Razvoj informacijskog sustava za provođenje sportskog mentoriranja

Legović, Boris

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli
Odjel za informacijsko-komunikacijske tehnologije

BORIS LEGOVIĆ

RAZVOJ INFORMACIJSKOG SUSTAVA ZA PROVOĐENJE SPORTSKOG MENTORIRANJA

Diplomski rad

Pula,_____,___godine.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli
Odjel za informacijsko-komunikacijske tehnologije

BORIS LEGOVIĆ

RAZVOJ INFORMACIJSKOG SUSTAVA ZA PROVOĐENJE SPORTSKOG MENTORIRANJA

Diplomski rad

JMBAG: 0303022489**,** redoviti student
Studijski smjer: Informatika
Predmet: Izrada informatičkih projekata
Znanstveno područje: Informatika
Mentor: doc.dr.sc. Siniša Sovilj
Sumentor: dipl.ing.rač Nikola Tanković

Pula,_____,____godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Boris Legović, kandidat za magistra Informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

Puli,__,_____godine

IZJAVA
o korištenju autorskog djela

Ja, Boris Legović, dajem odobrenje Sveučilištu Jurja Dobrile u Puli Fakultetu informatike, kao nositelju prava iskorištavanja, da moj diplomski rad pod nazivom Izrada edukativnog alata koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli Fakultet te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.
Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli,_____(datum)

Potpis

_____

SVEUČILIŠTE JURJA DOBRILE U PULI
FAKULTET INFORMATIKE U PULI
ODJEL ZA INFORMACIJSKO-KOMUNIKACIJSKE TEHNOLOGIJE

Pula, 1. ožujka 2017.

Diplomski zadatak

Pristupnik:	Boris Legović (0303022489)
Studij:	Sveučilišni diplomski studij Informatike
Naslov (hrv.):	Razvoj informacijskog sustava za provođenje sportskog mentoriranja
Naslov (eng.):	Development of information system for sport mentoring conduction
Opis zadatka:	Zadatak je izraditi informacijski sustav koji spaja klijente i sportske trenere/mentore tako da se omogući trenerima provođenje svojih usluga poslovanja na globalnoj razini uz pomoć informatičkog sustava.

Mentor:

_____

doc.dr.sc Siniša Sovilj

SADRŽAJ

UVOD 1
1. Planiranje sustava 2
1.1 Modeli razvoja 2
1.2 Faze razvoja informacijskih sustava 3
1.3 Modeli razvoja – evolucijski model 5
2. Tehnologije u razvoju sustava 6
2.1 PHP 6
2.2 JavaScript 7
2.3 JQuery 8
2.4 Vue.JS 9
2.5 Laravel 11
2.6 XAMPP 13
2.6 Pusher 13
3. Razvoj sustava za sportsko mentoriranje 15
3.1 Opis aplikacije - use-case dijagram 15
3.2 Okruženje za razvoj 17
3.3 Konfiguracija Laravel okvira 17
3.4 Registracija i validacija korisnika 19
3.5 Vizualizacija grafova 23
3.6 Pohrana podataka na server 26
3.7 Registracija računa korisnika 28
3.8 Plaćanje i smjer transakcija 29
3.9 Integracija komunikacijskih kanala 32
3.10 Postavke komunikacijskih kanala 32
3.11 Postavke privatnih kanala 33
3.12 Postavke vanjskih diskova za pohranu podataka 36
3.13 Postavke diskova unutar Laravel okvira 40
4. Sustav za sportsko mentoriranje 43
4.1 Opis osnovne funkcionalnosti sustava 43
4.2 Registracija korisnika 44
4.3 Prijava korisnika 45
4.4 Vizualizacija napretka korisnika 45
4.5 Osobne vježbe – kreiranje, korištenje i vizualizacija 47
4.6 Dijeljenje rutina za vježbanje 48
4.7 Mentoriranje 51
4.8 Korisničke opcije i ostale funkcionalnosti 54

ZAKLJUČAK 56
Literatura 57
Sažetak 60
Abstract 61

UVOD

Naslov diplomskog zadatka “Razvoj informacijskog sustava za provođenje sportskog mentoriranja“ obuhvaća širok pojam ponude usluga. Sam naslov sastoji se od pojmova koji obuhvaćaju određene funkcije pa tako sama riječ “informacijski sustav“ asocira na procesuiranje informacija, spremanje i obradu istih. ”Sportsko mentoriranje” u širokom smislu može biti primjenjivo u jednom ili nekoliko tipova sporta. U ovome radu sustav će obuhvaćati nekoliko specifičnih grana sporta. Projekt ima svrhu modernizirati sportsko mentoriranje, omogućiti korisnicima kvalitetno odrađivanje svih usluga koje ono uključuje s težištem na globalizaciju. Konvencionalna definicija globalizacije podrazumijeva proces otvaranja i liberalizacije nacionalnih financijskih tržišta i njihova stapanja u globalno tržište kapitala. No danas se češće pod tim pojmom podrazumijeva "međunarodna integracija" dobara, tehnologija, rada i kapitala pa se može govoriti o globalizaciji u širem smislu (DUJŠIN, 1999.). Diplomski rad je podijeljen u 4 ključna dijela. Prvi dio fokusira se na planiranju sustava. Opisuju se poslovni modeli i njihova primjena. U drugom dijelu prikazan je kratak opis tehnologija koju su korištene u radu i njihove osnove prednosti i mane. Treći dio prikazuje način na koji se razvija sustav putem prethodno opisanih tehnologija. Posljednje poglavlje prikazuje finalni proizvod i opis rada i funkcionalnosti za krajnje korisnike.

1. Planiranje sustava

U ovome poglavlju analizirat će se temeljni pojmovi vezani uz informatički sustav i gradnju. Fokus će biti usmjeren na arhitekturu sustava i planiranje. Sustav ne uključuje samo pisanje koda već i hardverske mogućnosti i ljudske resurse. Iz toga bi definicija sustava po Pavliću (2011:14) odredila sustav kao: "skup povezanih dijelova (softver, hardver, ljudi, procedure, informacije te komunikacijske mreže) kojima je cilj pribaviti i prenijeti informacije i podatke za funkcioniranje, planiranje, odlučivanje i/ili upravljanje poslovnom organizacijom." Iz ovoga se da zaključiti da je za kreaciju potrebno isplanirati i povezati sve navedene pojmove u smislenu cjelinu.

1.1 Modeli razvoja

Proces razvijanja informacijskog sustava sastoji se od mnogih aktivnosti koje se grupiraju u klase sličnih aktivnosti pod nazivom faze. Postoje 3 osnovne faze: projektiranje, izgradnja i održavanje informacijskog sustava. Pod projektiranjem (analizom, oblikovanjem, dizajnom) podrazumijevaju se sve misaone aktivnosti potrebne da se pristupi fizičkoj izgradnji (implementaciji) sustava. Cilj projekcije sustava je izgraditi modele (nacrte, zamisli, skice, vizije) na osnovi kojih će se razviti sustav. Pod izgradnjom podrazumijeva se izvođenje aktivnosti u poslovnom sustavu za organiziranje baza podataka, izrada programskog proizvoda i organizacija ljudskih resursa vezanih uz informacijski sustav. Održavanje podrazumijeva izvođenje aktivnosti promjena u informacijskom sustavu kako bi on tijekom vremena dobio promjenjive potrebe korisnika. Informacijski sustav ažurira se tako da se sukcesivno izvode aktivnosti i primjenjuju metode za izradu modela. U ovome će se poglavlju obraditi modeli razvoja. Model razvoja prijedlog je semantike nizanja razvojnih faza i njihovih veza. Osim prikaza oblika slaganja faza u metodologijama razvoja informacijskih sustava, modeli razvoja dobru su i za definiranje faza softverskog inženjerstva. Softversko inženjerstvo je inženjerska disciplina čiji je cilj naći metodologiju proizvodnje softvera (Pavlić, 2011.).

