, POST, itd.) te odgovara natrag klijentu s kôdom 400 ukoliko nije ispravan zahtjev.

Ukoliko je zahtjev ispravan, dohvaćaju se potrebni podaci te se instancira objekt odgovarajućeg servisa. Servisi su odgovorni za provođenje određene programske logike te iako su zamišljeni kako bi bili ponovno iskoristivi, najčešće se nalaze u 1:1 odnosu s metodama kontrolera (drugim riječima, svaka krajnja točka ima svoj predani servis). Ovakav pristup se u nekim krugovima naziva "tankim" kontrolerom (engl. *thin controller*). Servisi najčešće moraju raditi s podacima iz baze podataka te se za takve svrhe koristi uzorak dizajna *Repozitorij*.

Repozitorij predstavlja dodatni sloj apstrakcije između poslovne logike i podatkovnoga sloja aplikacije. Umjesto da servis (ili kontroler) direktno radi s podacima i modelima, takve radnje delegiraju se predanoj klasi koja je odgovorna za sve potrebne operacije nad podacima u aplikaciji. Takve operacije mogu biti višekriterijska pretraga, stvaranje, ažuriranje i brisanje podataka i tako dalje. Kako bi postigli spomenute operacije, repozitoriji se koriste entitetima koje generira PhalconPHP putem svojih alata naredbenog retka i njihovim ugrađenim graditeljem za upite (engl. *query builder*). Kao rezultat vraćaju objekte modela koji predstavljaju podatke iz baze podataka.

Konačno, spomenuti podaci se prosljeđuju tzv. "resursima" koji predstavljaju povratne podatke REST API-ja. Resursi se uglavnom temelje na entitetima i modelima kakvim se barata unutar aplikacije. Svaki resurs prihvaća objekt ili kolekciju objekata koje prosljeđuje tzv. "transformatoru" (engl. *transformer*). Transformator je odgovoran za oblikovanje odgovora, odnosno za eventualne naknadne operacije nad podacima. Oblikovani podaci se na kraju vraćaju iz kontrolera kao JSON odgovor.

Takav način organizacije kôda ima naravno svoje nedostatke, prvenstveno u količini datoteka i ponavljajućeg kôda koji se mora pisati kako bi se zadovoljila takva organizacija. No također prednost je jasno razdvajanje odgovornosti, pojednostavljenje pojedinih elemenata i izravan tijek rada.



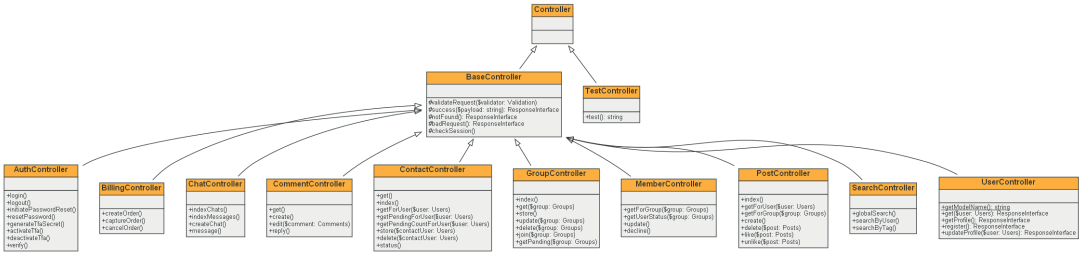
Slika 41: Životni tijek API poziva

Kako to točno izgleda u konkretno implementiranoj aplikaciji će biti prikazano u kasnijem odjeljku 7.8.

Kroz nekoliko dijagrama klasa pobliže će se pojasniti struktura kôda i na koji način se različiti dijelovi aplikacije uklapaju.

7.5.1.1. Kontroleri

Svi kontroleri aplikacije nasljeđuju roditeljsku klasu *BaseController* koja objedinjuje zajedničko ponašanje svih kontrolera. Svaki kontroler odgovara logičkoj skupini funkcionalnosti odnosno entitetu u bazi podataka.

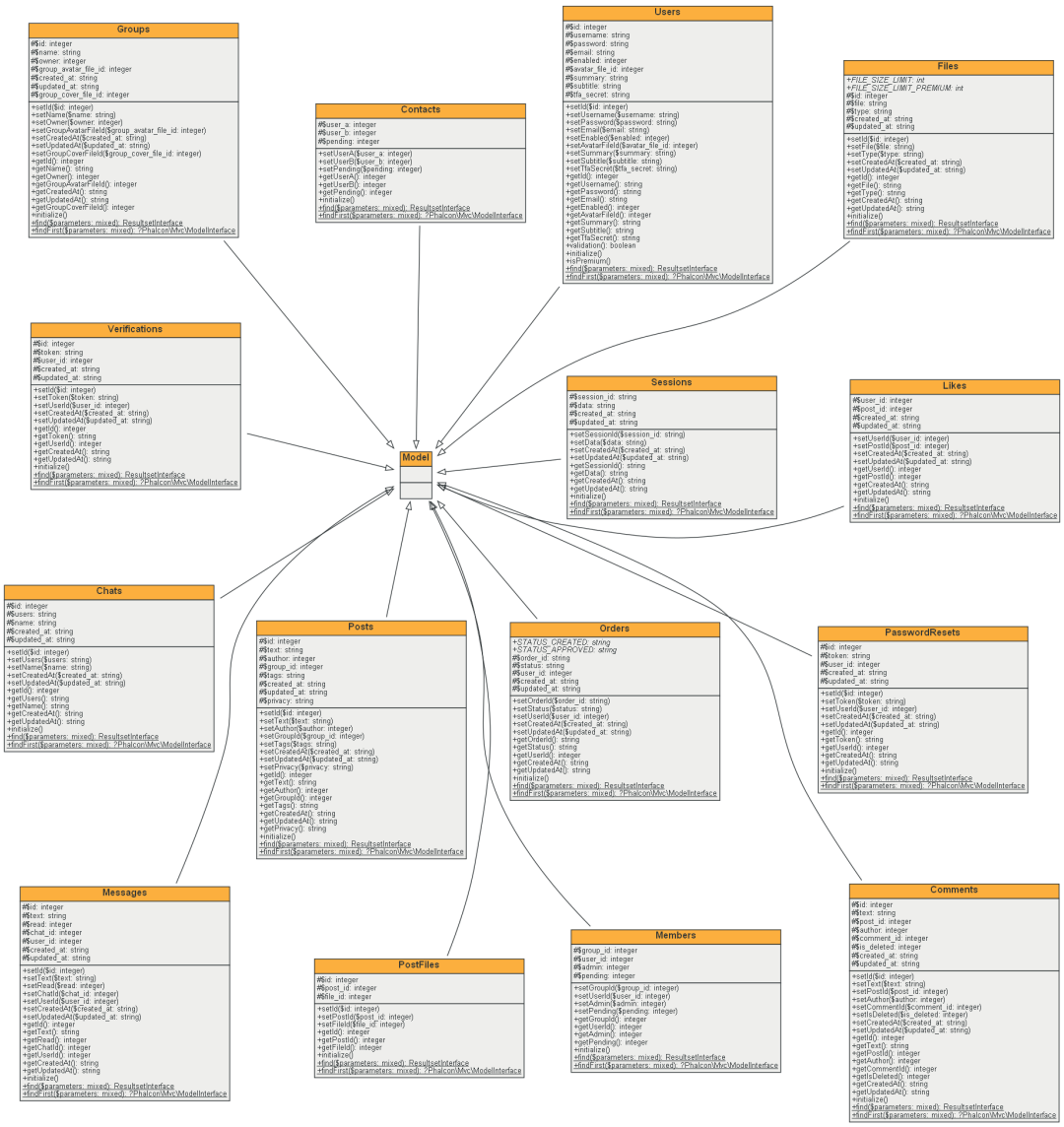


Slika 42: Dijagram klasa kontrolera

Izvor: Generirano putem Phuml skripte

7.5.1.2. Entiteti

Svi entiteti nasljeđuju Phalconovu temeljnu klasu *Model* koja sadrži svu logiku za pristup bazi podataka i predstavljanje istih podataka u obliku objekata.



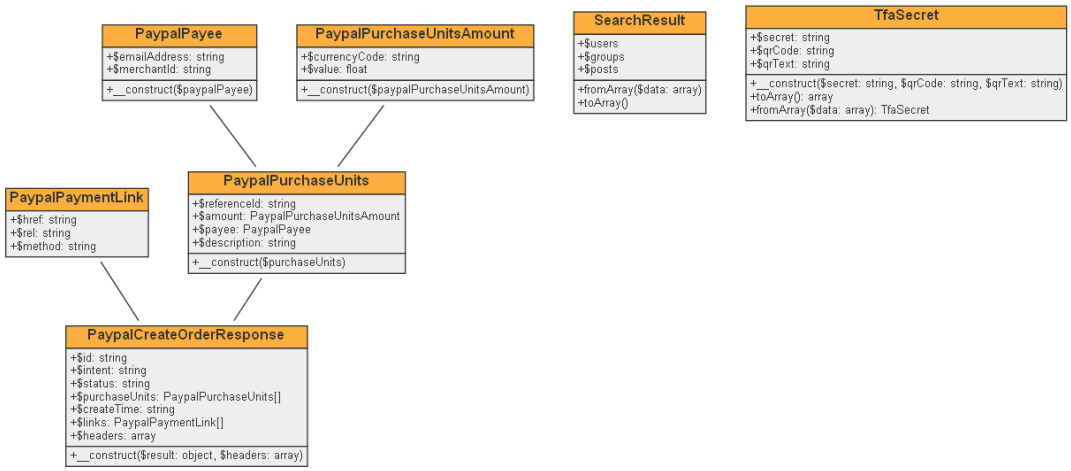
Slika 43: Dijagram klasa entiteta

Izvor: Generirano putem Phuml skripte

Treba napomenuti da nije greška što je klasa *Model* prazna iz praktičnih razloga budući da sadrži veliki broj funkcionalnosti.

7.5.1.3. Modeli

Modeli koji ne odgovaraju entitetima u bazi podataka, već se koriste za prezentiranje određenih informacija koje se obrađuju unutar aplikacije.

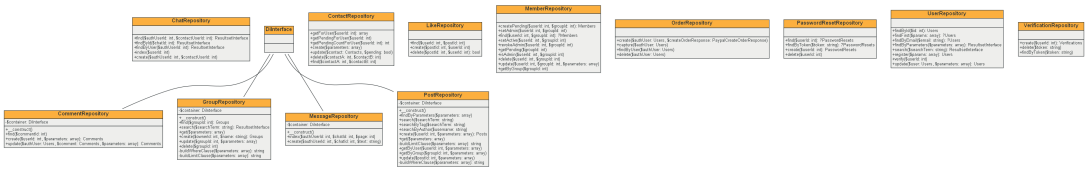


Slika 44: Dijagram klasa modela

Izvor: Generirano putem Phuml skripte

7.5.1.4. Repozitoriji

Repozitoriji su specijalizirane klase čija je jedina odgovornost baratanje s podacima iz baze podataka. U slučaju nekih, koriste se zajedničkim aplikacijskim "kontejnerom" koji sadrži dijeljene servise, primarno za potrebe vođenja dnevnika rada i zapisivanja grešaka u obradi.

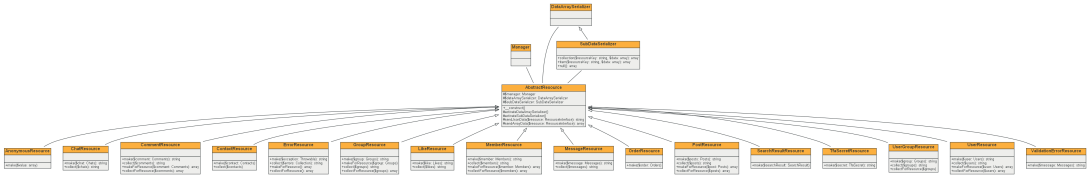


Slika 45: Dijagram repozitorijskih klasa

Izvor: Generirano putem Phuml skripte

7.5.1.5. Resursi

Resursi predstavljaju podatke kakve poslužiteljska aplikacija oblikuje i vraća klijentskoj aplikaciji. Budući da se svi resursi ponašaju slično, zajedničko ponašanje podignuto je na razinu roditeljske apstraktne klase *AbstractResource*. Ona inicijalizira i sadrži rukovoditeljsku klasu *Fractal* biblioteke koja nudi prezentacijski i transformacijski sloj za rad s kompleksnim strukturama podataka. Prednost takve biblioteke je što nudi kontrolu nad prikazom podataka i čini prikaz podataka neovisnim o strukturama podataka.



Slika 46: Dijagram klasa resursa

Izvor: Generirano putem Phuml skripte

Klasa *AbstractResource* sadrži metode za upravljanje "serijalizatorima" - klasama koje transformirane podatke upakiraju u standardiziranu povratnu strukturu. Primjerice, *DataArraySerializer* sve podatke vraća kao polje pod ključem "data". U nekim slučajevima resursi unutar sebe sadrže druge resurse. Tada se ne može koristiti *DataArraySerializer* budući da bi on podzapis upakirao u podpolje ključa "data" i završna struktura bi izgledala primjerice na sljedeći način:

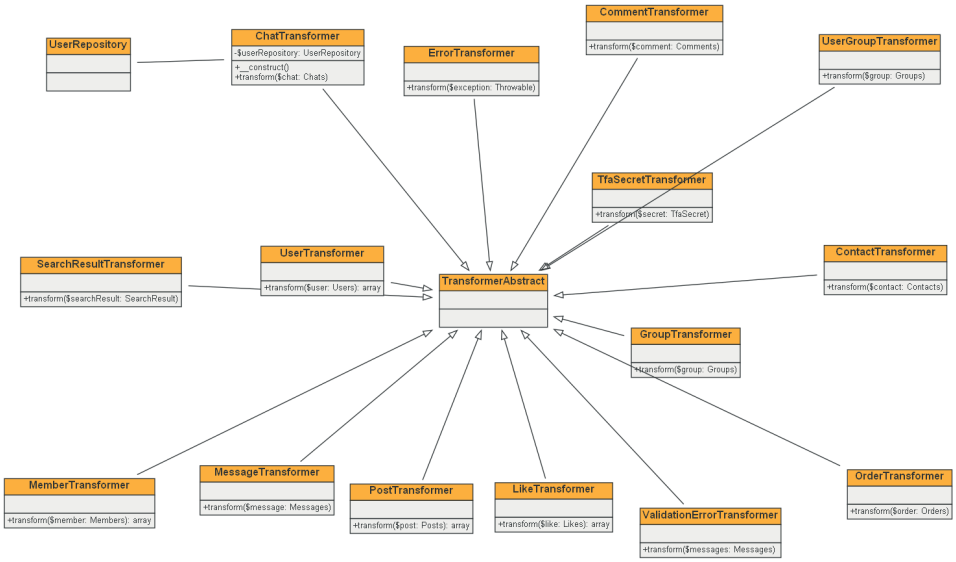
```
{
    "data": [
        {
            "id": "12",
            "text": "Neki sadrzaj",
            "tags": ["slike"],
            "files": {
                "data": [
                {
                    "id": "24",
                    "file": "MIMIKA-FRONT.jpg",
                    "type": "image/jpeg",
                    "created_at": "2021-11-23 13:54:02",
                    "updated_at": null
                }
                ]
            }
        }
    ]
}
```

Izvorni kôd 7.1: Primjer podataka kakve poslužitelj vraća klijentu

Podaci o datotekama nalaze se pod ključem data, što je svakako neintuitivno i otežava jasan rad s podacima. Stoga se uvodi *SubDataSerializer* koji vraća same podatke, bez roditeljskog "data" ključa.

7.5.1.6. Transformatori

Nadovezujući se na resurse, svaki od njih se koristi predanom klasom za transformiranje podataka koja uopće određuje internu strukturu podataka u povratnom odgovoru (sve ispod "data" ključa).

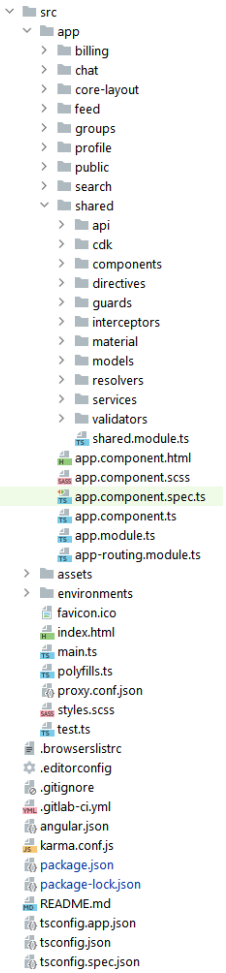


Slika 47: Dijagram klasa transformera

Izvor: Generirano putem Phuml skripte

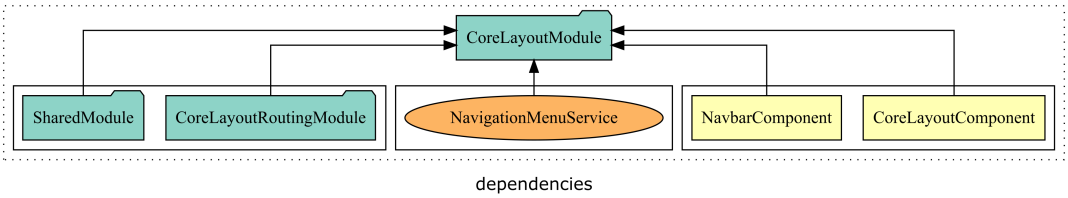
7.5.2. Klijentska strana

Klijentska aplikacija implementirana je u razvojnom okviru Angular. Kao što je bilo prikazano u odijeljku 3.3, aplikacija se sastoji od 3 glavne komponente - moduli, komponente i servisi. Struktura kôda klijentske aplikacije organizirana je u skup modula koji odgovaraju funkcionalnostima ili skupinama funkcionalnosti kakve su implementirane u aplikaciji (korisnički profil, grupe, chat, itd...). Svaki modul unutar sebe sadrži komponente putem kojih gradi različite poglede i podelemente pogleda. Od modula (direktorija) prikazanih u slici 48, valja bolje pojasniti module: *core-layout*, *public* i *shared*.