1.2 Faze razvoja informacijskih sustava

Prema Pavliću (Pavlić, 2011), opće faze životnog ciklusa svakog proizvoda su:

1. Razvoj proizvoda
   - Istraživanje (metodološka istraživanja, istraživanje primjene, planiranje marketinga)
   - Razvoj (razvoj prototipa, ispitivanje prototipa nulte serije, planiranje marketinga)
2. Sazrijevanje proizvoda
   - Rana primjena (ispitivanje proizvoda kod prvih korisnika, marketing)
   - Rana masovna proizvodnja (pojava više istih proizvoda, porast potražnje, poboljšanje kvalitete proizvoda i usluga)
3. Zrelost proizvoda
   - Masovna proizvodnja (pad cijena, pad potraživanja, pojavljuju se ”klonovi”).

Prema Olleu (Stahonja, 1992.), faze životnog ciklusa razvoja informacijskog sustava jesu :

1. Strategijske studije,
2. Planiranje informacijskih sustava,
3. Analiza poslovanja,
4. Oblikovanje sustava,
5. Izradba (konstruiranje),
6. Razvojno ispitivanje,
7. Postavljanje,
8. Ispitivanje postavljenog sustava,
9. Rad sustava,
10. Proširivanje i održavanje,
11. Povlačenje iz upotrebe
12. Post mortem

Istraživanja u predočenju znanja, na području projektiranja informacijskih sustava, semantičkih modela podataka, programskih specifikacija, baza podataka i

programskih jezika, prema paradigmi "inteligentnih" sustava, teže je izraditi metode koje izravno i prirodno odgovaraju ljudskom ”poimanju” stvarnosti, a iz kojih će se moći na postojećem IT-u implementirati informacijski sustav. Osobni je problem naći dovoljno precizan sustav zapisivanja relevantnoga znanja. Pri crtanju različitih vrsta dijagrama primjenjuju se temeljni principi organizacije znanja kao: klasifikacija, agregacija, generalizacija i hijerarhija (Brody,1984.).

Ne postoji zajedničko stajalište o glavnim neriješenim problemima u prezentaciji znanja ni u npr. užem području objektno orijentiranoga projektiranja jezicima UML i OML, ali je prisutno spajanje različitih pristupa (Coad,1991.). Različite metodologije rješavaju problem rasporeda faza životnog ciklusa razvoja informacijskog sustava uglavnom na originalan način. Svi pristupi oblikovanju metodologije imaju faze životnog ciklusa, ali ističu neki od vlastitih značajki: faze, aktivnosti u fazama, metode, alati, brzina, ugrađeno mjerenje, poboljšanja i prilagodbeno korištenje gotovim komponentama, objektna paradigma i dr.

U osnovi faze životnog ciklusa prema kaskadnome modelu jesu projektiranje, izvedba i održavanje (Cippico, 1996.) ili analiza, dizajn i implementacija (Peters, 1988.) s preklapanjem aktivnosti pojedine faze i evaluacijom sustava tijekom faze održavanja. Detaljnija raščlamba životnog ciklusa u faze (Varga, 1996.), (Simon, 2001.), (Kovačić, 1994.) izgleda ovako:

1. Strategija – odgovara na pitanje zašto
2. Analiza – što
3. Dizajn – kako
4. Gradnja, tj. programiranje – izraditi sustav,
5. Testiranje – ocjena kvalitete
6. Uvođenje – prva primjena s provjerom
7. Održavanje

Prema prototipu modela, koji se vrlo rano počeo primjenjivati, a dobro je proučen i prikazan u nizu radova, u osnovi se dodaje jedna faza ”prototipiranja” koja vrlo rano daje ”osjećaj” kakav će sustav biti kada bude izgrađen (Peters, 1998.).

Yourdon je životni ciklus metodom DTP prikazao kao mrežu povezanih procesa koji se koriste nizom ulaza i kreiraju niz izlaznih međurezultata te upozorio na: mogućnost konzervativnog pristupa prema kojemu se procesi izvršavaju jedan po jedan kaskadno, mogućnost "radikalnog" pristupa u kojem se od prvoga dana započinje sa

svim procesima paralelno (npr. programiranje započinje kad i analiza zahtijeva) i mogućnošću niza srednjih putova (Yourdon 1989.). Nasuprot tom pristupu moguće je kao pretpostavku uzeti "dekompoziciju procesa pakiranja". Prirodno je odgovoriti na pitanje što treba raditi najprije, a što poslije te je hipoteza da životni ciklus treba složiti prema redoslijedu izvođenja aktivnosti jednoga podsustava postala *de facto* pretpostavka većine metodologija (Inmon, 1986.), (Hawryszkiewycz, 1988.), (Barker, 1989.).

Modeli razvoja su opće ideje o mogućim putovima razvoja informacijskog sustava. Metodologije su intelektualni proizvodi koji uključuju ideje iz više različitih modela i sadržavaju vlastite ideje o redoslijedu modeliranja te imaju detaljno razrađen i opisan postupak modeliranja informacijskog sustava.

1.3 Modeli razvoja – evolucijski model

Postoje sljedeći modeli razvoja: kaskadno (vodopadni), pseudostrukturni, strukturni, V/model, prototipski model, RAD (*Rapid Application Development*), *Agile* metode i dr. U ovom projektu će se koristiti Evolucijski model.
Evolucijski (inkrementalni, iterativni) model sadržava ideju kontinuiranog rasta i razvoja. Informacijski sustav nikada nije završen i on se stalno mijenja. Izgradnjom nekog djela informacijskog sustava korisnici tijekom rada počinju uočavati moguća poboljšanja sustava i daju prijedloge izmjena. Primjena aplikacije mijenja pogled korisnika. Potrebe se mijenjaju, a paralelno i rastu. U prvim mjesecima uporabe, broj zahtjeva je maksimalan, a zatim ostaje konstantan tijekom uporabe, sve do velikih reorganizacija sustava. Informacijski sustav raste rastom organizacijskog sustava. Ideja dekompozicije uključena je u evolucijski model i to tako da se sustav dijeli u niz manjih cjelina gdje se svaka stavka pojedinačno može dizajnirati, izgraditi i uvesti u sustav. Takve se cjeline jedna po jedna grade i uvode, a informacijski se sustav širi i postupno raste. Sustav se izrađuje u malim inkrementalnim koracima. Evolucijski je pristup paralelnih sljedova aktivnosti, takvih da svaki pojedini slijed vodi prema proizvodu koji se isporučuje korisniku. Proizvod nakon uvođenja generira daljnje zahtjeve i pojavljuje se novi slijed aktivnosti (Pavlić, 2011).

2. Tehnologije u razvoju sustava

Prilikom izrade projekta bilo je potrebno koristiti više različitih razvojnih okvira, servisa i programskih jezika. Većina jezika i radnih okvira ima slično obilježje, stoga korištenje različitih tehnologija ne predstavlja izazov, već brzo rješenje. U sljedećim cjelinama detaljnije će biti opisane tehnologije, njihova obilježja i razlozi korištenja unutar projekta.