Slika 48: Struktura kôda klijentske aplikacije

Core-layout Kao što naziv modula implicira, radi se o središnjem rasporedu prikaza svih drugih elemenata. Modul sadrži podmodul za definiranje sve navigacije u aplikaciji, dvije komponente te predani servis za dinamičku izgradnju navigacijskog menija.



Slika 49: Struktura i zavisnosti Core-layout modula

Izvor: Generirano putem Compodoc skripte

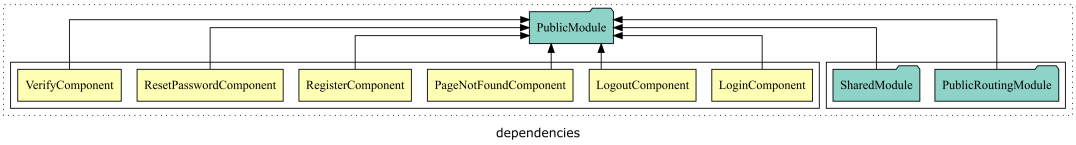
Navbar komponenta predstavlja navigacijski prozor aplikacije sa zaglavljem i navigacijskim menijem s boka.

```
1    <app-navbar>
2        <div class="core-container">
3            <!--  Content  -->
4            <router-outlet></router-outlet>
5        </div>
6    </app-navbar>
```

Izvorni kôd 7.2: core-layout.component.html

Modul *core-layout* također je odgovoran za rukovođenje svom navigacijom unutar aplikacije te se sadržaj uhvaćene rute dinamički ubacuje u *<router-outlet>* element.

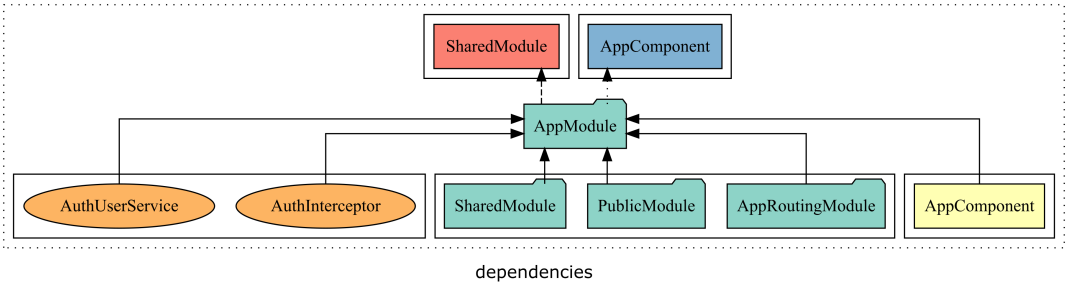
Public Modul public sadrži sve javno dostupne stranice aplikacije koje ne zahtijevaju prijavljenog korisnika. To uključuje prijavu, registraciju, verifikaciju korisnika itd.



Slika 50: Struktura i zavisnosti Public modula

Izvor: Generirano putem Compodoc skripte

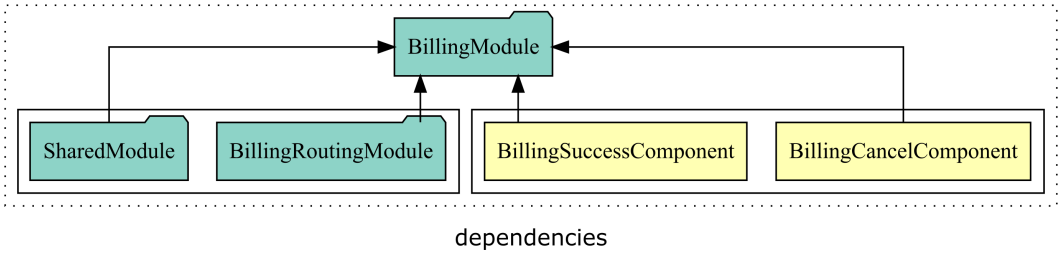
App *App* modul poseban je korijenski modul s kojim svaka Angular aplikacija započinje. Sva daljnja funkcionalnost se grana iz njega. Sadrži svoju komponentu *AppComponent* i modul za navigaciju *AppRoutingModule* koji odrađuju osnovnu navigaciju. Također na razini cjelokupne aplikacije kao *singleton* servise učitava *AuthUserService* i *AuthInterceptor*, koji redom služe za učitavanje podataka ulogiranog korisnika i za letimično podešavanje postavki HTTP poziva.



Slika 51: Struktura i zavisnosti App modula

Izvor: Generirano putem Compodoc skripte

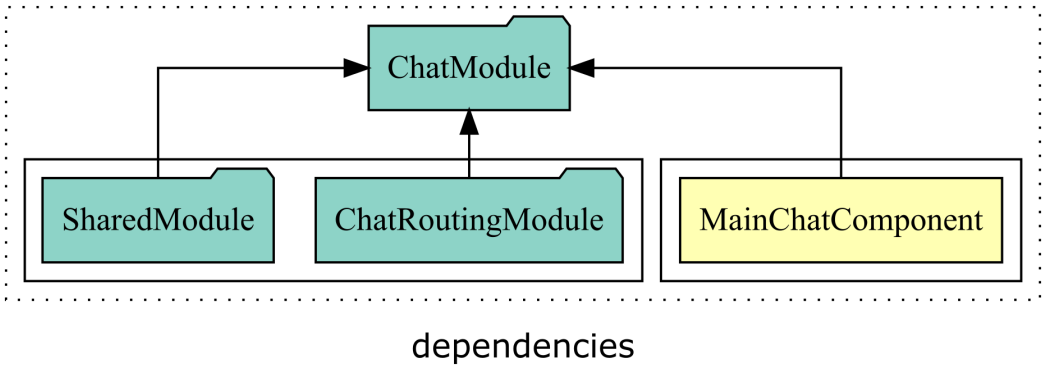
Billing Modul zadužen za dio aplikacije vezane uz naplatu. Sadrži zapravo samo dvije komponente odgovorne za obradu uspješne ili otkazane naplate.



Slika 52: Struktura i zavisnosti Billing modula

Izvor: Generirano putem Compodoc skripte

Chat Modul koji sadrži komponente vezane za čavrljanje unutar aplikacije.

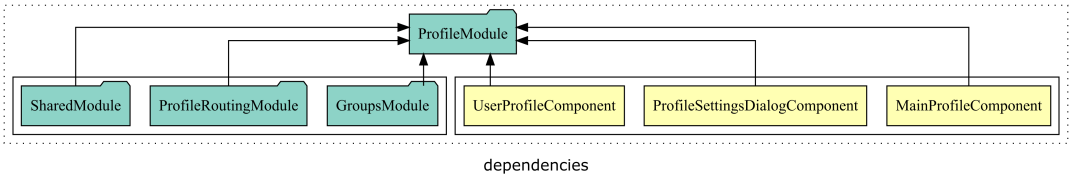


Slika 53: Struktura i zavisnosti Chat modula

Izvor: Generirano putem Compodoc skripte

Profile Odgovoran za sav prikaz korisničkih profila kao i ažuriranje profila. *MainProfileComponent* predstavlja prijavljenog korisnika, dok *UserProfileComponent* služi prikazivanju tuđih

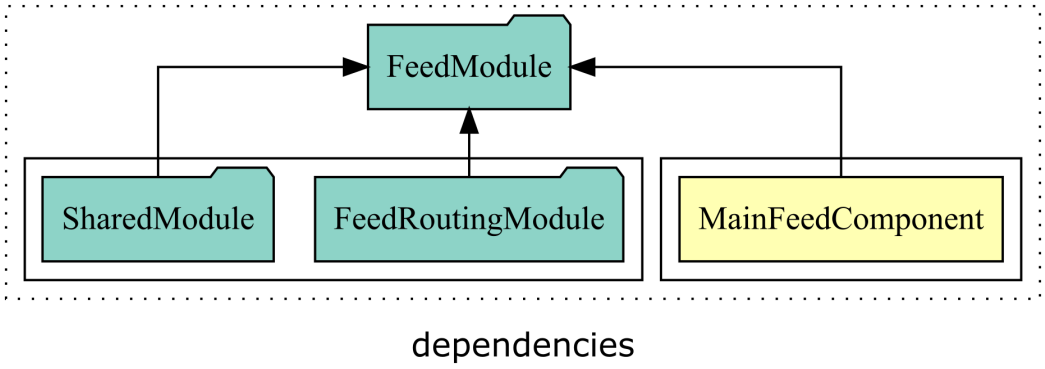
profila. *ProfileSettingsDialogComponent* je iskočni prozor (modal) za ažuriranje korisničkih postavki.



Slika 54: Struktura i zavisnosti Profile modula

Izvor: Generirano putem Compodoc skripte

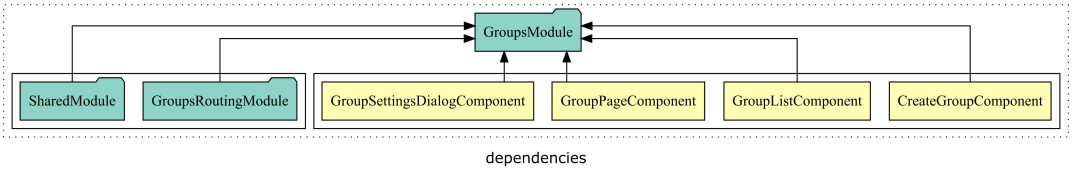
Feed *Feed* odnosno tok sadržaja je dio aplikacije koji prikazuje sadržaj koji korisnici objavljuju.



Slika 55: Struktura i zavisnosti Feed modula

Izvor: Generirano putem Compodoc skripte

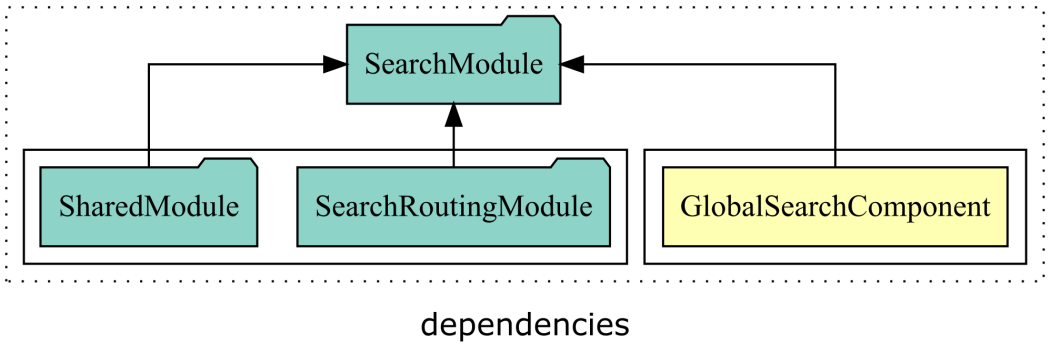
Groups Modul odgovoran za prikazivanje popisa grupa, individualnih stranica grupa i njihovo stvaranje i rukovođenje.



Slika 56: Struktura i zavisnosti Groups modula

Izvor: Generirano putem Compodoc skripte

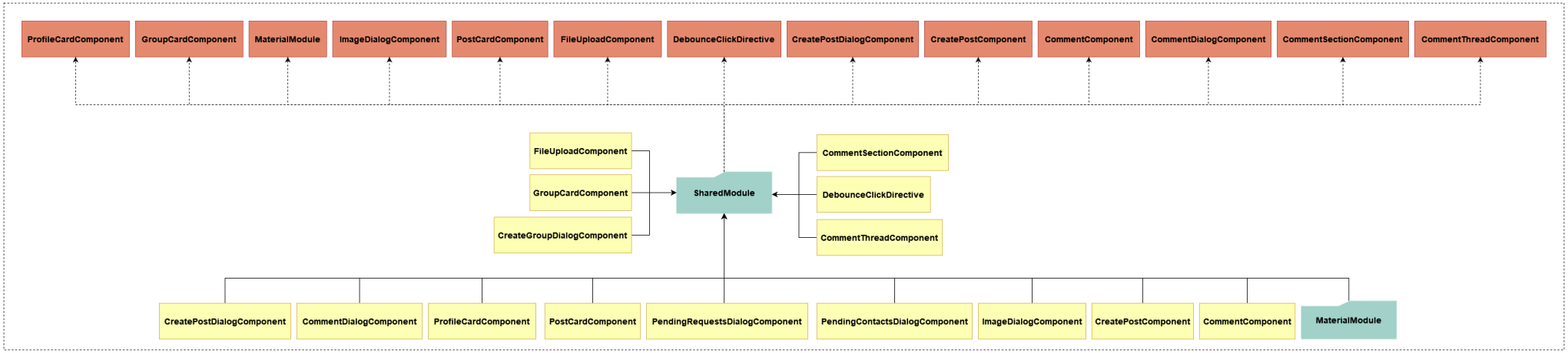
Search Modul za pretraživanje sadržaja. Sadrži jednu komponentu koja je odgovorna za dotičnu funkcionalnost.



Slika 57: Struktura i zavisnosti Search modula

Izvor: Generirano putem Compodoc skripte

Shared Modul koji sadrži sav kôd, komponente i servise koji su dijeljeni među više drugih modula. Stoga je i najveći modul.



Slika 58: Struktura i zavisnosti Shared modula

Izvor: Ručna izrada

Sadrži servise za rad sa REST API-jem, dijaloške okvire koji se pozivaju, podatkovne modele (povratni resursi, podaci koji se šalju preko API poziva, itd), druge komponente koje se koriste diljem aplikacije, druge servise, direktive, validatore za obrasce i servise koji u letu učitavaju neke potrebne podatke ovisno o navigaciji (engl. *resolver*).

7.5.3. Navigacija

SPA aplikacije posjeduju navigacijski modul preko kojeg se definiraju dijelovi aplikacije koji se učitavaju za određene URL adrese.

Slika 59: Definirane rute klijentske aplikacije

Izvor: Generirano putem Compodoc skripte

Svaka ruta odgovara određenoj komponenti ili modulu. Svaki modul za sebe definira svoje rute i međusobnim referenciranjem drugih modula u jednome, gradi se hijerarhija ruta. Na isječku 7.3 prikazana je navigacija osnovnog *AppModule* modula.

```
const routes: Routes = [
    {
        path: '',
        loadChildren: () => import('./core-layout/core-layout.module').then(m => m.
        ↪  CoreLayoutModule),
    },
    {
        path: '**',
        component: PageNotFoundComponent,
        data: {title: '404', breadcrumb: '404'},

    },
];
```

Izvorni kôd 7.3: Navigacija AppModule-a

Odmah se učitavaju rute modula *CoreLayout*. Ukoliko se učita ruta koja ne postoji, ona se lovi posebnim uzorkom '**' koji predstavlja svaku drugu rutu koja ne odgovara onima definiranima unutar aplikacije.

```
const routes: Routes = [
    {
        path: '',
        component: CoreLayoutComponent,
        runGuardsAndResolvers: "always",
        resolve: {user: ProfileResolver},
        children: [
            {
                path: '',
                loadChildren: () => import('../public/public.module').then(m => m.
                ↪  PublicModule),
            },
            {
                path: 'feed',
                loadChildren: () => import('../feed/feed.module').then(m => m.
                ↪  FeedModule),
            },
            {
                path: 'groups',
                loadChildren: () => import('../groups/groups.module').then(m => m.
                ↪  GroupsModule),
            },
            {
                path: 'profile',
                loadChildren: () => import('../profile/profile.module').then(m => m.
                ↪  ProfileModule),
            },
            {
                path: 'search',
                loadChildren: () => import('../search/search.module').then(m => m.
                ↪  SearchModule),
            },
```

```
28                {
29                    path: 'chat',
30                    loadChildren: () => import('../chat/chat.module').then(m => m.
                      ↪  ChatModule),
31                },
32                {
33                    path: 'billing',
34                    loadChildren: () => import('../billing/billing.module').then(m => m.
                      ↪  BillingModule)
35                }
36            ]
37        },
38    ];
```

Izvorni kôd 7.4: Navigacija CoreLayoutModule-a

Navigacija definirana pod modulom *CoreLayoutModule* glavna je točka račvanja koja učitava komponente svih drugih modula. Prema zadanim postavkama, modul javnih stranica (*PublicModule*) se učitava. U protivnome, ovisno o navigiranoj ruti, otvaraju se različiti drugi podmoduli koji odgovaraju različitim skupinama funkcionalnosti aplikacije. Također, za svaku uhvaćenu rutu učitat će se korisnički profil kako bi se potvrdilo da je korisnička sjednica i dalje aktivna.

7.6. Zavisnosti

Većina problema u razvoju programa su standardni i susretani su stotine puta. Stoga nema smisla iznova osmišljati kotač i raditi vlastite implementacije rješenja poznatih problema. Većina aplikacija se oslanja na postojeće pakete i biblioteke koji rješavaju određene stvari. Takvi paketi i biblioteke se nazivaju *zavisnostima* (engl. *dependencies*) aplikacije. Jednako tako i ovaj projekt ovisi o određenim zavisnostima, odnosno koristi pakete. U najmanju ruku, to su razvojni okviri koji su korišteni za njegovu realizaciju, ali i dodatni pomoćni paketi. Slijedi popis zavisnosti za klijentsku i poslužiteljsku aplikaciju te čemu oni služe.

7.6.1. Klijentska aplikacija

Zavisnostima Angular aplikacija upravlja se putem poznatoga *node package managera* (Npm) koji je odgovoran za rad s programskim zavisnostima javascript (nodejs) aplikacija. Informacije o tome zapisuje unutar *package.json* datoteke. Sljedeći paketi su navedeni kao zavisnosti.

- @angular-material-components/file-input - Specijalizirana komponenta za unos datoteka unutar obrazaca
- @angular/animations - Angularov paket za CSS animacije
- @angular/cdk - Angularov *component dev kit*, sadrži razna korisna ponašanja koja se mogu ugraditi u komponente
- @angular/common - Zajedničke komponente za Angularov okvir
- @angular/compiler - Angularov kompilator
- @angular/core - Temeljna Angular biblioteka
- @angular/flex-layout - Direktive za jednostavno upravljanje flex CSS rasporedom
- @angular/forms - Angularove direktive i servisi za rad s obrascima
- @angular/material - Komponente, servisi i direktive koje implementiraju materijalni dizajn
- @angular/platform-browser - Biblioteka koja omogućava rad Angulara unutar web preglednika
- @angular/platform-browser-dynamic - Biblioteka koja omogućava rad Angulara unutar web preglednika koristeći JIT kompilaciju
- @angular/router - Angularova biblioteka za navigaciju
- rxjs - Biblioteka za asinkronu obradu događaja
- tslib - Dodatne pomoćne funkcije za TypeScript
- zone.js - Biblioteka koja implementira odvojene kontekstualne zone za provođenje asinkronih radnji

7.6.2. Poslužiteljska aplikacija

PHP aplikacije također imaju svoj program za upravljanje zavisnostima zvan *Composer*. Isto kao i NPM, koristi *package.json* datoteku kako bi pamtio i vodio računa o zavisnostima aplikacije.

- monolog/monolog - Biblioteka za zapisivanje i rad s dnevnicima
- ext-phalcon - PhalconPHP okvir pakiran kao PHP proširenje
- ext-json - Podrška za JSON
- vlucas/phpdotenv - Podrška za .env datoteke koje sadrže varijable okoline
- league/fractal - Biblioteka za prezentaciju i transformaciju kompleksnih struktura podataka
- ext-gd - Biblioteka za generiranje slika
- phpmailer/phpmailer - Biblioteka za rad s e-mail serverima i slanje poruka
- robthree/twofactorauth - Podrška za višefaktorsku autentikaciju
- bacon/bacon-qr-code - Biblioteka za generiranje i prikaz QR kôdova
- paypal/paypal-checkout-sdk - PayPal-ova klijentska biblioteka

7.7. Konfiguracija

Razvoj aplikacije i njezino distribuiranje su dva odvojena koraka koja podrazumijevaju drugačije okoline. Dapače, u svrhu sigurnosti i jednostavnosti, određeni elementi aplikacije i njene arhitekture mogu se razlikovati između različitih okolina i ovisiti o njoj. Najjednostavniji primjer je baza podataka na koju se aplikacija povezuje. Za potrebe razvoja, ne smije se raditi sa produkcijskim podacima jer bi to moglo dovesti do havarije sustava. Stoga mora postojati način da se takvi dinamični, promjenjivi elementi mogu jednostavno definirati i učitavati ovisno o okolini. Upravo tomu služe takozvane *varijable okoline* (engl. environment variables).

Angular nudi ugrađenu podršku za varijable okoline kojih može biti proizvoljan broj ovisno o potrebama projekta. Varijable okoline su definirane unutar JSON datoteka pod zasebnim direktorijem *environments*.

```
export const environment = {
    production: false,
    api_url: 'http://commune.local/',
    media_url: 'http://commune.local/static/'
};
```

Izvorni kôd 7.5: Konfiguracijska datoteka za varijable okoline u Angularu

PhalconPHP sam po sebi ne nudi nikakvu podršku pored one ugrađene u samom jeziku PHP. No PHP-ova ugrađena podrška nije adekvatna za potrebe projekta pa se umjesto toga koristi namjenska biblioteka za podršku s takozvanim *dot env* datotekama (.env). ".env" posebna je tekstualna datoteka koja se nalazi unutar projekta i sadrži parove ključeva i pripadnih vrijednosti kako bi odredila te dinamičke varijable.

```
ENV=dev
DEBUG=false

DB_TYPE=Mysql
DB_HOST=localhost
DB_NAME=commune
DB_USER=root
DB_PASS=
DB_PORT=3306

SMTP_HOST=smtp.mailtrap.io
SMTP_PORT=2525
SMTP_USER=f252afc674c14e
SMTP_PASS=8c988b8df9116c
SMTP_FROM=noreply@commune.com

SUCCESS_URL=https://lisacpisac.gitlab.io/commune-frontend/billing/return
CANCEL_URL=https://lisacpisac.gitlab.io/commune-frontend/billing/cancel
```

Izvorni kôd 7.6: Konfiguracijska datoteka za varijable okoline u PhalconPHP

Konfiguracija prikazana na slici 7.6 poziva se u nekoliko mjesta unutar poslužiteljske aplikacije - prvenstveno kod učitavanja podešenja Phalcon aplikacije.

```php
use Dotenv\Dotenv;
use Phalcon\Config;

defined('BASE_PATH')
|| define(
'BASE_PATH',
getenv('BASE_PATH') ?: realpath(dirname(__FILE__).'/../..')
);
defined('APP_PATH') || define('APP_PATH', BASE_PATH.'/App');

require_once(BASE_PATH.'/vendor/autoload.php');
Dotenv::createImmutable(BASE_PATH)->load();

return new Config([
    'database' => [
        'adapter'  => $_ENV['DB_TYPE'],
        'host'     => $_ENV['DB_HOST'],
        'username' => $_ENV['DB_USER'],
        'password' => $_ENV['DB_PASS'],
        'dbname'   => $_ENV['DB_NAME'],
        'charset'  => 'utf8',
    ],

    'application' => [
        'controllersDir' => APP_PATH.'/Http/Controllers/',
        'modelsDir'      => APP_PATH.'/Entities/',
        'baseUri'        => '/',
        'debug'          => $_ENV['DEBUG'],
    ],

    'logging' => [
        'general'  => 'errors.log',
        'database' => 'query.log',
    ],
]);
```

Izvorni kôd 7.7: Konfiguracijski objekt PhalconPHP okvira

Iznimno na još nekim mjestima se varijable okoline direktno čitaju kao npr. kod podešavanja nekih globalnih aplikacijskih servisa.

```php
// Log queries
if ($_ENV['DEBUG'] === 'true') {
    /** @var \Phalcon\Events\Manager $eventManager */
    $eventManager = $container->get('events-manager');
    $eventManager->attach(
        'db:beforeQuery',
        function (Event $event, AbstractAdapter $connection) use ($container) {
            /** @var Logger $logger */
            $logger = $container->getShared('logger');
            $logger->withName('db-logger')->debug(
```

```
11                  $connection->getSQLStatement(),
12                  ['variables' => json_encode($connection->getSqlVariables())]
13              );
14          }
15      );
16      $connection->setEventsManager($eventManager);
17  }
```

Izvorni kôd 7.8: Učitavanje varijable okoline prilikom pripreme servisa

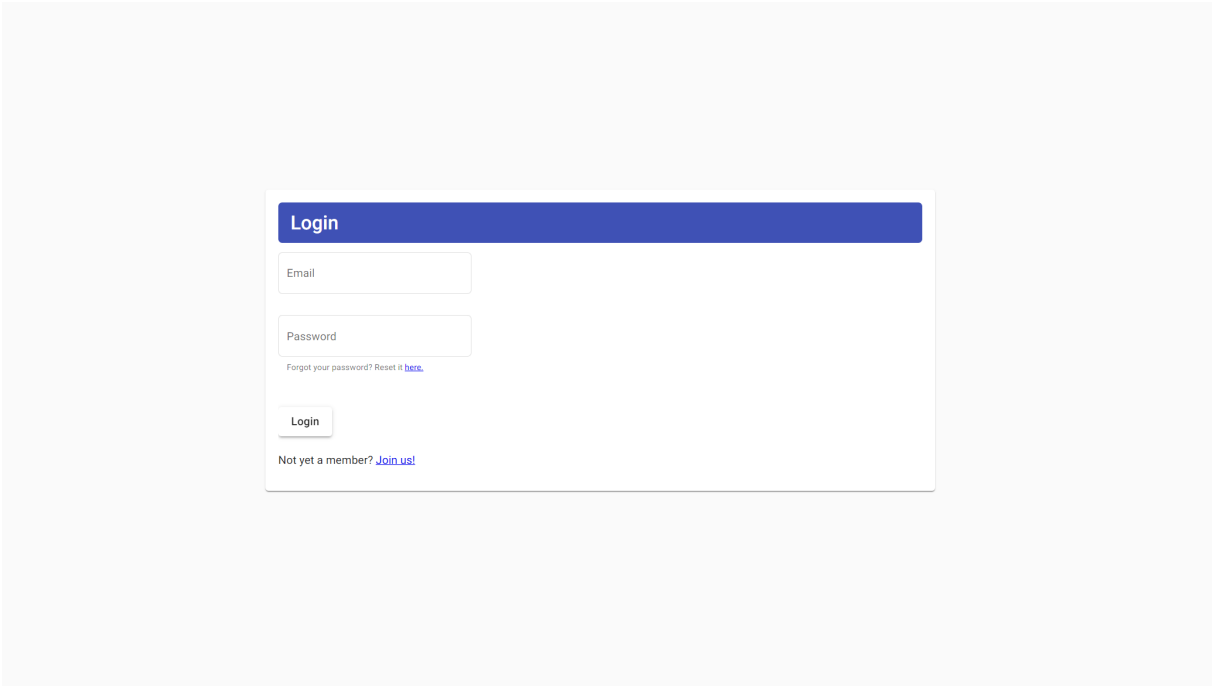
Ovisno o okolini poželjno je imati zastavicu koja upravlja tzv. *debug* načinom rada gdje se bilježe različite dodatne informacije koje su korisne prilikom razvoja. Naravno, takav način rada ne smije biti aktivan u produkciji jer može predstavljati sigurnosni rizik. Jedna takva korisna informacija su SQL upiti koje se upućuju bazi podataka. Servis za rad s dnevnikom će, ako je *debug* zastavica postavljena na istinitu vrijednost, zapisivati u zasebnu datoteku sve upite koje aplikacija radi.

7.8. Detalji implementacije

U ovom odjeljku slijede detalji implementacije nekih funkcionalnosti kako bi se konkretno prikazalo na koji način funkcioniraju pojedini elementi sustava i kakva im je međusobna interakcija.

7.8.1. Prijava

Prijava omogućuje korisniku pristup društvenoj mreži. Kako bi se to postiglo korisniku je prikazan obrazac za unos korisničkih podataka i gumb klikom kojeg se ti podaci šalju poslužitelju na obradu.



Slika 60: Prikaz obrasca prijave

Prikaz je ostvaren sljedećim HTML kôdom prikazanim u isječku 7.9:

```
<div class="container">
    <mat-card>
        <mat-card-title class="card-title">Login</mat-card-title>

        <mat-card-content>
            <form [formGroup]="form" fxLayout="column" fxLayoutGap="1em">
                <mat-form-field *ngIf="!tfaEnabled" appearance="outline">
                    <mat-label>Email</mat-label>
                    <input type="text" matInput [formControlName]="'email'">
                    <mat-error *ngIf="form.hasError('email')"
                    ↪  >{{form.getError('email')}}</mat-error>
                </mat-form-field>
                <mat-form-field *ngIf="!tfaEnabled" appearance="outline">
                    <mat-label>Password</mat-label>
```

```
                    <input type="password" matInput [formControlName]="'password'">
                    <mat-hint>Forgot your password? Reset it <a routerLink=
                    ↪  "/pw-reset">here.</a></mat-hint>
                </mat-form-field>
                <mat-form-field *ngIf="tfaEnabled" appearance="outline">
                    <mat-label>Authenticator code</mat-label>
                    <input type="text" matInput [formControlName]="'tfa_code'">
                    <mat-hint>Enter your Two-Factor authentication code</mat-hint>
                </mat-form-field>
                <mat-error *ngIf="form.hasError('tfa_code')"
                ↪  >{{form.getError('tfa_code')}}</mat-error>
                <div *ngIf="!tfaEnabled" fxLayout="column" fxLayoutGap="1em">
                    <mat-error *ngIf="form.hasError('not_found')"
                    ↪  >{{form.getError('not_found')}}</mat-error>
                    <button mat-raised-button type="button" (click)="onSubmit()"
                    ↪  >Login</button>
                    <p>Not yet a member? <a routerLink="/register">Join us!</a></p>
                </div>
                <div *ngIf="tfaEnabled">
                    <button mat-raised-button type="button" (click)="onSubmitTfa()"
                    ↪  >Confirm</button>
                </div>
            </form>
        </mat-card-content>
    </mat-card>
</div>
```

Izvorni kôd 7.9: HTML kôd stranice za prijavu

Kartica je sadržana unutar div-a koji poravnava sadržaj na sredinu prikaza. Kartica ima svoj naslov i sadržaj unutar kojeg su obrazac za unos korisničkih podataka kao i poveznice na registraciju i zaboravljenu lozinku.

Uz standardne HTML atribute i oznake koriste se i Angularove posebne direktive i naredbe. To je vidljivo u obrascu koji se veže uz konkretnu instancu modela obrasca *form*.

```
<form [formGroup]="form" fxLayout="column" fxLayoutGap="1em">
```

Izvorni kôd 7.10: Podatkovno vezanje obrasca

Pojedinačna polja unosa su vezana uz svoje kontrole na temelju naziva.

```
<input type="text" matInput [formControlName]="'email'">
```

Izvorni kôd 7.11: Podatkovno vezanje polja obrasca

U samom TypeScript-u deklarira se model obrasca i unutar inicijalizacije komponente definiraju se njezini članovi.

```
form!: FormGroup;

```

```
ngOnInit(): void {
    this.form = this.formBuilder.group({
        email: ['', [Validators.required, Validators.email]],
        password: ['', [Validators.required]],
        tfa_code: [''],
    });
}
```

Izvorni kôd 7.12: Priprema modela obrasca prijave

Uz zadane vrijednosti, dodjeljuju im se i tzv. *validatori* koji su odgovorni za ispravnost unosa te ažuriraju određene zastavice u modelu obrasca (*valid*).

Pritiskom na gumb za prijavu poziva se metoda *onSubmit* koja je vezana uz "click" događaj odnosno uz događaj pritiska mišem.

```
<button mat-raised-button type="button" (click)="onSubmit()">Login</button>
```

Izvorni kôd 7.13: Osluškivanje događaja klika gumba

```
onSubmit() {
    const payload: LoginRequestPayload = {
        email: this.form.get('email')?.value,
        password: this.form.get('password')?.value,
    };

    this.subscription = this.loginApi.login(payload).subscribe(
        (response: User) => {
            if (!response.tfa_enabled) {
                this.authUser.user = response;
                this.router.navigate(['/profile']);
            } else {
                this.tfaEnabled = true;
            }
        },
        (error: HttpErrorResponse) => {
            if (error.status === 404) {
                this.form.setErrors({not_found: 'No existing user.'});
            }
        },
    );
}
```

Izvorni kôd 7.14: Obrada prijave

U metodi *onSubmit* pripremaju se podaci koji se šalju preko REST API poziva poslužitelju. *this.subscription* atribut je koji pamti tzv. "pretplatu" odnosno "subscription" na asinkrono dostupni resurs. U pozadini "pretplate" nalazi se tzv. "*Observable*" koji predstavlja objekt na čije promjene se može "pretplatiti".

Ukoliko je došlo do greške tipa 404, korisniku se javlja da podaci nisu valjani. Ukoliko je sve u redu, preusmjerava ga na početnu stranicu.