2.1 PHP

Svaka operacija izvršena na webu uključuje klijent računalo i server. Klijent je jedan uređaj (primjerice, jedan preglednik) koji izvršava jedan zahtjev na jedan udaljeni poslužitelj. Poslužitelj putem jednog jezičnog skriptiranja interpretira zahtjev klijenta i šalje odgovor (npr. HTML stranicu, JSON objekt ili XML) klijentu. Tako klijent interpretira odgovor pa u slučaju web preglednika, generirat će HTML.
PHP je jezična skripta na strani poslužitelja, jezik koji se nalazi na udaljenom serveru, koji u fazama izvršavanja interpretira informacije primljene od klijenta pomoću web poslužitelja, izrađuje i vraća rezultat klijentu koji je formulirao zahtjev. Php obrađuje podatke putem HTTP zahtjeva i kreira odgovor koji vraća klijentu.
PHP je prvenstveno razvijen za web, što implicira da je većina njegovih značajki implementirana u funkciji toga. Primjerice, pristupanje zahtjevima HTTP tipa GET ili POST omogućeno je u nekoliko znakova koda.
HTTP definira 9 metoda za slanje zahtjeva, no najčešće su to dvije prethodno spomenute. GET je metoda kojom se zahtijeva većina informacija na web poslužitelju, te se zahtjevi prenose putem upita, tj. dio je URL-a koji sadrži parametre (npr. www.fitnessbook.org/ test?page?id=123). Metoda POST, s druge strane, omogućuje slanje podataka na poslužitelj bez prikazivanja u nizu upita.
Važna značajka prilikom čitanja/pisanja kolačića (eng. Cookie) na pregledniku je njegova podrška za sesije. Kolačići su strateški tekstovi koji se koriste za traganje informacija u vezi s web stranicom korisnika. Upotrebljavaju se za pohranu informacija koje se odnose na korisnika, ali se spremaju na poslužitelju i klijentu. Koriste se za upravljanje pristupom osobnim sadržajima registriranih korisnika.
PHP također pruža funkcije za bazu podataka (MySQL, Postgres, SQLite itd.), po interakciji s web poslužiteljem (Apache), za manipuliranje slikama, za izvršavanje

veze s udaljenim serverom (cUrl). Iako je nastao kao web jezik, može se koristiti kao jezik za skriptiranje unutar terminala.
Jedna od glavnih vrlina PHP-a je vrlo aktivna zajednica. Postoje tisuće biblioteka koje proširuju funkcionalnost baze i u većini slučajeva, sve su licence izdane s licencom Open Source (Otvoreni kod besplatan za korištenje). Brojni su i razvojni okviri nastali ovim jezikom (Zend Framework, Symfony Framework, CakePHP, Laravel), tako i CMS (WordPress, Drupal, Joomla).
PHP ima osobinu multiplatforme (crossplatform), tj. može se koristiti u okruženjima temeljenima na Unix (Linux, Mac OSX) te Windowsima. Unutar ovog projekta koristit će se Laravel za PHP razvojno okruženje.

2.2 JavaScript

JavaScript je skriptni objektno orijentiran jezik, uobičajeno korišten u programiranju Web stranica koje rade na principu klijentske strane, web aplikacijama, za pozivanje dinamičke interaktivnosti povezane funkcijama koje se vežu na direktne ulazne naredbe korisnika računala (tipkovnica, miš itd.).
JavaScript se može koristi za HTML, izradu JSP stranica ili unutar zasebne logike povezane datotekama ekstenzije “js“. Nedavno je njegovo područje korištenja prošireno na hibridne aplikacije namijenjene višeplatformskom radu kojima se izvorni kod temelji na JavaScriptu, HTML-u i CSS-u.
Razlike u ostalim jezicima, kao što je C++, koji omogućuje pisanje programa potpuno samostalno, JavaScript je korišten prije svega kao skriptni jezik, integrirani, unutar drugog programa. Funkcionalnost je zasnovana na pretpostavci da skripta radi u skladu s definiranim API-jem (*Application user interface*), što omogućuje pristup specifičnim operacijama, čija implementacija radi na opterećenju programa domaćina. Kada se skripta izvršava, upotrebljava zahtjev na API za izvršavanje pojedinih operacija, ne predviđajući da funkcije postoje u jeziku.
Slična bazna funkcionalnost postoji u jeziku Java, koji se također oslanja na funkcije iz biblioteka.
Tipičan primjer rada može se vizualizirati u web pregledniku. Isti koristi JavaScript kojim interpretira kod, zauzima memoriju i izvršava operacije.
Sučelja koje dopuštaju JavaScript povezana su preglednikom DOM. DOM je programsko sučelje na HTML dokumentu. DOM predstavlja HTML dokument (ali i

web čitač) u obliku JavaScript objekta. To znači da se pomoću JavaScripta može mijenjati svaki dio dokumenta. Web DOM je tijekom
godina evoluirao u pravcu odvajanja od JavaScripta.
DOM je nezavisni objektni model, koji se može koristiti iz svakog programskog jezika zasnovanog na objektima.
Izvan mreže, JavaScript se integrira u različite aplikacije. Adobe Acrobat i Adobe Reader podržavaju JavaScript u PDF datotekama. Preglednik Mozilla, koja je baza mnogih web preglednika, koristi JavaScript za implementaciju korisničkog sučelja i logike transakcije svojih različitih proizvoda.

2.3 JQuery

jQuery je JavaScript biblioteka otvorenog koda koja pojednostavljuje interakcije između HTML dokumenta ili, preciznije, objektnog modela dokumenta (DOM) i JavaScripta. Grubo rečeno, jQuery pojednostavljuje dinamički HTML. Konkretno, jQuery pojednostavljuje manipuliranje HTML dokumentom i kretanje po njemu, obradu događaja čitača mreže (engl. *web browser*), DOM animacije, Ajax interakcije i razvoj skripta na JavaScriptu za različite čitače.

jQuery je danas jedina dostupna biblioteka koja odgovara i dizajnerskim i programerskim zahtjevima. U tom smislu, jQuery je klasa za sebe. Prednosti koji donosi jQuery su mnogobrojne, a neke od bitnijih su sljedeće:

- Otvorenog je koda i riječ je o projektu s licencama MIT i GNU *General Public License* (GPL).
- Mala je (18 KB u minimalnom obliku) i zip kompresirana (nekompresirana je 114 KB).
- Nevjerojatno je popularna, što znači da je korisnička zajednica ogromna i da ima mnogo onih koji pružaju pomoć kao programeri i/ili instruktori.
- Usklađuje razlike između mrežnog čitača umjesto korisnika.
- Namjerno je napravljena sa svedenom osnovom, s jednostavnom, ali ipak pametno osmišljenom arhitekturom proširivanja dodatnim programskim modulima (engl. *plugins*).
- Ima veliki fond programskih modula koji se postojano širi otkad se biblioteka jQuery pojavila.

- Njen API je potpuno dokumentiran, uključujući primjere ugrađenog koda, što je u svijetu biblioteka JavaScripta rijetkost.
- Namjenski je napravljena tako da se izbjegnu konflikti s drugim JavaScript bibliotekama.
- Podrška korisničke zajednice je prilično upotrebljiva, i obuhvaća liste slanja, IRC kanale i ogromne količine uputstava, članaka i blogova.
- Razvija se otvoreno, što znači da svatko može doprinijeti ispravljanju grešaka, unaprjeđenju i razvoju.
- Razvija se postojano i konzistentno.
- Činjenica da su je prihvatile velike organizacije (na primer, Microsoft, Dell, Bank of America, Digg, CBS, Netflix) doprinijet će njenoj dugotrajnosti i stabilnosti.
- Njena elegancija, metodologije i filozofije koje mijenjaju način na koji se piše JavaScript kod, postaju standard sam za sebe.
- Njena dokumentacija sadrži mnoge elemente(npr., API čitač, aplikacije kontrolne tablice itd.) uključujući i API čitač van mreže (AIR aplikacija).
- Namjerno je prilagođena kako bi mogla služiti za programiranje nenametljivom JavaScriptu.
- Ostala je JavaScript biblioteka (a ne razvojni okvir, engl. *framework*), ali usporedno s njom postoji i srodan projekt za elemente grafičkog korisničkog sučelja i razvoj aplikacija (jQuery UI) (Lindley, 2012).