Sam poziv prema poslužitelju ostvaren je unutar predanog servisa *AuthApiService* koji je zadužen za sve pozive prema vezanome kontroleru na poslužiteljskoj strani *AuthController*.

```
readonly basePath = `${environment.api_url}`;

login(payload: LoginRequestPayload): Observable<User> {
    const path = this.basePath + 'login';

    return this.http.post<User>(path, payload).pipe(
        map(response => (response as any).data)
    );
}
```

Izvorni kôd 7.15: Metoda poziva REST API-ja za prijavu

Kada se poziv usmjeri poslužitelju, na temelju putanje učitava se potreban kontroler i metoda unutar njega. U ovom slučaju, to će biti *AuthController* i metoda *login*.

```
public function login()
{
    if ($this->request->isPost()) {
        $service = new LoginService();

        $service->handle($this->request->getJsonRawBody(true));
        $this->session->set('user', $service->getUser());

        if ( ! empty($service->getUser())) {
            return $this->success((new UserResource())->make($service->getUser()));
        }
        return $this->notFound();
    }
    return $this->badRequest();
}
```

Izvorni kôd 7.16: Kontrolerska metoda za obradu prijave

Prvo se provjerava da li je metoda HTTP poziva ispravna. Ukoliko nije, uvjet se ne zadovoljava i nastavlja se na vraćanje zadanog odgovora 400 - neispravan zahtjev. Ukoliko je ispravna metoda dohvata, inicijalizira se servis za obradu prijave kojemu se prosljeđuju parametri poziva. Iz servisa se onda dohvaća korisnički zapis koji se bilježi u sjednicu te se vraća poruka uspjeha koja sadrži korisnički zapis.

```
class LoginService
{
    private UserRepository $userRepository;
    private ?Users $user = null;

    public function __construct()
    {
        $this->userRepository = new UserRepository();
    }
```

```
10
11      /**
12      * @throws ValidationException
13      */
14      public function handle(array $parameters): void
15      {
16          $this->validate($parameters);
17
```

Izvorni kôd 7.17: Servis za obradu prijave

Sama klasa servisa sadrži dva atributa - zapis o korisniku koji može biti prazan (null) i referencu na korisnički repozitorij koji se inicijalizira u konstruktoru.

Na početku obrade servis provjerava ispravnost podataka. Inicijalizira se objekt klase Validation i zadaju mu se pravila. Nakon toga se poziva validacijska metoda koja ta pravila primjenjuje nad poljem koje sadrži proslijeđene parametre.

```
1  public function validate(array $parameters)
2  {
3      $validator = new Validation();
4
5      $validator->add(
6          "email",
7          new Email(["message" => "The e-mail is not valid"])
8      );
9
10     $messages = $validator->validate($parameters);
11
12     if ( ! empty($messages->count())) {
13         throw new ValidationException($messages);
14     }
15 }
```

Izvorni kôd 7.18: Validacija korisničkih podataka

Ukoliko validacija nije ispravna, okida se iznimka validacije sa podacima o samoj grešci. U protivnome se tok programa nastavlja. Dohvaća se korisnički zapis na temelju e-mail adrese. Ukoliko ne postoji, preskače se ostatak kôda što tok programa vodi natrag u kontroler gdje se vraća informacija o grešci 404 - korisnički zapis ne postoji. U protivnome, inicijalizira se objekt Phalconove klase za sigurnost i provjerava da li proslijeđena lozinka, kada ju se kriptografski sažima, bude jednaka onoj zapisanoj u bazi podataka. Ako jest, znači da korisnik postoji. Jedino što se još provjerava jest da li je proslijeđen kôd za višefaktorsku autentikaciju koji se onda provjerava. Ako taj kôd nije ispravan, okida se validacijska iznimka. Također, ukoliko korisnik nije obavio verifikaciju, odnosno nije aktiviran, također se vraća validacijska greška.

Na kraju, sprema se u svoj atribut zapis o korisniku kojeg kontroler onda čita i vraća.

```
1  $user = $this->userRepository->findByEmail($parameters['email']);
2
```

```
if ( ! empty($user)) {
    $security = new Security();
    if ($security->checkHash($parameters['password'], $user->getPassword())) {
        if (array_key_exists('tfa_code', $parameters)) {
            $tfa = new TwoFactorAuth();
            $verify = $tfa->verifyCode($user->getTfaSecret(), $parameters['tfa_code'
            ↪ ]);

            if ( ! $verify) {
                $messages = new Messages();
                $messages->appendMessage(new Message('Invalid code', 'tfa_code'));
                throw new ValidationException($messages);
            }
        }
        $this->user = $user;
        if ($this->user->getEnabled() !== 1) {
            $messages = new Messages();
            $messages->appendMessage(new Message('User not enabled', 'email'));
            throw new ValidationException($messages);
        }
    }
}
```

Izvorni kôd 7.19: Provjera korisničkih podataka

Kontroler vraća odgovor 200 sa JSON sadržajem koji se gradi na temelju resursa *UserResource*.

```
return $this->success((new UserResource())->make($service->getUser()));
```

Izvorni kôd 7.20: Povratak resursa

```
public function make(Users $user): string
{
    $resource = new FractalItem(
        $user,
        new UserTransformer()
    );
    $this->manager->setSerializer($this->dataArraySerializer);
    return $this->manager->createData($resource)->toJson();
}
```

Izvorni kôd 7.21: Korisnički resurs

Resurs može sadržavati jedan objekt ili kolekciju objekata. S obzirom da se za prijavu vraćaju podaci prijavljenog korisnika, vraća se jedan objekt i za to se poziva metoda *make*. Ona poziva *UserTransformer* kako bi pripremila podatke i postavlja preostale potrebne parametre za vraćanje podataka.

```
class UserTransformer extends TransformerAbstract
{
```

```
    public function transform(Users $user): array
    {
        $avatarFile = $user->getFiles();

        $avatarInfo = [
            'avatar'      => null,
            'avatar_type' => null,
        ];
        if ( ! empty($avatarFile)) {
            $avatarInfo = [
                'avatar'      => $avatarFile->getFile(),
                'avatar_type' => $avatarFile->getType(),
            ];
        }

        $groups = [];
        $membership = $user->getMembers();
        foreach ($membership as $member) {
            /** @var Members $member */
            $groups[] = $member->getGroups();
        }

        return
            array_merge(
            [
                'id'          => $user->getId(),
                'email'       => $user->getEmail(),
                'subtitle'    => $user->getSubtitle(),
                'summary'     => $user->getSummary(),
                'username'    => $user->getUsername(),
                'enabled'     => $user->getEnabled(),
                'groups'      => (new UserGroupResource())->collectForResource(
                    $groups
                ),
                'tfa_enabled' => ! empty($user->getTfaSecret()),
                'is_premium'  => $user->isPremium(),
            ],
            $avatarInfo
        );
    }
}
```

Izvorni kôd 7.22: Transformer korisničkog resursa

Kako je transformer odgovoran za pripremu podataka, on dohvaća korisničke podatke te također vezane entitete (profilna slika i pridružene grupe) i ti podaci se spajaju u jedno polje koje se vraća natrag u resurs. Tako oblikovani podaci se vraćaju klijentskoj aplikaciji.

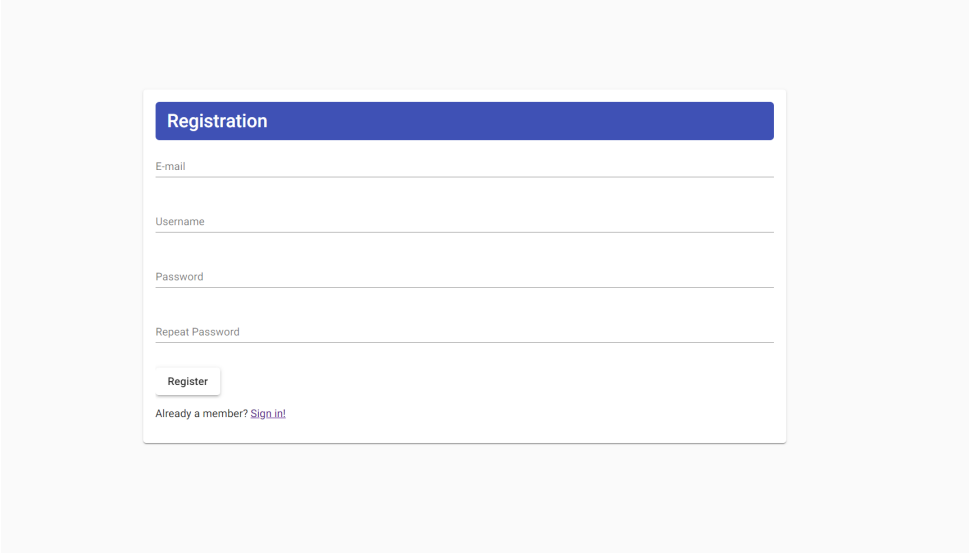
```
(response: User) => {
    this.authUser.user = response;
    this.router.navigate(['/profile']);
},
```

Izvorni kôd 7.23: Dohvat prijavljenog korisnika i preusmjeravanje na profilnu stranicu

Time je prijava uspješno odrađena i prikazani su svi koraci u toku obrade korisničke prijave. Istu ili sličnu logiku funkcioniranja prate sve preostale REST API krajnje točke.

7.8.2. Registracija

Kako bi korisnik postao član mreže i time dobio pristup korištenju iste, potrebno je obaviti postupak registracije. Registracija se može pristupiti pritiskom na poveznicu na prikazu prijave.



Slika 61: Prikaz registracijskog obrasca

Pogled registracije ostvaren je HTML kôdom prikazanim u isječku 7.24.

```
<mat-card-content fxLayout="column" fxLayoutGap="1em">
    <form *ngIf="!success" [formGroup]="form" (ngSubmit)="onSubmit()" fxLayout=
    ↪  "column" fxLayoutGap="1em">
        <mat-form-field>
            <mat-label>E-mail</mat-label>
            <input type="text" matInput [formControlName]="'email'">
        </mat-form-field>

        <mat-form-field>
            <mat-label>Username</mat-label>
            <input type="text" matInput [formControlName]="'username'">
        </mat-form-field>

        <mat-form-field>
            <mat-label>Password</mat-label>
            <input type="password" matInput [formControlName]="'password'">
        </mat-form-field>

        <mat-form-field>
            <mat-label>Repeat Password</mat-label>
            <input type="password" matInput [formControlName]="'password2'">
        </mat-form-field>

        <div fxLayout="row">
```

```
            <button mat-raised-button type="submit">Register</button>
        </div>

        <p>Already a member? <a routerLink="/login">Sign in!</a></p>
    </form>

    <div *ngIf="success">
        <p>A verification email has been sent to your e-mail address. You will need
        ↪  to verify before you can
        login.</p>
        <p>Click <a routerLink="/login">here</a> to return to the login page.</p>
    </div>
</mat-card-content>
```

Izvorni kôd 7.24: HTML kôd registracijske stranice

Struktura je ista kao i kod prijave. HTML se sastoji od kartice koja sadrži obrazac i preostale elemente. Poruka o uspjehu se uvjetno prikazuje ovisno o vrijednosti zastavice *success*.

Na strani programske logike komponente, prilikom inicijalizacije ponovno se priprema model obrasca sa zadanim vrijednostima i pripadnim validatorima.

```
this.form = this.formBuilder.group({
    email: ['', [Validators.required]],
    username: ['', [Validators.required]],
    password: ['', [Validators.required]],
    password2: ['', [Validators.required]]
});
```

Izvorni kôd 7.25: Inicijalizacija obrasca registracije

Obrazac prima podatke o e-mail adresi, korisničkome imenu i dva polja za lozinku i ponavljanje lozinke. Ti podaci se prilikom predaje obrasca sakupljaju i šalju prema poslužitelju, prikazano u isječku 7.26.

```
const payload: RegistrationRequestPayload = {
    email: this.form.get('email')?.value,
    username: this.form.get('username')?.value,
    password: this.form.get('password')?.value
}
this.subscription = this.usersApi.register(payload).subscribe(
    (response) => {
        this.success = true;
    },
    (error: HttpErrorResponse) => {
        this.errorService.handleErrors(error, this.form);
    }
);
```

Izvorni kôd 7.26: Slanje podataka o registraciji

Poslužitelj zaprima zahtjev na određenoj URL adresi koja se preusmjerava na pripadni kontroler i njegovu metodu. Ona poziva servis koji obrađuje registraciju i vraća podatke o korisniku.

```
public function register(): ResponseInterface
{
    if ($this->request->isPost()) {
        $service = new RegisterUserService();

        return (new Response())
            ->setSuccessPayload(
                (new UserResource())->make(
                        $service->handle(
                        $this->request->getJsonRawBody(true)
                    )
                )
            )
        ->send();
    }

    return $this->badRequest();
}
```

Izvorni kôd 7.27: Registracijska metoda kontrolera za autentikaciju

Servis inicijalizira objekt Phalconove ugrađene klase *Security* pomoću koje se kreira kriptografski sažetak korisničke lozinke. Parametri se prosljeđuju u korisnički repozitorij pozivanjem njegove metode *register*;

```
        public function handle(array $parameters): Users
        {
            $security = new Security();

            $encryptedPassword = $security->hash($parameters['password']);
            $parameters['password'] = $encryptedPassword;

            $user = $this->userRepository->register($parameters);
        }
```

Izvorni kôd 7.28: Registracijski servis

Repozitorijska metoda *register* kreira novi objekt korisničkog zapisa i postavlja njegove vrijednosti te ga sprema u bazu podataka. Ukoliko je uspješno spremljen, isti se vraća, a u protivnome se u aplikacijski dnevnik zapisuje greška.

```
    public function register(
    array $params
    ): Users {
        $container = FactoryDefault::getDefault();
        $user = new Users();

```

```
7           $user->setEmail($params['email']);
8           $user->setUsername($params['username']);
9           $user->setPassword($params['password']);
10          $user->setEnabled(0);
11
12          $success = $user->save();
13          if ( ! $success) {
14              $context = [];
15              foreach ($user->getMessages() as $message) {
16                  $context[] = [
17                      $message->getCode(),
18                      $message->getField(),
19                      $message->getMessage(),
20                      $message->getMetaData(),
21                  ];
22              }
23
24              /** @var Logger $logger */
25              $logger = $container->getShared('logger');
26              $logger->error(
27                  'Error while creating User!',
28                  $context
29              );
30          }
31
32          return $user->refresh();
33      }
```

Izvorni kôd 7.29: Korisnički repozitorij - registracija

Naposljetku, vraćeni korisnički zapis se koristi kako bi se u bazi podataka kreirao zapis za verifikaciju korisnika koji sadrži jedinstven token vezan uz tog korisnika. Nakon toga se korisniku šalje e-mail pošta s poveznicom za odrađivanje verifikacije.

```
1   $verification = $this->verificationRepository->create($user->getId());
2
3   $this->sendMailService->handle(
4       "Please verify your Commune account",
5       [$user->getEmail()],
6       'user_verify',
7       [
8           'user' => $user->getUsername(),
9           'token' => $verification->getToken(),
10      ]
11  );
12
13  return $user;
```

Izvorni kôd 7.30: Zapisivanje registriranog korisnika i verifikacije s popratnim slanjem poruke elektroničke pošte

Servis *sendMailService* odgovoran je za slanje poruka i prima nekoliko parametera:

- Predmet poruke

- Popis primatelja
- Naziv HTML predloška za e-mail poruku
- Popis dinamićkih varijabli za predložak

Servis iz globalnog dijeljenog spremnika dohvaća instancu klase poznate biblioteke PHPMailer pomoću koje se šalju poruke elektroničke pošte. Pripremaju se proslijeđeni podaci i sadržaj e-mail poruke u metodi *parseHtml* kojoj se prosljeđuju naziv predloška i varijable.

```php
public function handle(string $subject, array $recipients, string $templateName, array $templateVars)
{
    /** @var PHPMailer $mailer */
    $mailer = $this->container->get(GlobalServicesEnum::EMAIL);

    foreach ($recipients as $recipient) {
        $mailer->addAddress($recipient);
    }
    $mailer->Subject = $subject;
    $mailer->msgHTML($this->parseHtml($templateName, $templateVars));
    $mailer->send();
}
```

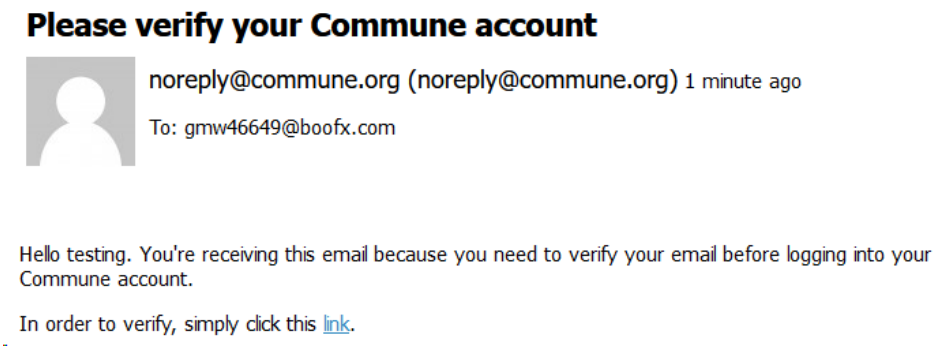
Izvorni kôd 7.31: Servis slanja e-mail pošte

Metoda *parseHtml* jednostavno učitava sadržaj HTML datoteke predloška u varijablu i provodi zamjenu podstringova unutar nje na temelju Regex pravila (regularnih izraza, engl. *regular expression*).