2.4 Vue.JS

Vue.js (poznat kao Vue) je *open source* JavaScript okvir za izgradnju korisničkih sučelja. Integracija u projekte koji koriste druge JavaScript biblioteke je jednostavna s Vueom jer je dizajniran da bude inkrementalno prilagodljiv. Vue također može funkcionirati kao okvir za web aplikacije sposoban za prikaz naprednih aplikacija. Vue.js je popularan JavaScript sučeljni (eng. front-end) okvir koji je izgrađen za organiziranje i pojednostavljivanje razvoja web stranica.
Projekt se usredotočuje na stvaranje ideja u razvoju web korisničkog sučelja (komponente, deklarativno korisničko sučelje, vruće učitavanje (eng. hot-reloading), uklanjanje pogrešaka u realnom vremenu i sl.).

Sadrži inkrementalno prilagodljivu arhitekturu. Jezgra biblioteke usredotočuje se na deklarativno prikazivanje i sastav komponente i može se ugraditi u postojeće stranice. Napredne značajke potrebne za složene aplikacije kao što su usmjeravanje, upravljanje i alati za gradnju nude se putem službeno održavanih biblioteka i paketa. Vue je stvorio Evan You nakon što je radio za Google pomoću AngularJS-a. Vue koristi sintaksu predloška temeljenog na HTML-u koji omogućuje da deklarirano veže DOM s podacima iz Vueovih primjeraka. Svi Vue predlošci su važeći HTML koji se može analizirati pomoću preglednika i HTML *parser-a*. U pozadini, Vue sastavlja predloške u virtualne DOM funkcije *renderiranja*. U kombinaciji s reaktivnim sustavom, Vue može izračunati minimalni broj komponenata kako bi ponovno prikazao i primijenio minimalnu količinu manipulacija DOM-a kada se promijeni stanje aplikacije.

Vue omogućuje korištenje sintakse predloška ili odabir izravnog pisanja funkcija *renderiranja* pomoću JSX (React – JavaScript biblioteka namijenjena razvoju sučelja). *Render* funkcije otvaraju mogućnosti za snažne komponente na temelju uzoraka. Jedna od Vueovih najrazličitijih značajki je neupadljiv sustav reaktivnosti. Modeli su samo obični JavaScript predmeti. Kada ih se izmijeni, prikaz se ažurira. Vue pruža optimizirano ponovno prikazivanje bez potrebe za ponovnim programiranjem. Svaka komponenta prati svoje reaktivne zavisnosti tijekom njenog prikazivanja tako da sustav zna točno kada treba ponovo prikazati i koje će komponente ponovno prikazati.
Komponente su jedna od najmoćnijih značajki Vuea. U velikoj aplikaciji potrebno je podijeliti cijelu aplikaciju u male, samostalne i često ponovljive komponente kako bi se razvoj mogao kontrolirati. Komponente proširuju osnovne HTML elemente za inkapsuliranje ponovljivog koda. Na visokoj razini, komponente su prilagođeni elementi na koje Vueov prevoditelj pridaje ponašanje.
Vue nudi različite načine primjene efekata prijelaza kada se stavke umetnu, ažuriraju ili uklone iz DOM-a. To uključuje alate za:

- automatsko primjenjivanje klasa za CSS prijelaze i animacije
- integrirane biblioteke animacija CSS-a treće strane (eng. “third party“), kao što je Animate.css
- upotrebljavanje JavaScripta za izravno manipuliranje DOM-om
- integriranje biblioteka animacije JavaScripta treće strane, kao što je Velocity.js.

Vue će automatski provjeravati je li ciljni element primijenjen CSS prijelaz ili animacija. Ako se to dogodi, CSS prijelazne klase bit će dodane/uklonjene u odgovarajućim vremenskim razmacima.

JavaScript biblioteke poput Vue pružaju jednostavno sučelje za promjenu onoga što se prikazuje na stranici na temelju trenutnog URL-a - bez obzira na to kako je promijenjeno (bilo putem veze s e-poštom, osvježavanjem ili veze na stranici). Osim toga, korištenje usmjerivačkog sučelja omogućava namjerno prelazak puta preglednika pri određenim preglednim događajima (npr. klikovima) na gumbima ili vezama. Podržava mapiranje ugniježđenih ruta do ugniježđenih komponenata i nudi finu zrnatu tranzicijsku kontrolu. Uz Vue, programeri već sastavljaju aplikacije s malim građevinskim blokovima koji grade veće komponente. S ruterom, komponente se mogu lako mapirati na putovima kojima pripadaju. Roditeljski/korijenski putovi moraju navesti gdje bi se djeca trebala prikazati.
Vue.js biti će korišten unutar ovog projekta za *renderiranje* poruka i sustav slanja poruka, obavijesti, kao i ažuriranja istih.

2.5 Laravel

Laravel je besplatan PHP web-okvir otvorenog izvornog koda kojeg je stvorio Taylor Otwell i namijenjen je razvoju web aplikacija nakon arhitektonskog uzorka modela preglednog kontrolera (MVC) temeljenog na Symfony-u (PHP razvojni okvir). Neke od značajki Laravela su modularni sustav pakiranja s namjenskim upraviteljem ovisnosti, različitim načinima pristupanja relacijskim bazama podataka, uslužnim programima koji pomažu u primjeni i održavanju aplikacija te usmjerenosti prema čitljivosti koda. Izvorni kod Laravela je dostupan je na GitHub-u i licenciran pod uvjetima MIT licence.
Taylor Otwell je stvorio Laravel kao pokušaj da pruži napredniju alternativu okviru CodeIgniter, koji nije osigurao određene značajke kao što je ugrađena podrška za autorizaciju korisnika. Laravelovo prvo izdanje beta verzije objavljeno je 9. lipnja 2011., nakon čega slijedi Laravel 1 izdanje kasnije u istom mjesecu. Laravel 1 uključuje ugrađenu podršku za autentifikaciju, lokalizaciju, modele, poglede, sesije, usmjeravanje i ostale mehanizme, ali nije imao potporu za kontrolere koji su ga spriječili da budu pravi MVC okvir.

Laravel 2 je objavljen u rujnu 2011., donoseći razna poboljšanja od strane autora i zajednice. Glavne nove značajke uključuju podršku za kontrolere, što je Laravelu 2 donijelo potpunu kompatibilnost s MVC-om, ugrađenom podrškom za inverziju principa kontrole (IoC) i sustavom predložaka Blade. Kao nedostatak, podrška za pakete treće strane uklonjena je u Laravel-u 2.

Laravel 3 je objavljen u veljači 2012. s nizom novih značajki, uključujući sučelje naredbenog retka (CLI) zvanom Artisan, ugrađenu podršku za više sustava za upravljanje bazom podataka, migracije baza podataka kao oblik kontrole verzije za raspored baze podataka, podršku za rukovanje događaja i sustav pakiranja nazvanih Paketi.