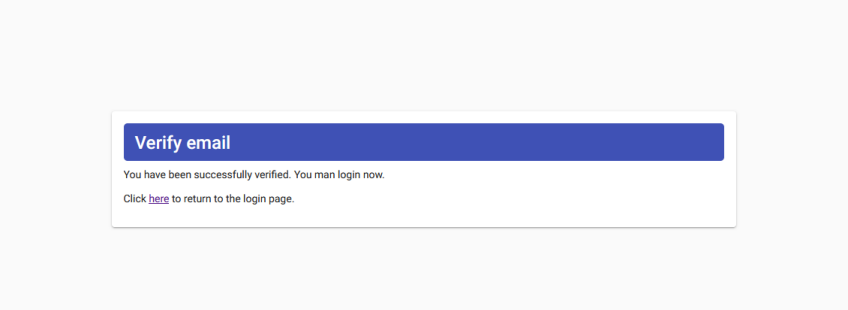
```php
private function parseHtml(string $templateName, array $templateVars): string
{
    $body = file_get_contents(APP_PATH . '/Storage/Mail/' . $templateName . '.html');

    foreach ($templateVars as $key => $var) {
        $body = preg_replace("/:$key:/", $var, $body);
    }

    return $body;
}
```

Izvorni kôd 7.32: Dinamičko popunjavanje sadržaja e-maila

E-mail poruka koju korisnik dobije prikazana je na slici 62. Poveznica sadrži u sebi prije spomenuti verifikacijski token i posjećivanjem iste korisnik se preusmjerava na stranicu za verifikaciju vidljivu na slici 63.



Slika 62: Verifikacijska e-mail poruka



Slika 63: Uspješna verifikacija korisnika

```
<div class="container">
    <mat-card>
        <mat-card-title class="card-title">
            Verify email
        </mat-card-title>

        <mat-card-content>
            <div *ngIf="!success">
                Verifying your email, please wait...
            </div>

            <div *ngIf="success">
                <p>You have been successfully verified. You man login now.</p>
                <p>Click <a routerLink="/login">here</a> to return to the login
                ↪  page.</p>
            </div>
        </mat-card-content>
    </mat-card>
</div>
```

Izvorni kôd 7.33: HTML struktura verifikacijske stranice

Sama komponenta je jednostavna te prilikom inicijalizacije iz URL adrese vadi token te ga šalje na poslužitelj za provjeru.

```
this.token = this.activatedRoute.snapshot.paramMap.get('token');

```

```
if (this.token) {
    this.subscription = this.authApi.verify(this.token).subscribe(
        (result) => {
            this.success = result;
        },
        (error: HttpErrorResponse) => {
            this.snackBar.open(this.errorService.handleErrors(error), 'X');
            this.error = true;
        }
    );
} else {
    this.error = true;
}
```

Izvorni kôd 7.34: Obrada verifikacije na klijentskoj strani

Poslužitelj zaprima zahtjev i učitava se metoda *verify* pripadnog *AuthController* kontrolera koja poziva metodu *handle* servisa *VerifyUserService* prikazanog u isječku 7.35

```
public function handle(array $parameters)
{
    $verification = $this->verificationRepository->findByToken($parameters['token'
    ↪ ]);

    if (empty($verification)) {
        throw new NotFoundException();
    }

    $user = $verification->getUsers();

    $user = $this->userRepository->verify($user->getId());

    return $user->getEnabled();
}
```

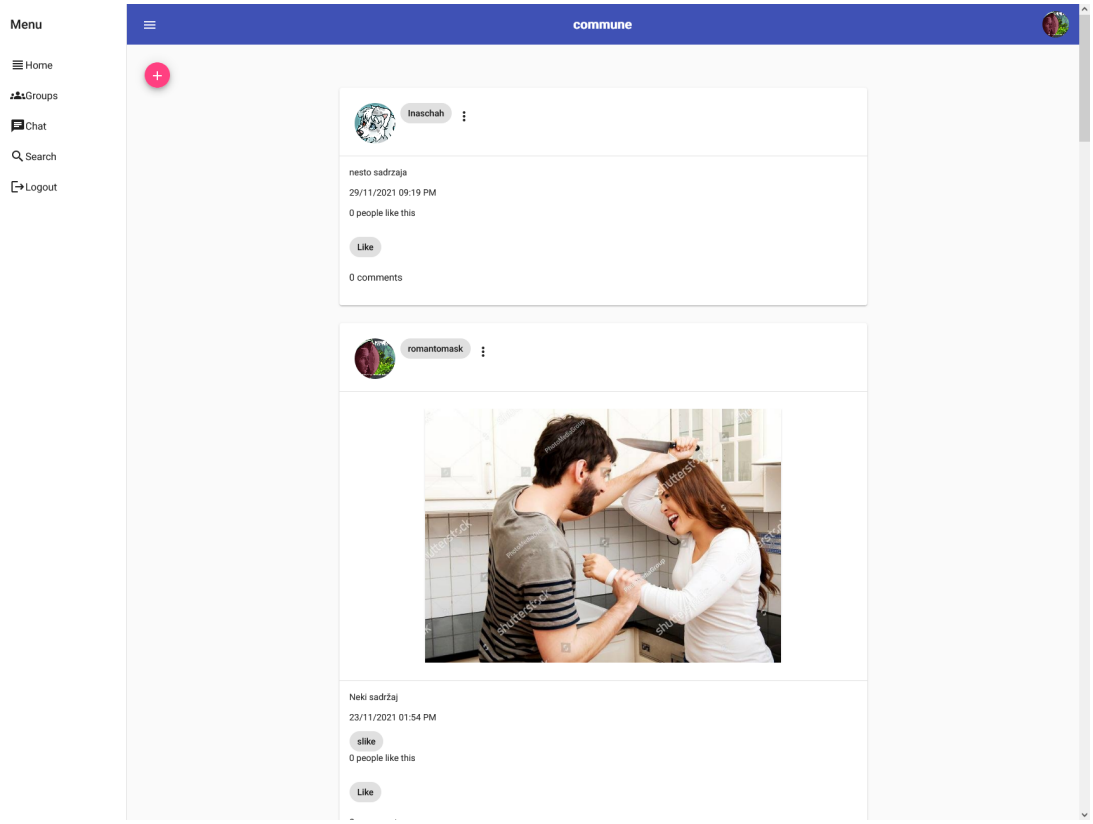
Izvorni kôd 7.35: Obrada verifikacije na poslužiteljskoj strani

Provjerava se da li zapis o verifikaciji postoji i ukoliko ne postoji vraća se greška 404. U protivnome se dohvaća povezani korisnički zapis i zastavica za aktivnost se postavlja na istinitu vrijednost.

Ukoliko je sve u redu, korisnik je uspješno verificiran i može se prijaviti u društvenu mrežu.

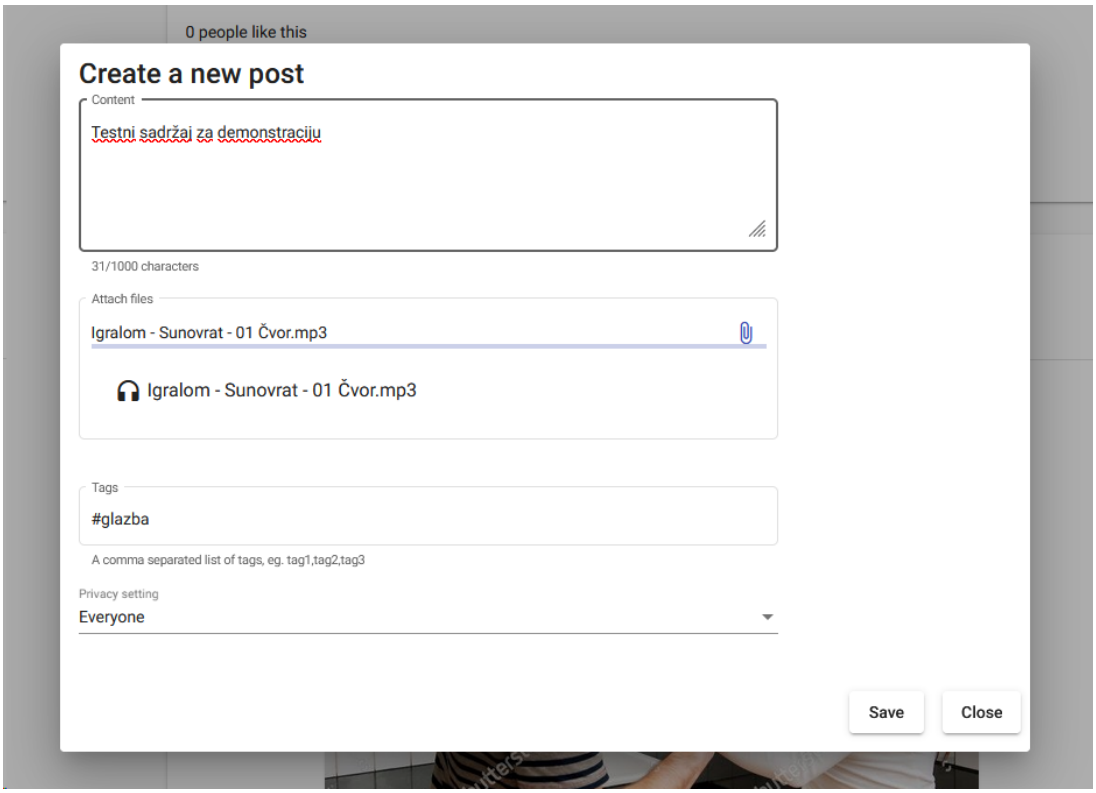
7.8.3. Objavljivanje

Korisnici mogu objavljivati različiti sadržaj unutar aplikacije u obliku tekstualnih objava koje također mogu biti popraćene slikama i snimkama. Korisnici su tako u mogućnosti pregledavati sadržaj koji drugi korisnici učitavaju u aplikaciju što je vidljivo na slici 64.



Slika 64: Prikaz objava

Kako bi stvorili novu objavu potrebno je stisnuti na gumb za dodavanje nove objave koji se nalazi na stranici s objavama ili na profilnoj stranici korisnika.



Slika 65: Obrazac za stvaranje nove objave

Slika 65 prikazuje dijaloški okvir s obrascem za dodavanje objave i isječak kôda 7.36 prikazuje model koji je vezan uz obrazac.

```
ngOnInit(): void {
    this.form = this.formBuilder.group({
        text: [null],
        files: [null],
        tags: [null, [postTagsValidator]],
        privacy: ['everyone']
    });

    this.group = this.data.group;

    this.subscriptions.add(
        this.form.get('files').valueChanges.subscribe(
            (files) => {
                this.files = files;
            }
        )
    )
}
```

Izvorni kôd 7.36: Priprema modela obrasca za novu objavu

Objava može sadržavati tekst, datoteke (slika, video, audio), oznake te sadrži postavku privatnosti. Također za svaku promjenu vrijednosti nad poljem datoteka ažurira se referenca na popis datoteka.

S obzirom da je već prikazano kako su u HTML-u realizirane kartice i obrasci, bit će prikazana jedina novost, a to je polje za unos datoteka koje je realizirano pomoću dodatne biblioteke *ngx-mat-file-input*.

```
<mat-form-field appearance="outline">
    <mat-label>Attach files</mat-label>
    <ngx-mat-file-input [formControlName]="'files'" [multiple]="true"></
    ↪  ngx-mat-file-input>
    <mat-progress-bar [value]="uploadProgress"></mat-progress-bar>

    <mat-list class="file-list">
        <mat-list-item *ngFor="let file of files">
            <mat-icon mat-list-icon>{{resolveFileIcon(file)}}</mat-icon>
            {{file.name}}
        </mat-list-item>
    </mat-list>
    <mat-error *ngIf="form.hasError('files')">{{form.getError('files')}}</mat-error>
</mat-form-field>
```

Izvorni kôd 7.37: HTML kôd polja za učitavanje datoteka

Unos za datoteke vezan je uz *files* polje unutar modela obrasca. Zastavicom *multiple* postavljenom na istinitu vrijednost dozvoljava se učitavanje većeg broja datoteka istovremeno. Informacije o učitanim datotekama prikazuju se unutar *<mat-list>* oznake koja sadrži pojedine elemente unutar *<mat-list-item>* oznake. Prikazuje se ikona ovisna o tipu datoteke i njeno ime.

Tip datoteke se iščitava pozivom metode *resolveFileIcon* koja kao parametar prima zapis o datoteci. Ona vrši provjeru *mime-type* na temelju Regex pravila kao što je vidljivo u isječku 7.38.

```
resolveFileIcon(file: File) {
    const imageRegex = /image/g;
    const audioRegex = /audio/g;
    const videoRegex = /video/g;

    if (imageRegex.test(file.type)) {
        return 'image';
    }

    if (audioRegex.test(file.type)) {
        return 'headphones';
    }

    if (videoRegex.test(file.type)) {
        return 'movie';
    }

    return 'attach_file';
}
```

Izvorni kôd 7.38: Provjera tipa datoteke

Kada je korisnik unio potrebne podatke, obrazac se predaje poslužitelju gdje se nastavlja obrada. Učitava se metoda *create* u kontroleru *PostController*. Ona poziva servis *CreatePostService* koji je odgovoran za kreiranje novih objava.

```
public function create()
{
    if ($this->request->isPost()) {
        $authUser = $this->session->get('user');
        $service = new CreatePostService();

        return (new Response())
            ->setSuccessPayload(
            (new PostResource())->make(
                $service->handle(
                    $authUser,
                    $this->request->getPost(),
                    $this->request->getUploadedFiles()
                )
            )
        )
        ->send();
    }

    return $this->badRequest();
}
```

Izvorni kôd 7.39: Obrada poziva za stvaranje nove objave

Budući da se prilikom kreiranje objave provodi više operacija nad bazom podataka, potrebno ih je provesti unutar transakcije.

```
public function handle(
        Users $authUser,
        array $parameters,
        array $files = []
    ): Posts {
    /** @var AdapterInterface $dbAdapter */
    $dbAdapter = $this->container->get(GlobalServicesEnum::DB);
    $dbAdapter->begin();
```

Izvorni kôd 7.40: Servis za kreiranje nove objave

Na početku obrade kreira se zapis o objavi sa podacima koji se prvo validiraju kao što je prikazano u isječku 7.42.

```
    $post = $this->postRepository->create(
        $authUser->getId(),
        $this->validateAndPrepareParamters($parameters)
    );
```

Izvorni kôd 7.41: Kreiranje objave s validiranim podacima

Prvenstveno što se u ovom koraku validira jest količina oznaka koje je korisnik unio za objavu, koja je ograničena na maksimalno 5 različitih oznaka.

```
private function validateAndPrepareParamters(array $parameters)
{
    $prepared = [];

    if (array_key_exists('text', $parameters)) {
        $prepared['text'] = $parameters['text'];
    }

    if (array_key_exists('tags', $parameters)) {
        $tags = explode(
            ',',
            preg_replace('/\s/', '', strtolower($parameters['tags']))
        );

        if (count($tags) > 5) {
            $messages = new Messages();
            $messages->appendMessage((new Message('Up to 5 tags allowed.', 'tags'
            ↪  )));

            throw new ValidationException($messages);
        }

        $prepared['tags'] = json_encode($tags);
    }

    if (array_key_exists('group_id', $parameters)) {
        $prepared['group_id'] = intval($parameters['group_id']);
    }

    if (array_key_exists('privacy', $parameters)) {
        $prepared['privacy'] = $parameters['privacy'];
    }

    return $prepared;
}
```

Izvorni kôd 7.42: Validacija podataka o objavi

Ukoliko su podaci u redu, kreće se na daljnju provjeru veličine datoteka koja ovisi o tome da li je korisnik pretplaćen ili ne.

```
        if ( ! empty($files)) {
            /** @var File $file */
            foreach ($files as $file) {
                if ($authUser->isPremium()) {
                    if ($file->getSize() > Files::FILE_SIZE_LIMIT_PREMIUM) {
                        $messages = new Messages();
                        $messages->appendMessage(new Message(
                        ↪  'Some of your files are too big', 'files'));
                        throw new ValidationException($messages);
                    }
                } else {
                    if ($file->getSize() > Files::FILE_SIZE_LIMIT) {
                        $messages = new Messages();
                        $messages->appendMessage(new Message(
                        ↪  'Some of your files are too big', 'files'));
                        throw new ValidationException($messages);
                    }
                }
```

Izvorni kôd 7.43: Validacija veličine datoteke

Ukoliko je datoteka dozvoljene veličine, dohvaćaju se osnovni podaci o njoj te se ona seli na posebno mjesto unutar datotečnog sustava gdje se čuvaju datoteke. Kreira se novi zapis o datoteci i veže se uz objavu.

```
                $fileType = $file->getType();
                $fileName = $file->getName();
                $file->moveTo(
                    APP_PATH.DIRECTORY_SEPARATOR.'..'.DIRECTORY_SEPARATOR.'uploads'.
                    ↪  DIRECTORY_SEPARATOR.$fileName
                );
                $dbFile = new Files();
                $dbFile->setFile($fileName)->setType($fileType)->save();
                $postFile = new PostFiles();
                $postFile->setPostId($post->getId())->setFileId(
                    $dbFile->getId()
                )->save();
            }
        }
```

Izvorni kôd 7.44: Obrada datoteka

Ako je tijekom obrade došlo do nekakve iznimke, transakcija se poništava, a u protivnom se završava te se vraća objekt koji sadrži podatke o objavi.

```
        $dbAdapter->commit();
    } catch (Throwable $throwable) {
        $dbAdapter->rollback();
        throw $throwable;
    }

```

```
7       return $post->refresh();
8   }
```

Izvorni kôd 7.45: Završavanje transakcije i vraćanje objave

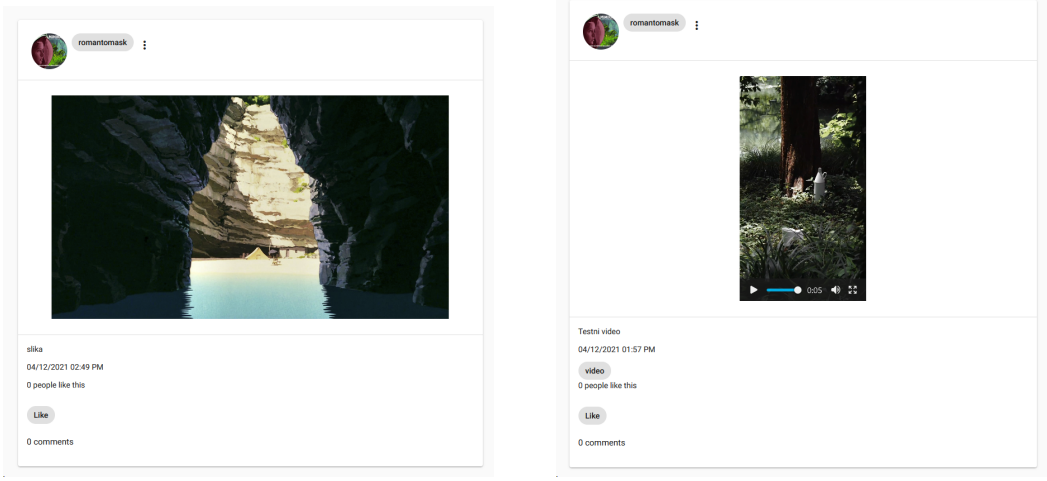
Vraćena objava se prosljeđuje u resurs *PostResource* (kako resursi funkcioniraju prikazano je u ranijim odjeljcima) i šalje prema klijentskoj aplikaciji.