Laravel 4, kodnog naziva Illuminate, objavljen je u svibnju 2013. godine. Izrađen je kao kompletna kopija Laravelovog okvira, prebacujući svoj raspored u skup posebnih paketa distribuiranih putem *skladatelja* (eng. “composer“), koji služi kao upravitelj paketa aplikacije. Ostale nove značajke u izdanju Laravel-a 4 uključuju sazivanje baze podataka za početnu populaciju baza podataka, podršku za redove poruka, ugrađenu podršku za slanje različitih vrsta e-pošte i podršku za odgodu brisanja podataka baze podataka zvanih ‘’meka brisanja’’ (eng. ”soft deletion”).
Laravel 5 je objavljen u veljači 2015. kao rezultat internih promjena koje su završile preuređivanjem tadašnjeg Laravel 4.3 izdanja. Nove značajke Laravel 5 izdanja uključuju podršku za raspoređivanje periodično izvršenih zadataka kroz paket nazvan Scheduler, apstrakcijski sloj nazvan Flysystem koji omogućuje daljinsko pohranjivanje na isti način kao i lokalni datotečni sustavi, poboljšano rukovanje paketima preko Elixir-a i pojednostavljeno provjeravanje autentičnosti izvana putem dodatnog Socialite paketa. Laravel 5 također je uveo novu strukturu stabla unutarnjeg direktorija za razvijene primjene. Laravel 5.1, objavljen u lipnju 2015., prvo je izdanje Laravela koje obuhvaća dugotrajnu podršku (LTS), koje podrazumijeva popravak grešaka (eng. ”bug”) u trajanju dvije godine i zakrpa na 3 godine. Laravel 5.3, objavljen je 23. kolovoza 2016. Nove značajke u 5.3 su usredotočene na poboljšanje brzine razvojnog programera dodavanjem dodatnih poboljšanja u okviru uobičajenih zadataka. Laravel će unutar ovog projekta biti korišten kao primarni razvojni okvir.

2.6 XAMPP

XAMPP je besplatan i open source multiplatformski paket za web poslužitelja kojeg su razvili Apache Friends, a sastoji se od Apache HTTP poslužitelja, MariaDB baze podataka i tumača za skripte napisane u PHP-u i Perl programskim jezicima. Ime XAMPP je kratica od Cross-Platform (X), Apache (A), MariaDB (M), PHP (P) i Perl (P). To je jednostavna, lagana Apache distribucija koja ga čini iznimno jednostavnim za razvojne programere kod stvaranja lokalnog web poslužitelja za testiranje i implementacijske svrhe. Sve što je potrebno za postavljanje web poslužitelja - aplikacija poslužitelja (Apache), baze podataka (MariaDB) i skriptnog jezika (PHP) – uključeno je u kompresiranoj datoteci. XAMPP je također cross-platform, što znači da jednako dobro funkcionira na Linuxu, Macu i Windowsu. Budući da većina stvarnih implementacija web poslužitelja koristi iste komponente kao XAMPP, prelazak s lokalnog poslužitelja za testiranje na live poslužitelj je izuzetno jednostavan.

2.6 Pusher

Najčešći poznati oblik push notifikacija jesu SMS poruke, gdje korisnici primaju obavijesti u tekstualnom obliku.
Da bi bilo moguće odašiljati poruke, poslužitelj treba znati IP klijenta. Nadalje, potrebno je otvoriti vezu s klijentom kako bi se poruka odašiljala. No, poslužitelj ne može zatražiti otvaranje veze na klijentu jer bi se time narušila privatnost korisnika. Umjesto toga, veza se drži otvorenom (duge veze). Kada preglednik otvori vezu pomoću websocket poslužitelja, poslužitelj pruža detalje putem odgovora i veza se zadržava otvorena. No, tradicionalni poslužitelj ne može zadržati mnoge otvorene veze u memoriji jer većina tradicionalnih poslužitelja koristi način thread-per-request pa bi već 100 otvorenih veza popunilo memoriju.

Za takve slučajeve koriste se neograničavajući IO web-poslužitelji. Takvi poslužitelji mogu upravljati stotinama otvorenih veza sa svojim klijentima bez konzumiranja puno resursa (CPU/memorija).

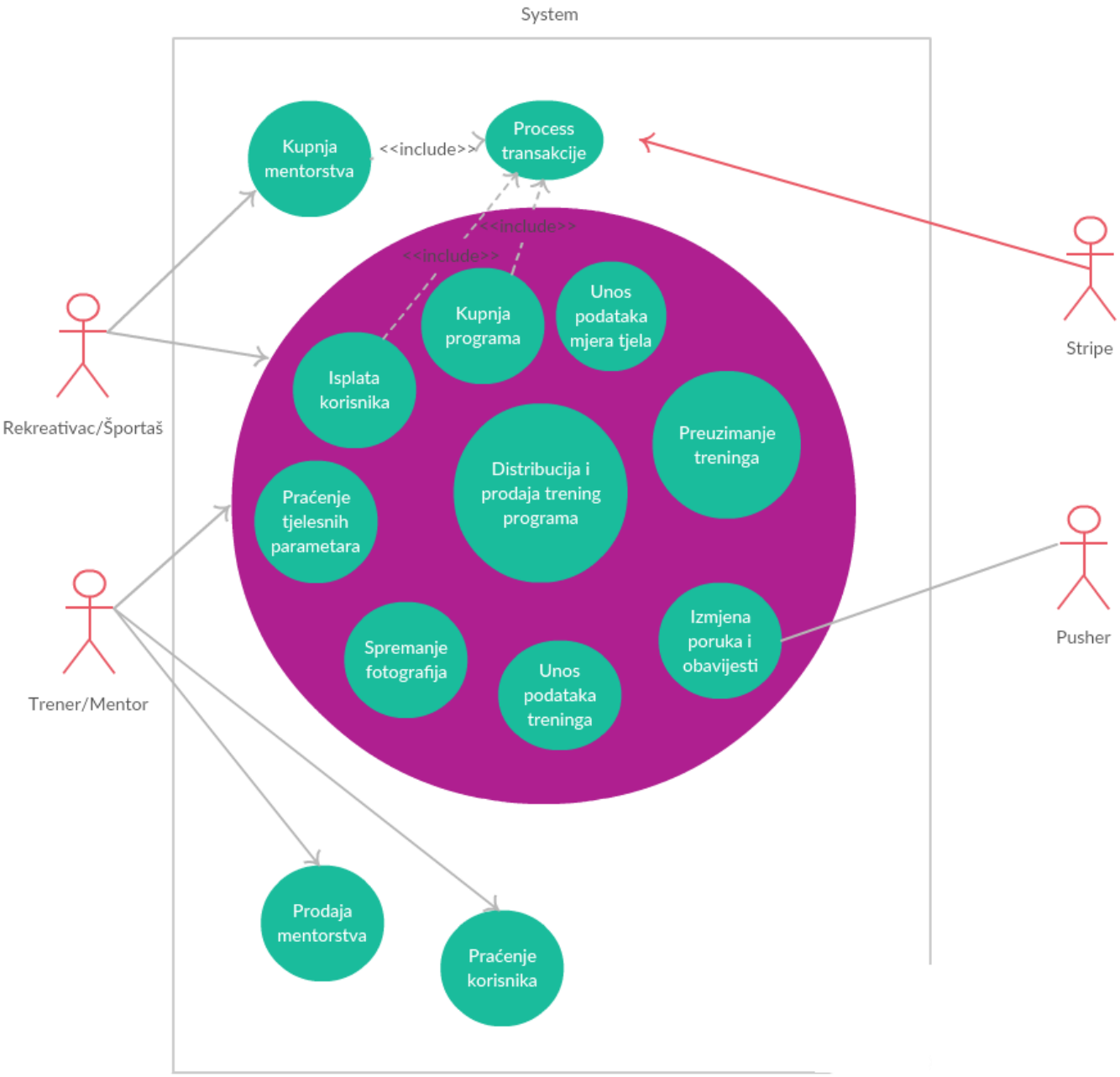
Unutar ovog projekta koristi se Pusher biblioteka kao sloj u stvarnom vremenu između poslužitelja i klijenata. Pusher održava stalne veze s klijentima - preko WebSocketa i koristi HTTP vezu - tako da čim poslužitelj ima nove podatke koje želi prikazati klijentima, to može učiniti odmah preko Pushera.
Pusher nudi bibliotekama integraciju u sve korištenije jezike i okvire kao PHP, Ruby, Python, Java, .NET, Go i Node, Vue.js, iOS-u i Javi (Android).

3. Razvoj sustava za sportsko mentoriranje

Treća cjelina opisivat će izradu samog sustava i njegove funkcionalnosti. Cilj izrade projekta je kreacija skalabilnog sustava, lako razumljivog te implementacija arhitekture koja omogućuje daljnju nadogradnju. Moderan razvoj softvera bazira se na SCRUM metodologiji. Budući da je tržište vrlo promjenjivo s obzirom na utjecaj tehnologije, konkurencije i ostalih parametara koje utječu na kvalitetu i korisnost softvera, vrlo je bitno omogućiti konstantne promjene/iteracije, kako bi se u kratkome vremenu mogli adaptirati na želje korisnika. Ideja realizacije sustava proizlazi iz želje da se stvori sustav koji je jednostavan, gotovo minimalističan za korisnika, kako bi obuhvatio veću dobnu populaciju.

3.1 Opis aplikacije - use-case dijagram

Sustav je namijenjen korisnicima koji se dijele u dvije skupine - sportaši ili rekreativci i mentori ili treneri. Način na koji pristupaju sustavu je potpuno isti, a razlikuju ih dodatne funkcije koje mentor posjeduje. Pristup sustavu omogućen je putem 3 vrste platformi – Android, iOS i web. Razlika u pristupu je jednaka, no mobilni uređaji omogućuju spremanje podataka na uređaj te se u trenutku povezivanja s mrežom podaci šalju na server.



Slika 1. Use-case diagram sustava, izvor: izradio autor

3.2 Okruženje za razvoj

Nakon definiranja dijagrama i osnovnih funkcionalnosti sustava, potrebno je pripremiti okolinu za razvoj. Razvoj se odvija na lokalnoj mašini te se završni materijal postavlja na server. U praksi se uvijek radi na lokalnim mašinama te nakon testiranja softvera, verzija se odlaže na git repozitorij. Prije odlaganja softvera korisnicima, potrebno je provest Unit testove. To su skripte koje testiraju funkcionalnost sustava nakon izmjena u kodu.
Prvi korak uključuje instalaciju XAMPP paketa koji omogućuje podizanje lokalnog servera i interpretaciju PHP koda. Nakon uspješne instalacije, potrebno je preuzeti okvir Laravel. Preporučena metoda je preko Composera. Kada su dvije najvažnije stavke uspješno postavljene na lokalnoj mašini, slijedi konfiguracija.

3.3 Konfiguracija Laravel okvira

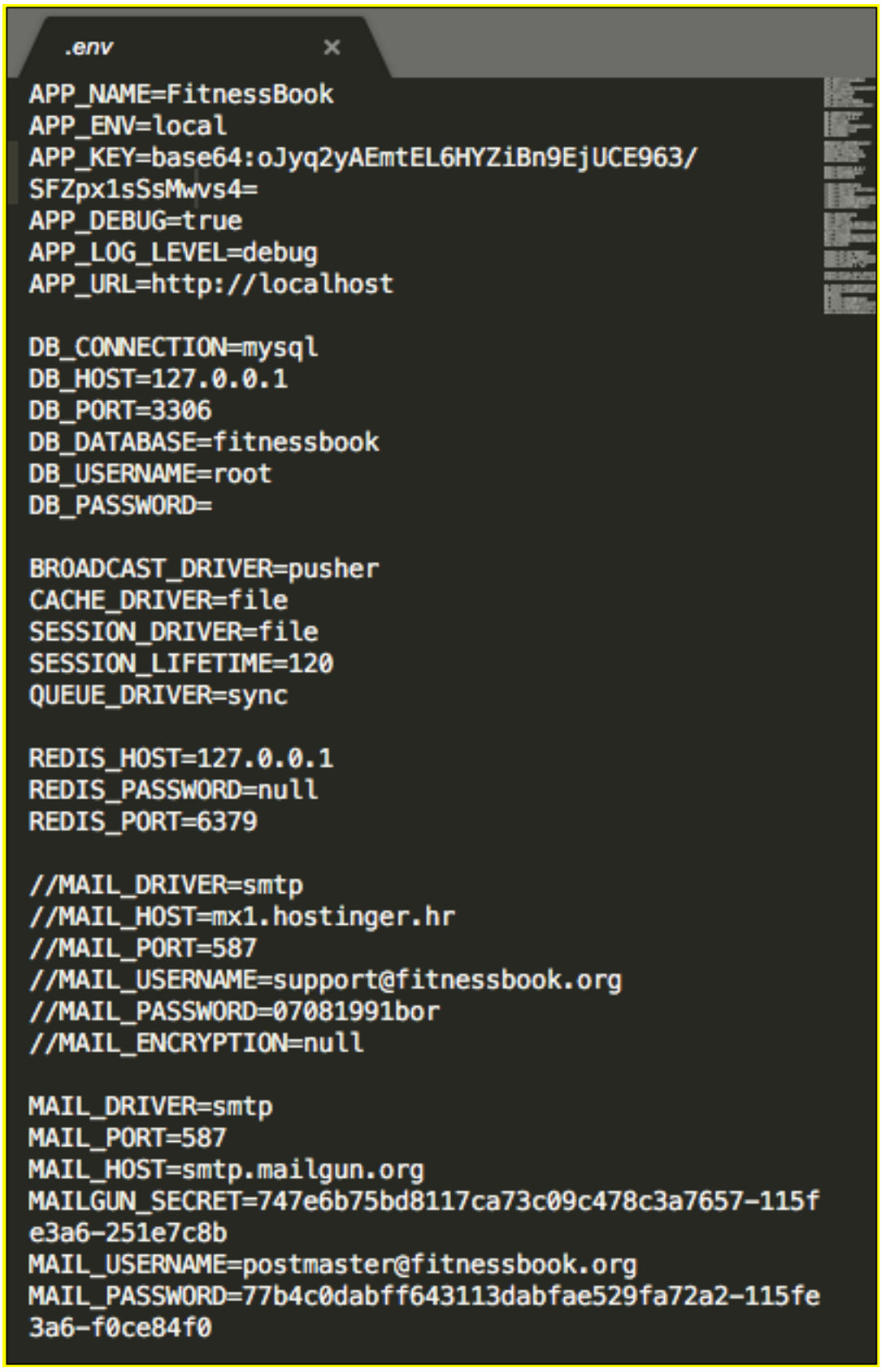
Laravel koristi globalne varijable za konfiguraciju projekta koje se nalaze u ”env.” datoteci. Unutar datoteke se konfiguriraju postojeće, ali i vlastite varijable koje se mogu pozvati unutar cijelog projekta.

```
PUSHER_APP_ID="485816"
PUSHER_APP_KEY="18942df6af39a769e942"
PUSHER_APP_SECRET="24fdsa34fw4323b912"
PUSHER_CLUSTER = "eu"

STRIPE_KEY=pk_test_D54mC7BNvjNgps8wHzDAO7A2
STRIPE_SECRET=sk_test_TpyHJ2L9w0Ya21GkcpAoZibV

DO_SPACES_KEY=SNVOPFLWFZ43RVV4D7TZ
DO_SPACES_SECRET=423wefdsfcisdoufisd2342
DO_SPACES_REGION=ams3
DO_SPACES_BUCKET=fitnessbookspace
DO_SPACES_ENDPOINT=https://ams3.digitaloceanspaces.com
```