Po zaprimanju podataka dijaloški prozor se zatvara i dobivena objava se prosljeđuje komponenti iz kojeg se on pozvao.

```
1   onPostCreated(post: Post) {
2       if (post) {
3           this.feedItems.unshift(post);
4       }
5   }
```

Izvorni kôd 7.46: Obrada novostvorene objave na klijentskoj aplikaciji

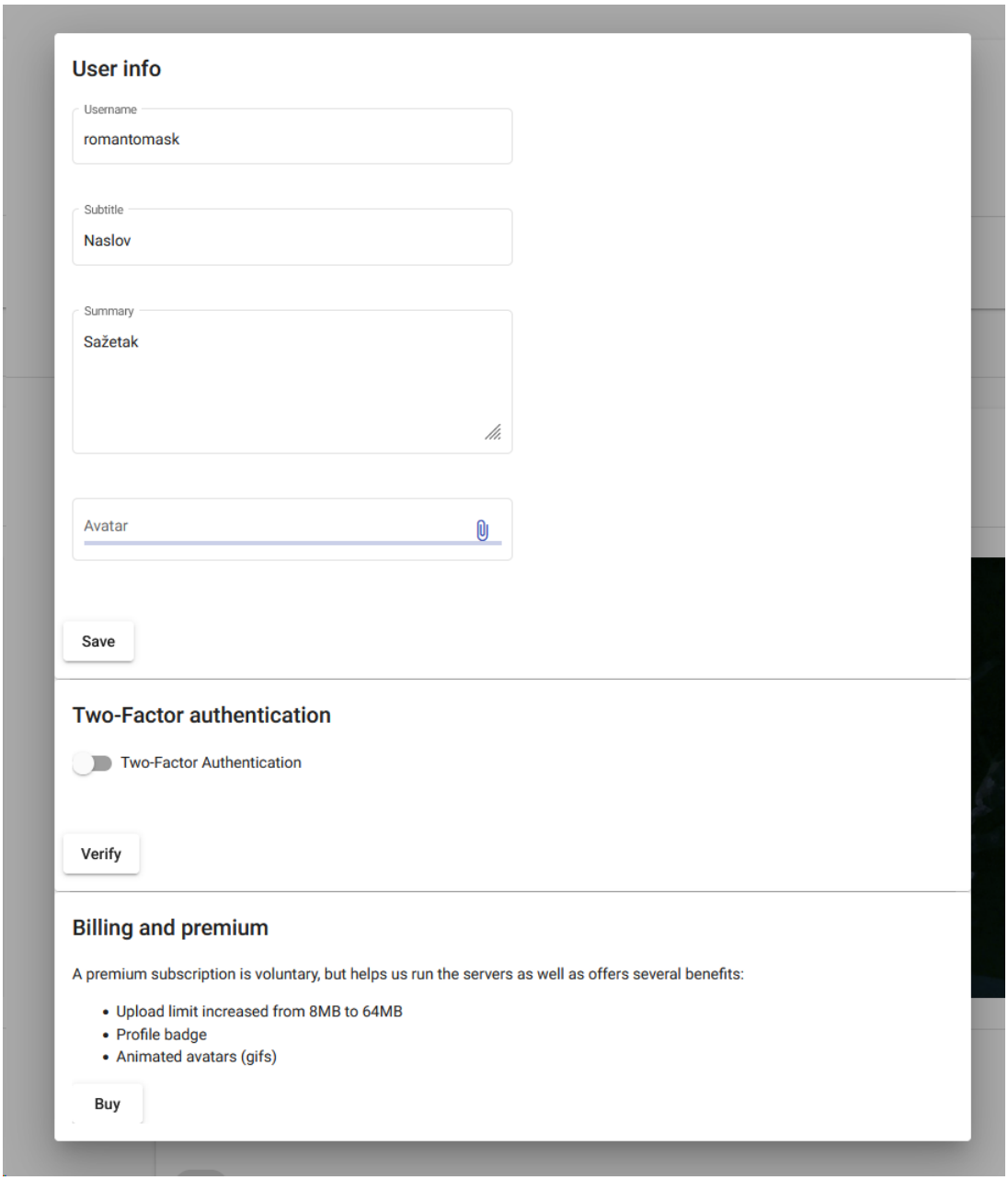
Provjerava se da li je vrijednost zapravo popunjena (ukoliko se okvir zatvori bez slanja objave, onda će se vratiti prazna vrijednost) i ona se umeće u popis objava kako bi se izbjeglo dodatno kontaktiranje poslužitelja.



Slika 66: Objave sa različitim vrstama sadržaja

7.8.4. Naplata

Društvena mreža nudi opciju naplate zauzvrat za dodatne korisničke pogodnosti u vidu povećanja dozvoljenih veličina datoteka i animiranih profilnih slika. Naplata je realizirana povezivanjem sa sustavom PayPal. Svaki korisnik može pokrenuti naplatu posjećivanjem svoje profilne stranice i otvaranjem korisničkih postavki gdje se na dnu nalazi gumb za pokretanje naplate, prikazano na slici 67.



Slika 67: Postavke korisničkog profila

Pritiskom gumba šalje se zahtjev prema poslužitelja kako bi se kreirao zapis o zahtjevu naplate. Na poslužiteljskoj strani pokreće se metoda *createOrder* kontrolera *BillingController* koja poziva servis *CreateOrderService* koji je prikazan u isječku 7.48.

```php
public function createOrder()
{
    if ($this->request->isPost()) {
        $user = $this->session->get('user');
        $service = new CreateOrderService();

        return $this->success(json_encode($service->handle($user)));
    }

    return $this->badRequest();
}
```

Izvorni kôd 7.47: Kontrolerska metoda za stvaranje naplate

Koristeći PayPal-ovu klijentsku biblioteku, priprema se zahtjev sa parametrima koji se onda šalju PayPal-ovom poslužitelju. Parametri su vidljivi u isječku 7.49.

```php
public function handle(Users $authUser)
{
    $request = new OrdersCreateRequest();

    $request->prefer('return=representation');

    $request->body = $this->buildRequestBody();

    $client = PayPalClient::client();

    try {
        $response = $client->execute($request);
        $paypalCreateOrderResponse = new PaypalCreateOrderResponse($response->result
        ↪  , $response->headers);

        $this->orderRepository->create($authUser, $paypalCreateOrderResponse);

        return $paypalCreateOrderResponse->links[1]->href;
    } catch (HttpException $e) {
        throw $e;
    } catch (IOException $e) {
        throw $e;
    }
}
```

Izvorni kôd 7.48: Servis za stvaranje zahtjeva naplate

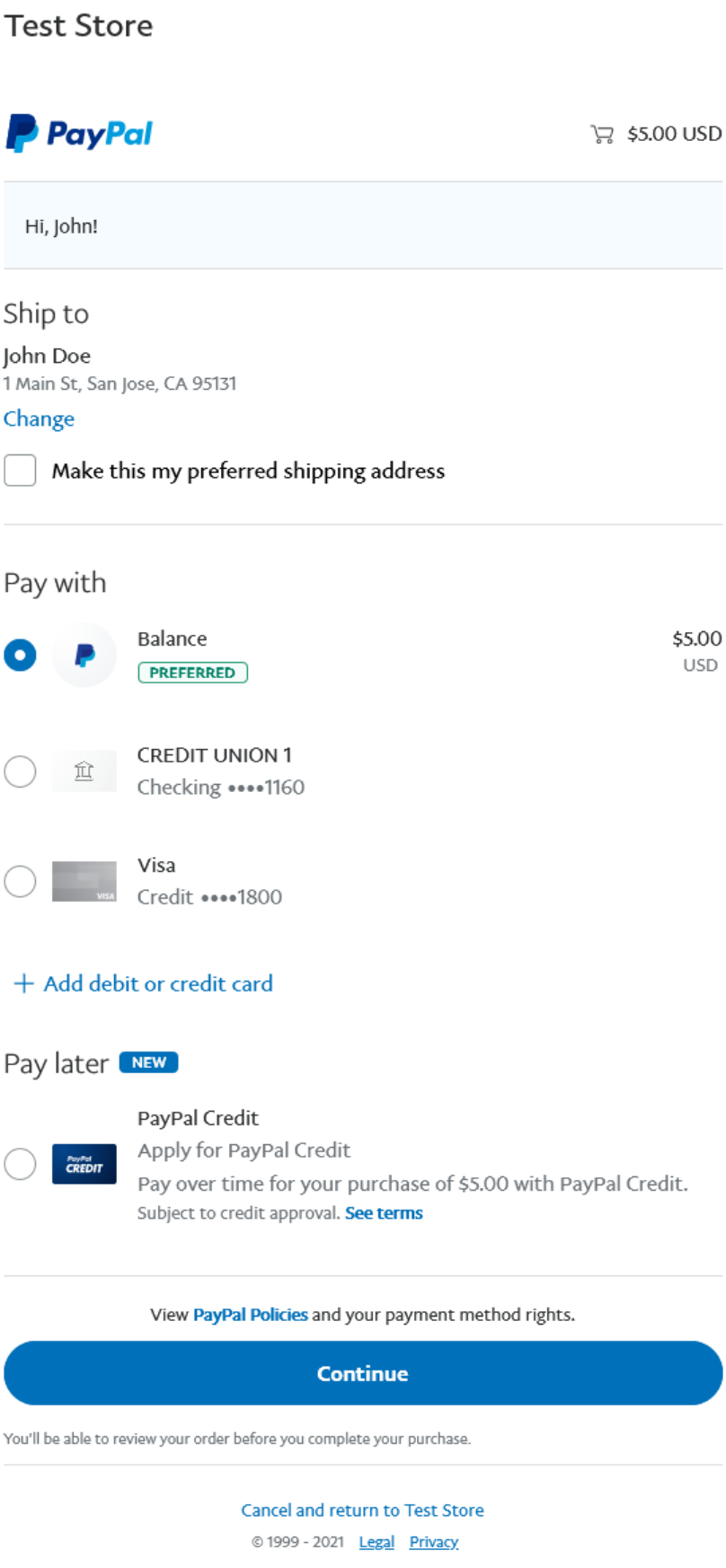
Zahtjev se šalje na PayPal-ov poslužitelj i natrag se dobiva odgovor na temelju kojeg se u bazu podataka zapisuje zahtjev za naplatu te se vraća URL za izvršenje naplate.

```php
[
    'intent'              => 'CAPTURE',
    'application_context' =>
        [
            'return_url' => $_ENV['SUCCESS_URL'],
```

```
            'cancel_url' => $_ENV['CANCEL_URL'],
        ],
    'purchase_units'      =>
        [
            [
                "reference_id" => "test_ref_id1",
                'description'  => 'Commune premium',
                'amount'       =>
                    [
                        'currency_code' => 'USD',
                        'value'         => '5.00',
                    ],
            ],
        ],
]
```

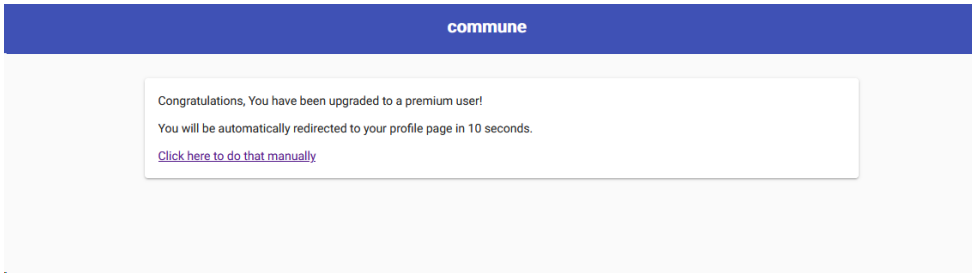
Izvorni kôd 7.49: Parametri za PayPal naplatu

Korisnik onda biva preusmjeren na spomenuti URL koji vodi do PayPal-ove stranice za naplatu, prikazane na slici 68.



Slika 68: PayPal stranica za naplatu

Ukoliko je naplata uspješna, korisnik će biti preusmjeren na posebnu stranicu klijentske aplikacije prikazane na slici 69. Vezana komponenta šalje zahtjev poslužitelju da zapiše uspješnu naplatu i ažurira zapis u bazi podataka.



Slika 69: Uspješna naplata

Poziva se *captureOrder* metoda kontrolera *BillingController* koji instancira servis *CaptureOrderService* odgovoran za obradu uspješne naplate.

```php
public function captureOrder()
{
    if ($this->request->isPost()) {
        $user = $this->session->get('user');
        $service = new CaptureOrderService();

        return $this->success((new OrderResource())->make($service->handle($user)));
    }

    return $this->badRequest();
}
```

Izvorni kôd 7.50: Obrada zahtjeva

Servis poziva metodu *capture* repozitorija *OrderRepository* koja jednostavno ažurira stanje narudžbe i time su dodatne pogodnosti od sada nadalje aktivne za tog korisnika.

```php
public function handle(Users $authUser)
{
    return $this->orderRepository->capture($authUser);
}
```

Izvorni kôd 7.51: Zapisivanje zahtjeva u bazu podataka

Vraća se ažurirani zapis o narudžbi iz baze podataka koji se prosljeđuje resursu i transformatoru te poslati natrag klijentskoj aplikaciji.

7.9. REST API

Slijedi kratka dokumentacija koja opisuje dio krajnjih točkaka REST API-ja razvijenoga za poslužiteljsku stranu projekta kako bi se prikazala njegova struktura.

7.9.1. Prijava

POST **/login**

prijava korisnika

Body application/json

```
{
    "email": "korisnik@korisnik.hr",
    "password": "zaPoRKa551"
}
```

Response application/json

200 ok

```
{
    "id": "19",
    "email": "korisnik@korisnik.hr",
    "subtitle": "samo obican korisnik",
    "summary": "najbolji spaghetti majstor u gradu",
    "username": "Majstor",
    "enabled": "1",
    "groups": [
        {...}
    ],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": "20c0aee8-3f5c-49b3-80bb-ea1ee14e8968",
    "avatar_type": "image/png"
}
```

7.9.2. Odjava

POST **/logout**

odjava korisnika

Response application/json

204 ok

7.9.3. Zaboravljena lozinka

POST **/init-password-reset**

zahtjev ponovnog postavljanja zaporke

Body application/json

```
{
    "email": "korisnik@korisnik.hr",
}
```

Response application/json

204 ok

POST **/reset-password**

Ponovno postavljanje zaporke

Body application/json

```
{
    "password": "a5m6pfENL3rgtBL",
    "token": "f0e696d209fad7f8ab628b739f870a4966f2bea1"
```

```
}
```

Response application/json

200 ok


```
{
    "id": "19",
    "email": "korisnik@korisnik.hr",
    "subtitle": "samo obican korisnik",
    "summary": "najbolji spaghetti majstor u gradu",
    "username": "Majstor",
    "enabled": "1",
    "groups": [
        {
            "id": "1",
            "name": "developerska",
            "owner": "14",
            "avatar_file": null,
            "cover_file": null,
            "created_at": "2021-11-18 21:52:51",
            "updated_at": null
        }
    ],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": "20c0aee8-3f5c-49b3-80bb-ea1ee14e8968",
    "avatar_type": "image/png"
}
```


7.9.4. Višefaktorska autentikacija

GET **/generate-tfa-secret**

generira QR-kôd za aktivaciju višefaktorske autentikacije

Response application/json

200 ok


```
{
    "secret": VF6MQNJGGMJUBTL2,
```

```
    "qr_code": "+063kd76yDeJEb4H3xtWXslVkYkcwvlqaJad8XzIlNYN9j145
    uhqB2mX/D14UcwEP5kSbMlthWgnMy6eUn/esH/rWpKXXr4Xm1ecHlbwimw06Q4x
    600SjNPfZ+fEdvMkbyPtqxn8Kxw92i3/igWXZ+ErGS+r5KCIf7S8Il4Rr4
    hXxCviFfGKeEW8Il4Rr8im0P8ALm8vyzt5VmUAAAAASUVORK5CYII=",
    "qr_text": otpauth://totp/user?secret=VF6MQNJGGMJUBTL2&issuer=
    commune&period=30&algorithm=SHA1&digits=6
}
```

POST **/activate-tfa**

Ponovno postavljanje zaporke

Body application/json

```
{
    "code": 123456,
}
```

Response application/json

200 ok

```
                {
                    "id": "19",
                    "email": "korisnik@korisnik.hr",
                    "subtitle": "samo obican korisnik",
                    "summary": "najbolji spaghetti majstor u gradu"
    ,
                    "username": "Majstor",
                    "enabled": "1",
                    "groups": [
                    {...}
                    ],
                    "tfa_enabled": true,
                    "is_premium": false,
                    "avatar": "20c0aee8-3f5c-49b3-80bb-ea1ee14e8968
    ",
                    "avatar_type": "image/png"
                }
```

POST **/activate-tfa**

Ponovno postavljanje zaporke

Response application/json

200 ok

```
{
    "id": "19",
    "email": "korisnik@korisnik.hr",
    "subtitle": "samo obican korisnik",
    "summary": "najbolji spaghetti majstor u gradu"
,
    "username": "Majstor",
    "enabled": "1",
    "groups": [
    {...}
    ],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": "20c0aee8-3f5c-49b3-80bb-ea1ee14e8968
",
    "avatar_type": "image/png"
}
```

GET **/users/{id}**

dohvacanje korisnika

Parameter

id id korisnika

Response application/json

200 ok

```
{
    "id": "6",
    "email": "romantomask@gmail.com",
    "subtitle": "Naslov",
    "summary": "Sa\u017eetak",
```

```
    "username": "romantomask",
    "enabled": "1",
    "groups": [],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": "ce894177-6170-42bd-8b72-3ba24bf3fa54",
    "avatar_type": "image\/jpeg"
}
```


7.9.5. Korisnički profil

GET **/users/profile**

dohvaćanje profila korisnika

Response application/json

200 ok


```
{
    "id": "19",
    "email": "korisnik@korisnik.hr",
    "subtitle": "samo obican korisnik",
    "summary": "najbolji spaghetti majstor u gradu",
    "username": "Majstor",
    "enabled": "1",
    "groups": [
        {...}
    ],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": "20c0aee8-3f5c-49b3-80bb-ea1ee14e8968",
    "avatar_type": "image\/png"
}
```


7.9.6. Registracija

POST **/users/register**

registracija korisnika

Body application/json

```
{
    "email": "testni@mail.hr",
    "username": "korisnik",
    "password": "zaporka54321"
}
```

Response application/json

200 ok

```
{
    "id": "33",
    "email": "testni@mail.hr",
    "subtitle": null,
    "summary": null,
    "username": "korisnik",
    "enabled": "0",
    "groups": [],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": null,
    "avatar_type": null
}
```

7.9.7. Ažuriranje korisnika

POST **/users/update/{id}**

azuriranje profila korisnika

Parameter

id id korisnika

Body application/json

```
{
```

```
    "username": "korisnik",
    "subtitle": "majstor",
    "summary": "majstor na kvadrat"
}
```

Response application/json

200 ok

```
{
    "id": "19",
    "email": "korisnik@korisnik.hr",
    "subtitle": "samo obican korisnik",
    "summary": "najbolji spaghetti majstor u gradu",
    "username": "Majstor",
    "enabled": "1",
    "groups": [
        {...}
    ],
    "tfa_enabled": false,
    "is_premium": false,
    "avatar": "20c0aee8-3f5c-49b3-80bb-ea1ee14e8968",
    "avatar_type": "image\/png"
}
```

7.9.8. Dohvaćanje korisničkih objava

POST **/posts/user/{id}**

dohvacanje objava korisnika

Parameter

id	id korisnika
page	broj stranice objava
page_size	kolicina objava po stranici

Response application/json

200 ok

```
{
    "id": "12",
    "text": "Neki sadrzaj",
    "tags": [
        "slike"
    ],
    "files": [
        {
            "id": "24",
            "file": "MIMIKA-FRONT.jpg",
            "type": "image\/jpeg",
            "created_at": "2021-11-23 13:54:02",
            "updated_at": null
        }
    ],
    "author": {...},
    "created_at": "2021-11-23 13:54:02",
    "comments": [],
    "comments_count": 0,
    "likes_count": 0,
    "likes": []
}
```


7.9.9. Brisanje objave

DELETE **/posts/{id}**

brisanje objave

Parameter

id id objave

Response application/json

200 ok

```
true
```

8. Zaključak

Kroz ovaj rad prikazana je izrada kompleksne aplikacije pomoću PhalconPHP razvojnog okvira i način monetizacije aplikacije. Pripadna aplikacija je društvena mreža koja omogućuje korisnicima objavljivanje sadržaja, međusobnu komunikaciju putem komentiranja i dopisivanja, pretragu sadržaja te stvaranje i pridruživanje grupama. Pored izrade, pokrile su se teorijske osnove društvenih mreža i pobliže je prikazana njihova opsežna povijest. Uz PhalconPHP, odabran je Angular razvojni okvir za izradu klijentske strane aplikacije.

Razvoj aplikacije opisan je na više razina, od tehničke prikazom pripadnih dijagrama koji opisuju arhitekturu sustava, njegove sastavne dijelove kao i detalje putem dijagrama korištenja, dijagrama klasa i slično pa sve do implementacijske razine opisom pojedinih dijelova aplikacije i načina na koji oni surađuju i bivaju u interakciji jedno s drugim. Različiti alati i programske biblioteke koje su korištene kako bi pospješile i pojednostavile razvoj su također bile prikazane.

Monetizacija aplikacije ostvarena je uvođenjem plaćenog modela korisnika koji se postiže jednokratnom uplatom te koji korisniku osigurava dodatne pogodnosti prilikom korištenja aplikacije. Također je otvoren korisnički račun na servisu za dobrovoljnu podršku kreatora zvanom Patreon gdje se korisnici mogu pretplatiti na mjesečne donacije, čime je cilj pokriti cijenu održavanja aplikacije.

Društvene mreže kompleksni su sustavi i zahtijevaju poznavanje više programskih jezika i alata. Potrebno je također razumijevanje drugih tehnologija kao što su baze podataka i web poslužitelji. Razni slojevi arhitekture moraju međusobno moći komunicirati i surađivati što zahtjeva ispravnu konfiguraciju pojedinih dijelova.

Cilj rada bio je primjerom prikazati proces izrade društvene mreže koristeći se PhalconPHP razvojnim okvirom. Društvene mreže su imale dugu povijest razvoja i prošle kroz nekoliko generacija kako se informatička tehnologija razvijala, ali u jednu ruku može se danas ustvrditi da društvene mreže doživljavaju svojevrsnu stagnaciju. Daljnjim razvojem web tehnologija kao što su Web 3 to će se svakako ponovno promijeniti.

Popis literature

Ahmed, Reshma (13. srpnja 2020.). What Is Git | Explore A Distributed Version Control Tool. Edureka. Section: DevOps. URL: https://www.edureka.co/blog/what-is-git/ (pogledano 7. 12. 2021.).

Almiray, Andres (15. prosinca 2015.). MVC Patterns. URL: http://aalmiray.github.io/griffon-patterns/ (pogledano 25. 7. 2021.).

Angular Components Team (11. prosinca 2021.). Angular Material. Angular Material. URL: https://material.angular.io/ (pogledano 11. 12. 2021.).

Angular Tim (2021.a). Attribute directives. URL: https://angular.io/guide/attribute-directives (pogledano 25. 7. 2021.).

— (2021.b). Built-in directives. URL: https://angular.io/guide/built-in-directives (pogledano 25. 7. 2021.).

— (2021.c). Components. URL: https://angular.io/guide/architecture-components (pogledano 25. 7. 2021.).

— (2021.d). Modules. URL: https://angular.io/guide/architecture-modules (pogledano 25. 7. 2021.).

— (2021.e). Services. URL: https://angular.io/guide/architecture-services (pogledano 25. 7. 2021.).

— (2021.f). Structural directives. URL: https://angular.io/guide/structural-directives (pogledano 25. 7. 2021.).

Apache Friends (2021.). XAMPP Installers and Downloads for Apache Friends. URL: https://www.apachefriends.org/index.html (pogledano 2. 12. 2021.).

Arrington, Michael (14. ožujka 2008.). Facebook To Launch Instant Messaging Service. TechCrunch. URL: https://social.techcrunch.com/2008/03/14/facebook-to-launch-instant-messaging-service/ (pogledano 26. 7. 2021.).

Atlassian (2021.). What is DevOps? Atlassian. URL: https://www.atlassian.com/devops (pogledano 10. 12. 2021.).

Bartle, Richard A. (15. studenoga 1990.). Richard A. Bartle: Early MUD History. URL: https://mud.co.uk/richard/mudhist.htm (pogledano 25. 7. 2021.).

— (21. siječnja 1999.). Richard A. Bartle: HOTLINE TO FANTASY. URL: https://mud.co.uk/richard/pcpsep87.htm (pogledano 16. 8. 2021.).

— (2016.). MMOs from the inside out: the history, design, fun, and art of massively-multiplayer online role-pla Books for professinals by professionals. New York, NY: Apress. 735 str. ISBN: 978-1-4842-1724-5 978-1-4842-1723-8.

Briggs, Asa, Peter Burke i Espen Ytreberg (2020.). A social history of the media: from Gutenberg to Facebook. Fourth edition. Medford: Polity. 1 str. ISBN: 978-1-5095-3374-9.

Britannica, Encyclopedia (2021.). Twitter | History, Description, & Uses. Encyclopedia Britannica. URL: https://www.britannica.com/topic/Twitter (pogledano 23. 7. 2021.).

Brooks, Aaron (9. svibnja 2019.). [Timeline] A Brief History of Influencers. Social Media Today. URL: https://www.socialmediatoday.com/news/timeline-a-brief-history-of-influencers/554377/ (pogledano 16. 6. 2021.).

Brotherton, Claire (6. srpnja 2021.). The Most Popular PHP Frameworks to Use in 2021. Kinsta. URL: https://kinsta.com/blog/php-frameworks/ (pogledano 26. 8. 2021.).

Burbeck, Steve (1992.). How to use Model-View-Controller (MVC). URL: https://web.archive.org/web/20120729161926/http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html (pogledano 19. 7. 2021.).

Burgess, Jean i Joshua Green (2018.). Youtube: online video and participatory culture. Second edition. Digital media and society. Cambridge, UK ; Medford, MA: Polity Press. 1 str. ISBN: 978-1-5095-3359-6.

Cai, Jason, Ranjit Kapila i Guarav Pal (21. srpnja 2000.). HMVC: The layered pattern for developing strong clie URL: https://web.archive.org/web/20160317103852/http://www.javaworld.com/article/2076128/design-patterns/hmvc--the-layered-pattern-for-developing-strong-client-tiers.html (pogledano 25. 7. 2021.).

Charalabidis, Alexander (2000.). The book of IRC. San Francisco : Berkeley, Calif: No Starch Press ; Publishers Group West [distributor]. 347 str. ISBN: 978-1-886411-29-6.

Comscore (24. lipnja 2010.). Comscore Releases May 2010 U.S. Online Video Rankings. Comscore, Inc. URL: https://www.comscore.com/Insights/Press-Releases/2010/6/Comscore-Releases-May-2010-US-Online-Video-Rankings (pogledano 26. 7. 2021.).

Dailymotion (25. lipnja 2021.). Wikipedia. Page Version ID: 1030377892. URL: https://en.wikipedia.org/w/index.php?title=Dailymotion&oldid=1030377892 (pogledano 27. 7. 2021.).

, danah m. boyd danah m. i Nicole B. Ellison (1. listopada 2007.a). „Social Network Sites: Definition, History, and Scholarship". Journal of Computer-Mediated Communication 13.1, str. 210–230. ISSN: 1083-6101. DOI: 10.1111/j.1083-6101.2007.00393.x. URL: https://doi.org/10.1111/j.1083-6101.2007.00393.x (pogledano 29. 5. 2021.).

— (listopad 2007.b). „Social Network Sites: Definition, History, and Scholarship". Journal of Computer-Mediate 13.1, str. 210–230. ISSN: 10836101. DOI: 10.1111/j.1083-6101.2007.00393.x. URL: https://academic.oup.com/jcmc/article/13/1/210-230/4583062 (pogledano 23. 7. 2021.).

Darwin, Pete Bacon (11. kolovoza 2020.). Stable AngularJS and Long Term Support. Medium. URL: https://blog.angular.io/stable-angularjs-and-long-term-support-7e077635ee9c (pogledano 25. 7. 2021.).

Dean, Brian (25. veljače 2021.). Reddit Usage and Growth Statistics: How Many People Use Reddit in 2021? Backlinko. URL: https://backlinko.com/reddit-users (pogledano 26. 7. 2021.).

Dijck, José van (2013.). The culture of connectivity: a critical history of social media. Oxford ; New York: Oxford University Press. 228 str. ISBN: 978-0-19-997077-3 978-0-19-997078-0.

Doyle, Brandon (14. lipnja 2021.). TikTok Statistics - Everything You Need to Know [Feb 2021 Update]. Wallaroo Media. URL: https://wallaroomedia.com/blog/social-media/tiktok-statistics/ (pogledano 20. 7. 2021.).

Dyer, Russell J. T. (2015.). Learning MySQL and MariaDB. First edition. OCLC: ocn881022270. Sebastopol, CA: O'Reilly. 381 str. ISBN: 978-1-4493-6290-4.

Encyclopedia Britannica (2021.). USENET | Definition, History, & Facts. Encyclopedia Britannica. URL: https://www.britannica.com/technology/USENET (pogledano 26. 7. 2021.).

Frier, Sarah (2020.). No filter: the inside story of Instagram. New York: Simon & Schuster. ISBN: 978-1-982126-80-3 978-1-982126-81-0.

Gavigan, Dave (25. svibnja 2018.). The History of Angular. Medium. URL: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7 (pogledano 25. 7. 2021.).

GitLab (2021.a). Features. GitLab. URL: https://about.gitlab.com/features/ (pogledano 2. 12. 2021.).

— (2021.b). Try GitLab for free. GitLab. URL: https://about.gitlab.com/free-trial/ (pogledano 2. 12. 2021.).

Hall, Mark (2021.). Facebook | Overview, History, & Facts. Encyclopedia Britannica. URL: https://www.britannica.com/topic/Facebook (pogledano 5. 6. 2021.).

Harrison, Guy (2015.). Next generation databases: NoSQL, NewSQL, and Big Data. U sur. s International Oracle Users Group. The expert's voice in Oracle. OCLC: ocn920849271. New York: Apress, IOUG. 235 str. ISBN: 978-1-4842-1330-8.

Hempel, Jessi (2021.). „Inside Reddit's Plan to Recover From Its Epic Meltdown". Wired (). Section: tags. ISSN: 1059-1028. URL: https://www.wired.com/2015/10/reddit-survived-meltdown-can-fix/ (pogledano 26. 7. 2021.).

Hiwarale, Uday (1. rujna 2020.). A beginner's guide to TypeScript (with history). Medium. URL: https://medium.com/jspoint/typescript-a-beginners-guide-6956fe8bcf9e (pogledano 18. 4. 2021.).

Jablonski, Adam i Marek Jablonski (2021.). Digital business models: perspectives on monetisation. Routledge studies in innovation, organizations and technology. London ; New York: Routledge, Taylor & Francis Group. 1 str. ISBN: 978-0-429-32267-9.

JetBrains (2021.a). 25 Years of PHP History. JetBrains: Developer Tools for Professionals and Teams. URL: https://www.jetbrains.com/lp/php-25 (pogledano 6. 3. 2021.).

— (2021.b). Features - PhpStorm. JetBrains. URL: https://www.jetbrains.com/phpstorm/features/ (pogledano 2. 12. 2021.).