Slika 2. Primjer Env konfiguracijske datoteke, izvor: izradio autor



Slika 3. Primjer Env konfiguracijske datoteke, izvor: izradio autor

Kod početne konfiguracije bitno je uspostaviti vezu s lokalnom bazom. Budući da se u projektu radi sa XAMPP bazom podataka koja po osnovnim konfiguracijama koristi korisnika ”root” bez lozinke, dovoljno je konfigurirati lokalnu adresu i ime baze. Parametar ”APP_URL” označava početnu adresu i ”APP_NAME” kao ime web stranice. Ostali parametri su proizvoljni, kao npr. ”SESSION_LIFETIME” – trajanje sesije prijave korisnika, Mail driveri i Kanali za komunikaciju koji su definirani radi drugih funkcionalnosti koje će biti opisane kasnije.

3.4 Registracija i validacija korisnika

Laravel 5.5 ima funkcije za kreiranje funkcija i pogleda za prijavu i registraciju. Nakon izvršavanja funkcije “php artisan make:auth” Laravel će kreirati potpunu konfiguraciju i poglede za spomenute funkcije.
Konfiguracijska datoteka autentifikacije nalazi se unutar datoteke config/auth.php, koja sadrži nekoliko dokumentiranih funkcija za provjeru i autentifikaciju.
Laravelove autentifikacijske jedinice čine jezgru "čuvara" (eng. ”guards”) i "pružatelja" (eng. ”providers”). *Čuvari* određuju na koji način se odvija autentifikacija korisnika, a rade na principu sesije i kolačića.
Pružatelji usluga određuju na koji način korisnici koriste podatke iz baze podataka. Laravel pruža podršku za pronalaženje korisnika pomoću ”Eloquent-a” i graditelja upita baze podataka (eng. ”Database query builder”). Laravel također omogućuje modifikaciju navedenih datoteka.
Registracija korisnika unutar izrade projekta bit će izmijenjena kako bi bila specifična za projekt. Unutar registracije potrebno je dodati nove varijable, slanje verifikacijske poštanske adrese, spremanje verifikacijskog koda na bazu podataka i ostalih potrebnih parametara za izvedbu.
Prvi korak kod modifikacije registracije uključuje izmjenu pogleda. Laravel poglede sprema unutar datoteke ”resources/views”. Unutar pogleda ”Register.blade.php” dodani su dodatni podaci za opis korisnika. Gumb za registraciju pokreće POST zahtjev i prenosi informacije unutar kontrolera.

```
     *
     * @param  array  $data
     * @return \Illuminate\Contracts\Validation\Validator
     */
    protected function validator(array $data)
    {
        return Validator::make($data, [
            'name' => 'required|string|max:255',
            'lastname' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:6|confirmed',
            'type' => 'required',
            'gender' => 'required',
        ]);
    }

    /**
     * Create a new user instance after a valid registration.
     *
     * @param  array  $data
     * @return \App\User
     */
    protected function create(array $data)
    {
        Session::flash('status','Registered! but verify your email to activate your account!');

        $user = User::create([
            'name' => $data['name'],
            'lastname' => $data['lastname'],
            'email' => $data['email'],
            'password' => bcrypt($data['password']),
            'type' => $data['type'],
            'gender' => $data['gender'],
             'status' => 0,
            'verify_token' =>Str::random(40),
        ]);
        $thisUser = User::findOrFail($user->id);

        $coaching = "No";

        if($data['type'] == 'trainer')
        {
            $coaching = "Yes";
        }
```

Slika 4. Primjer kontrolera za registraciju korisnika, izvor: izradio autor

```php
public function sendEmail($thisUser)
{
    $userMail = $thisUser->email;
    $data = ['email'=>$thisUser->email, 'token'=>$thisUser->verify_token];

    Mail::send('email.backup_email', $data, function ($m) use ($userMail)
    {
        $m->from('Fitnessbook-support@no-reply.org', 'Fitnessbook-support@no-reply.org');
        $m->to($userMail)->subject('Account verification');
    });
}

public function sendEmailDone($email,$verifyToken)
{
    $user = User::where(['email'=>$email,'verify_token'=>$verifyToken])->first();

    if($user)
    {
        user::where(['email'=>$email,'verify_token'=>$verifyToken])->update(['status'=>'1','verify_token'=>
            NULL]);
        return redirect('/login')->with('verification',true);
    }
    else
    {
          return redirect('/login');
    }
}
```

Slika 5. Primjer slanja poštanske adrese registriranom korisniku, izvor: izradio autor

Nakon uspješne registracije, korisniku se šalje elektronička pošta koja sadrži verifikacijsku šifru nastalu pri registraciji, a nalazi se unutar baze podataka. Unutar funkcije ”sendEmail” koristi se funkcija ”Mail::send()” koja pripada Laravelovom API-u. Ono što korisnik projekta mora podesiti jesu upravljački programi (eng. ”driver”) za servise elektroničke pošte (SMTP, Mailgun, Mandrill, Amazon SES, PHP). Funkcija slanja elektroničke pošte prima za prvi parametar cijeli pogled i polja s podacima. Slanje cijelog pogleda korisniku omogućuje izradu izgleda pomoću HTML-a. Nakon registracije, korisnika se usmjerava na pogled za prijavu. Unutar baze podataka u tablici ”user” nalazi se stupac ”status”. On služi kao verifikacija za registraciju korisnika. Kada korisnik zaprimi poštu, klikom na poveznicu preusmjerava ga se na kontroler koji poziva funkciju ”sendEmailDone” unutar koje se mijenja status verifikacije adrese i preusmjerava korisnika na pogled za prijavu. U slučaju da korisnik nije verificirao račun, pojavit će se poruka koja upućuje korisnika na provjeru poštanske adrese. Provjera je postignuta korištenjem JavaScripta i asinkrone funkcije ajax, putem koje se informacije prikupljene putem DOM elementa šalju putem POST zahtjeva kontrolera. Funkcija unutar kontrolera vraća rezultat provjere varijable na bazi ”status” koja je prethodno bila spomenuta. U slučaju da je varijabla potvrdna, unutar JavaScripta se poziva POST zahtjev forme.

```
<script type="text/javascript">

function checkIfUserConfirmedAcc()
{

   $("form_login").submit(function(e){
    e.preventDefault();
    });

    var email = document.getElementById('email').value;

    $.ajaxSetup({
              headers:
              {
                'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
              }
            });

       $.ajax({
                type:'POST',
                url:'/checkIfUserHaveStatusVerified',
                data:{email:email},
                success:function(data)
                {

                    if(data == '0')
                    {
                        alert('Please go to Your email and confirm account!')
                    }
                    else
                    {
                        $("#form_login").submit();
                    }
                 }
            });
   }

</script>
```

Slika 6. Primjer slanja poštanske adrese registriranom korisniku, izvor: izradio autor

Kontroler ”Login.php” odgovoran je za provjeru autentičnosti korisnika, odnosno elektroničke adrese i lozinke. U slučaju pogrešnog unosa, vraća se rezultat s porukom upozorenja, dok u suprotnom, korisnika se prosljeđuje na web sustav.