JoinPeerTube (2021.). JoinPeerTube. URL: https://joinpeertube.org (pogledano 27. 7. 2021.).

Jones, Paul M. (31. svibnja 2020.). Action Domain Responder. original-date: 2014-05-06T15:01:09Z. URL: https://github.com/pmjones/adr/blob/ad009df5bf066dbd97317718c9b1af4c093f1e/ADR.md (pogledano 24. 7. 2021.).

— (18. srpnja 2021.a). Model View Controller and "Model 2". original-date: 2014-05-06T15:01:09Z. URL: https://github.com/pmjones/adr/blob/ad009df5bf066dbd97317718c9b1af4c093f1e/MVC-MODEL-2.md (pogledano 25. 7. 2021.).

— (2021.b). Paul M. Jones | Action Domain Responder. URL: http://pmjones.io/adr/ (pogledano 30. 6. 2021.).

Kouraklis, John (2016.). MVVM in Delphi: architecting and building model view ViewModel applications. For professionals by professionals. New York, NY: Apress, Springer Science+Business Media. 143 str. ISBN: 978-1-4842-2214-0 978-1-4842-2213-3.

Krawczyk, Jan (2021.). A brief history of PHP. URL: https://ifj.edu.pl/private/krawczyk/php/intro-history.html#:~:text=PHP%20was%20conceived%20sometime%20in,the%20Personal%20Home%20Page%20Tools. (pogledano 23. 7. 2021.).

Lagorio-Chafkin, Christine (2018.). We are the nerds: the birth and tumultuous life of Reddit, the internet's cult First Edition. New York: Hachette Books. 492 str. ISBN: 978-0-316-43537-6 978-1-4789-4745-5.

Lee, Dami (2. kolovoza 2018.). The popular Musical.ly app has been rebranded as TikTok. The Verge. URL: https://www.theverge.com/2018/8/2/17644260/musically-rebrand-tiktok-bytedance-douyin (pogledano 21. 7. 2021.).

Lerdorf, Rasmus (11. ožujka 2010.). php.internals: PHP 6. URL: https://news-web.php.net/php.internals/47120 (pogledano 24. 7. 2021.).

Mailtrap (2021.a). Automated Email Testing with API | Mailtrap. URL: https://mailtrap.io/automated-email-testing/ (pogledano 7. 12. 2021.).

— (2021.b). Fake SMTP Server for Testing by Mailtrap. URL: https://mailtrap.io/fake-smtp-server/ (pogledano 7. 12. 2021.).

— (2021.c). HTML & CSS Email Checker by Mailtrap. URL: https://mailtrap.io/html-email-checker/ (pogledano 7. 12. 2021.).

— (2021.d). Pricing | Mailtrap. URL: https://mailtrap.io/pricing/ (pogledano 7. 12. 2021.).

— (2021.e). QA Automation | Mailtrap. URL: https://mailtrap.io/qa-automation/ (pogledano 7. 12. 2021.).

Malloy, Judy (2016.). Social media archeology and poetics. OCLC: 956990555. ISBN: 978-0-262-33687-1 978-0-262-33686-4 978-0-262-33688-8. URL: http://site.ebrary.com/id/11249447 (pogledano 23. 7. 2021.).

McNamee, Roger (2019.). Zucked: the education of an unlikely activist. New York: Penguin Press. 1 str. ISBN: 978-0-525-56136-1.

MDN (2021.). SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | M URL: https://developer.mozilla.org/en-US/docs/Glossary/SPA (pogledano 27. 7. 2021.).

Model, Mitchell (26. prosinca 2014.). Model View Controller History. URL: http://wiki.c2.com/?ModelViewControllerHistory (pogledano 19. 7. 2021.).

Music Business Worldwide (30. travnja 2018.). More music is played on YouTube than on Spotify, Apple Music Music Business Worldwide. URL: https://www.musicbusinessworldwide.com/more-music-is-played-on-youtube-than-on-spotify-apple-music-and-every-audio-streaming-platform-combined/ (pogledano 26. 7. 2021.).

Netsplit (2021.). IRC Networks - Top 10 in the annual comparison. URL: https://netsplit.de/networks/top10.php?year=2004 (pogledano 26. 7. 2021.).

Pardes, Arielle (2. travnja 2018.). „The Inside Story of Reddit's Redesign". Wired. Section: tags. ISSN: 1059-1028. URL: https://www.wired.com/story/reddit-redesign/ (pogledano 26. 7. 2021.).

Patreon (2021.). What is Patreon? Patreon Help Center. URL: https://support.patreon.com/hc/en-us/articles/204606315-What-is-Patreon- (pogledano 8. 12. 2021.).
PeerTube (6. srpnja 2021.). Wikipedia. Page Version ID: 1032308248. URL: https://en.wikipedia.org/w/index.php?title=PeerTube&oldid=1032308248 (pogledano 27. 7. 2021.).
Phalcon Tim (12. travnja 2017.). Benchmarking Phalcon. Phalcon Blog. URL: https://blog.phalcon.io (pogledano 20. 2. 2021.).
— (19. kolovoza 2020.). The Future of Phalcon. Phalcon Blog. URL: https://blog.phalcon.io (pogledano 23. 6. 2021.).
— (2021.a). High Performance PHP Framework - Phalcon Framework. Phalcon Website. URL: https://phalcon.io (pogledano 24. 7. 2021.).
— (2021.b). Phalcon Documentation - Micro Application. URL: https://docs.phalcon.io (pogledano 22. 11. 2021.).
PHP Grupa (2021.a). PHP 7: New features - Manual. URL: https://www.php.net/manual/en/migration70.new-features.php (pogledano 24. 7. 2021.).
— (2021.b). PHP: History of PHP - Manual. URL: https://www.php.net/manual/en/history.php.php (pogledano 13. 2. 2021.).
— (2021.c). PHP: PHP 5 ChangeLog. URL: https://www.php.net/ChangeLog-5.php#PHP_5_0 (pogledano 24. 7. 2021.).
— (2021.d). PHP: PHP 8.0.0 Release Announcement. URL: https://www.php.net/releases/8.0/en.php (pogledano 23. 7. 2021.).
Popović, Elvis i Roman Tomašković (6. lipnja 2016.). „PHP 7.0 Pregled noviteta, testiranje i tranzicija". CASE. Sv. 28. CASE d.o.o., str. 10. URL: http://download.case.hr/Zbornici/Zbornik_CASE28_final.pdf.
Prokofyeva, Natalya i Victoria Boltunova (2017.). „Analysis and Practical Application of PHP Frameworks in Development of Web Information Systems". Procedia Computer Science 104, str. 51–56. ISSN: 18770509. DOI: 10.1016/j.procs.2017.01.059. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050917300601 (pogledano 23. 7. 2021.).
RxJS (11. prosinca 2021.). RxJS - Introduction. URL: https://rxjs.dev/guide/overview (pogledano 2. 12. 2021.).
Skilton, Mark (2015.). Building a digital enterprise: a guide to constructing monetization models using digital te OCLC: 922951304. ISBN: 978-1-137-47772-9. URL: http://www.books24x7.com/marc.asp?bookid=89345 (pogledano 23. 7. 2021.).
Skólski, Paweł (30. ožujka 2018.). Single-page application vs. multiple-page application. Medium. URL: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58 (pogledano 27. 7. 2021.).
Späth, Peter (2021.). Beginning Java MVC 1.0: Model View Controller Development to Build Web, Cloud, and Berkeley, CA: Apress. ISBN: 978-1-4842-6279-5 978-1-4842-6280-1. DOI: 10.1007/978-1-4842-6280-1. URL: http://link.springer.com/10.1007/978-1-4842-6280-1 (pogledano 25. 7. 2021.).

Statista (2021.). Most used social media 2021. Statista. URL: https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ (pogledano 1. 6. 2021.).

Stenberg, Daniel (2021.). History of IRC (Internet Relay Chat). URL: https://daniel.haxx.se/irchistory.html (pogledano 26. 7. 2021.).

TikTok (11. prosinca 2021.). About TikTok | TikTok. URL: https://www.tiktok.com/about?lang=en (pogledano 11. 12. 2021.).

Trygve, Reenskaug (22. ožujka 1979.). Trygve/MVC. URL: https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html (pogledano 19. 7. 2021.).

Vance, Lucas (20. travnja 2011.). php - What is the HMVC pattern? Stack Overflow. URL: https://stackoverflow.com/a/5736164 (pogledano 25. 7. 2021.).

Vimeo (19. srpnja 2021.). Wikipedia. Page Version ID: 1034293044. URL: https://en.wikipedia.org/w/index.php?title=Vimeo&oldid=1034293044 (pogledano 27. 7. 2021.).

Wang, Mengmeng i dr. (2020.). „TIKTOK'S RISE TO GLOBAL MARKETS 1".

Welch, Chris (19. svibnja 2013.). YouTube users now upload 100 hours of video every minute. The Verge. URL: https://www.theverge.com/2013/5/19/4345514/youtube-users-upload-100-hours-video-every-minute (pogledano 26. 7. 2021.).

Zephir Language Team (2021.a). Zephir Documentation. URL: https://docs.zephir-lang.com/0.12/en/motivation (pogledano 30. 6. 2021.).

— (2021.b). Zephir Website. Zephir Website. URL: https://zephir-lang.com/en (pogledano 24. 7. 2021.).

Zmievski, Andrei (22. travnja 2011.). „The Good, the Bad, and the Ugly: What Happened to Unicode and PHP 6". URL: https://www.slideshare.net/andreizm/the-good-the-bad-and-the-ugly-what-happened-to-unicode-and-php-6 (pogledano 24. 7. 2021.).

Popis slika

1. Popularnost društvenih mreža po broju korisnika 1

2. Glavna stranica repozitorija na GitLab-u 7
3. GitLab-ovo sučelje za pregled kontinuiranje integracije 8
4. Kontrolna ploča XAMPP programa 9
5. Mailtrap sandučić 10

6. Dijagram ADR uzorka 15
7. Broj zahtjeva koje okvir u prosjeku obrađuje u sekundi 18
8. Broj zahtjeva koje okvir u prosjeku obrađuje u sekundi 18
9. Prosječno potrebno vrijeme za obradu jednoga zahtjeva 19
10. Prosječno potrebno vrijeme za obradu jednoga zahtjeva 19
11. Broj uključenih datoteka kroz obradu zahtjeva 20
12. Broj uključenih datoteka kroz obradu zahtjeva 20
13. Prosječna uporaba radne memorije 21
14. Prosječna uporaba radne memorije 21
15. Laravel-ov ekosustav 24
16. Opisni dijagram MVC uzorka 29

17. Jedan od terminala za Community Memory 39
18. Sjednica u Talkomatic sustavu 41
19. Emotikoni podržani u sustavu PLATO 42
20. Početni prikaz igre prilikom spajanja 43
21. Uvodna poruka CWRU BBS sustava 44
22. Primjer BBS-a sa kreativnijim i vizualno naprednijim prikazom 44
23. Staromodni *textmode* klijent za IRC 45

24. Grafički klijent za IRC . . . 45
25. Osnivač i izumitelj Facebook-a, Mark Zuckerberg . . . 48
26. Grafičko sučelje Facebook-a . . . 48
27. Novo korisničko sučelje Reddit-a . . . 50

28. Podešavanje Patreon stranice . . . 62
29. Postavke Patreon razina . . . 63
30. Razine podrške . . . 63
31. Pretpregled Patreon stranice . . . 64

32. Arhitektura sustava implementirane društvene mreže . . . 69
33. ERA dijagram društvene mreže . . . 71
34. Dijagram korištenja društvene mreže . . . 74
35. Dijagram slijeda uspješne prijave . . . 75
36. Dijagram slijeda neuspješne prijave . . . 76
37. Dijagram slijeda višefaktorske prijave . . . 76
38. Dijagram slijeda objavljivanja . . . 77
39. Dijagram slijeda naplate . . . 79
40. Struktura kôda za poslužiteljsku stranu aplikacije . . . 80
41. Životni tijek API poziva . . . 82
42. Dijagram klasa kontrolera . . . 83
43. Dijagram klasa entiteta . . . 84
44. Dijagram klasa modela . . . 85
45. Dijagram repozitorijskih klasa . . . 85
46. Dijagram klasa resursa . . . 86
47. Dijagram klasa transformera . . . 87
48. Struktura kôda klijentske aplikacije . . . 88
49. Struktura i zavisnosti Core-layout modula . . . 89
50. Struktura i zavisnosti Public modula . . . 89
51. Struktura i zavisnosti App modula . . . 90
52. Struktura i zavisnosti Billing modula . . . 90
53. Struktura i zavisnosti Chat modula . . . 90
54. Struktura i zavisnosti Profile modula . . . 91

55. Struktura i zavisnosti Feed modula 91
56. Struktura i zavisnosti Groups modula 91
57. Struktura i zavisnosti Search modula 92
58. Struktura i zavisnosti Shared modula 93
59. Definirane rute klijentske aplikacije 94
60. Prikaz obrasca prijave 102
61. Prikaz registracijskog obrasca 109
62. Verifikacijska e-mail poruka 114
63. Uspješna verifikacija korisnika 114
64. Prikaz objava 116
65. Obrazac za stvaranje nove objave 117
66. Objave sa različitim vrstama sadržaja 122
67. Postavke korisničkog profila 123
68. PayPal stranica za naplatu 126
69. Uspješna naplata 127

Popis tablica

1. Usporedba PHP razvojnih okvira 23

2. Popis metoda monetizacije 60

Popis isječaka kôda

2.1. CI/CD konfiguracijska datoteka za GitLab 8

3.1. Primjer komponente 33
3.2. Uporaba izraza kao proslijeđene vrijednosti 33
3.3. Primjer servisa 34
3.4. Injektiranje zavisnosti servisa 34
3.5. Primjer direktive 35
3.6. Navođenje direktive 35
3.7. Primjer: Pretpostavljeni tip podatka 36
3.8. Primjer: Zadani tip podatka 36
3.9. Primjer: Više mogućih tipova podataka 37
3.10. Primjer: Tip podatka sa neobaveznom vrijednošću 37
3.11. Primjer: Sučelja kao tipovi podataka 37
3.12. Primjer: Klase kao tip podataka 37

7.1. Primjer podataka kakve poslužitelj vraća klijentu 86
7.2. core-layout.component.html 89
7.3. Navigacija AppModule-a 95
7.4. Navigacija CoreLayoutModule-a 96
7.5. Konfiguracijska datoteka za varijable okoline u Angularu 99
7.6. Konfiguracijska datoteka za varijable okoline u PhalconPHP 99
7.7. Konfiguracijski objekt PhalconPHP okvira 100
7.8. Učitavanje varijable okoline prilikom pripreme servisa 101
7.9. HTML kôd stranice za prijavu 103
7.10. Podatkovno vezanje obrasca 103

7.11. Podatkovno vezanje polja obrasca . . . 103
7.12. Priprema modela obrasca prijave . . . 104
7.13. Osluškivanje događaja klika gumba . . . 104
7.14. Obrada prijave . . . 104
7.15. Metoda poziva REST API-ja za prijavu . . . 105
7.16. Kontrolerska metoda za obradu prijave . . . 105
7.17. Servis za obradu prijave . . . 106
7.18. Validacija korisničkih podataka . . . 106
7.19. Provjera korisničkih podataka . . . 107
7.20. Povratak resursa . . . 107
7.21. Korisnički resurs . . . 107
7.22. Transformer korisničkog resursa . . . 108
7.23. Dohvat prijavljenog korisnika i preusmjeravanje na profilnu stranicu . . . 108
7.24. HTML kôd registracijske stranice . . . 110
7.25. Inicijalizacija obrasca registracije . . . 110
7.26. Slanje podataka o registraciji . . . 110
7.27. Registracijska metoda kontrolera za autentikaciju . . . 111
7.28. Registracijski servis . . . 111
7.29. Korisnički repozitorij - registracija . . . 112
7.30. Zapisivanje registriranog korisnika i verifikacije s popratnim slanjem poruke elektroničke pošte . . . 112
7.31. Servis slanja e-mail pošte . . . 113
7.32. Dinamičko popunjavanje sadržaja e-maila . . . 113
7.33. HTML struktura verifikacijske stranice . . . 114
7.34. Obrada verifikacije na klijentskoj strani . . . 115
7.35. Obrada verifikacije na poslužiteljskoj strani . . . 115
7.36. Priprema modela obrasca za novu objavu . . . 117
7.37. HTML kôd polja za učitavanje datoteka . . . 118
7.38. Provjera tipa datoteke . . . 118
7.39. Obrada poziva za stvaranje nove objave . . . 119
7.40. Servis za kreiranje nove objave . . . 119
7.41. Kreiranje objave s validiranim podacima . . . 120

7.42. Validacija podataka o objavi 120
7.43. Validacija veličine datoteke 121
7.44. Obrada datoteka 121
7.45. Završavanje transakcije i vraćanje objave 122
7.46. Obrada novostvorene objave na klijentskoj aplikaciji 122
7.47. Kontrolerska metoda za stvaranje naplate 124
7.48. Servis za stvaranje zahtjeva naplate 124
7.49. Parametri za PayPal naplatu 125
7.50. Obrada zahtjeva 127
7.51. Zapisivanje zahtjeva u bazu podataka 127