3.5 Vizualizacija grafova

Unutar projekta korišteni su grafovi za vizualizaciju podataka korisnika koji se temelje na JavaScript bibliotekama. JavaScript grafovi nude dinamični prikaz te je njihova primjena vrlo proširena u web razvoju. Također, koriste se i na mobilnim uređajima, stoga govorimo o multiplatformskim bibliotekama. Osim široke zajednice, prednost korištenja grafova jest besplatna licenca za komercijalnu uporabu.
Podaci za grafove se dohvaćaju asinkrono putem ajax POST upita na ”DatabaseQuerryController” koji zatim šalje upit na bazu i vraća podatke. Spomenuti kontroler sadrži većinu funkcija koje se odnose na dohvaćanje podataka s baze.

```
function loadMeassurementData(Request $req)
{

    $id = Auth::user()->id;
     $userMeassureTypeValue = DB::table('user_options')->select('weight_unit','size_unit')
     ->where('user_id',$id)->get();

     $arrayOfDataGraphs = array();
     $exercizeName = ['Chest','Right_biceps','Right_forearm', 'Waist','Right_glut','Right_calf','
       Shoulders','Left_biceps','Left_forearm','Hip', 'Left_glut','Left_calf','Neck','Weight','Height
       ];

     for($i = 0; $i<15;$i++)
     {
         $data = DB::table('meassurement_data') ->select('date_of_meassure', 'size_cm_or_kg')
         ->where('user_id', '=', $id)
         ->where('exercize_name_or_weight', '=', $exercizeName[$i])
         ->orderBy('date_of_meassure', 'asc') ->get();
          $encodeData = $data->toArray();
         array_push($arrayOfDataGraphs,$encodeData);

     }

         $data = DB::table('body_rank')
         ->where('user_id', '=', $id)->get();

             $result = [0,0];

           if(count($data)>0)
           {
             $result = [$data[0]->body_fat,$data[0]->lean_mass];
           }

        $res = [$arrayOfDataGraphs,$result,$userMeassureTypeValue[0]];

        return  $res;
}
```

Slika 7. Dohvat podataka o mjerama tijela korisnika, izvor: izradio autor

Slika 7. prikazuje kreiranje polja kao parametra za JavaScript funkcije kreiranja grafova. Petlja prolazi kroz 15 polja koja označavaju dijelove tijela. Za svaki pojedini dio tijela dohvaća se polje podataka, koje se lijepi u polje sa svim podacima. Tu se odvija izračun podataka za ostala dva grafa i sve se zajedno vraća kao rezultat unutar polja.

Na sljedećoj, slici broj 8. nalazi se prikaz korištenja funkcije za kreiranje grafa pomoću biblioteke ”Charts.js”. Parametri unutar funkcije pokazuju koliko je zapravo mogućnosti pri izradi i personalizaciji grafa. Parametar ”type” označava tip grafa i pruža korisniku mogućnosti korištenja nekoliko desetaka tipova grafova. Parametar ”labels” označava podatke na x-osi opisne vrijednosti. U slučaju izrade ovog projekta radi se o datumu tipa *String*. ”Dataset” parametar prima polja objekata u kojem se definiraju podaci za pojedini graf. U slučaju na slici prikazane su modifikacije samo na jednom grafu. Unutar objekta definira se veličina, debljina, oblik, boja i način prikaza boje na grafu. Valja napomenuti kako objekt može sadržavati i više proizvoljnih parametara za modifikaciju, a u primjeru su prikazani samo oni koji se koriste za izradu projekta. U većem dijelu za izradu grafova koristi se spomenuta biblioteka, no kod izrade projekta korištene su dvije različite biblioteke za prikaz. Radi se o spomenutom ”Charts.js-u” i također besplatnim bibliotekama u vlasništvu Google-a, Google Charts. Iako biblioteke Google Charts-a imaju puno manje mogućnosti za osobnu modifikaciju, sadrže neke funkcionalnosti koje Charts.js ne posjeduje, kao npr. grupiranje grafova, koji unutar projekta služe za prikaz vježbanja korisnika.

```
var myChart = new Chart(ctx,
{
type: graphType,
data: {
    labels: arrayForData[m][0],
    datasets: [{
        label: exercizeName[m],
        borderColor: gradientStroke,
        pointBorderColor: gradientStroke,
        pointBackgroundColor: gradientStroke,
        pointHoverBackgroundColor: gradientStroke,
        pointHoverBorderColor: gradientStroke,
        pointBorderWidth: 10,
        pointHoverRadius: 10,
        pointHoverBorderWidth: 1,
        pointRadius: 1,
        fill: true,
        backgroundColor: gradientStroke,
        borderWidth: 4,
        data: arrayForData[m][1]
    }]
},
options: {
    legend: {
        position: "bottom"
    },
    scales: {
        yAxes: [{
            ticks: {
                fontColor: "rgba(0,0,0,0.5)",
                fontStyle: "bold",
                beginAtZero: true,
                maxTicksLimit: 5,
                padding: 20
            },
            gridLines: {
                drawTicks: false,
                display: false
            }
}],
        xAxes: [{
            gridLines: {
                zeroLineColor: "transparent"
},
            ticks: {
                padding: 20,
```

Slika 8. Korištenje ”Chart.js” biblioteke za prikaz grafova, izvor: izradio autor

3.6 Pohrana podataka na server

Server pohranjuje podatke na dva načina: u bazu podataka i kao datoteke na server. Ovo poglavlje bavit će se isključivo spremanjem datoteka. Unutar projekta omogućena je pohrana slika i proizvoljnih datoteka. Korisnici mogu pohraniti svoje fotografije i datoteke koje predstavljaju programe vježbanja. Pohrana je omogućena u više pogleda kod kojih svaka ima svoj tip verifikacije datoteke. Slika može imati maksimalno 9 mb dok datoteka smije imati najviše 20 mb. Unutar pogleda za pohranu programa za vježbanje koriste se *slušači* jQuery biblioteke. Funkcije unutar biblioteke osluškuju promjene na DOM objektu te se nakon promjene pozivaju funkcije. Na taj način omogućeno je odstupanje od korištenja klasičnih načina pohrane podataka HTML funkcije forme. Ono što zapravo izvode ove funkcije jest da omogućuju da se otvori lokalni prikaz podataka koji se nalazi na računalu ili mobitelu kada korisnik klikne na fotografiju. Također, funkcije omogućuju da se provjera podataka odvija unutar istog pogleda, pa u slučaju da korisnik pokuša pohraniti na mjesto slike datoteku, javit će se prozor upozorenja. Ako svi podaci na pogledu za pohranu podataka prođu JavaScript validaciju, poziva se funkcija za kreiranje forme kod pritiska na gumb za pohranu. Kroz formu se šalje POST zahtjev na ”UploadTrainingController” koji se brine za validaciju veličine podataka i pohranu na server. U slučaju da veličina datoteka za pohranu nije zadovoljila veličinu kod validacije, funkcija vraća pogled s porukom obavijesti. Za spremanje podataka u bazu koristi se Eloquent model.

```
$("#trainingImageId").click(function()
{
    $("#fileImageId").click();
});

$("#fileImageId").change(function(e)
{
    /// $('#imageNameId').html($("#fileImageId").val());
    // readURL(this);
});

$("#fileTrainingButtonId").click(function()
{
    $("#fileTrainingId").click();
});

$("#fileTrainingId").change(function(e)
{
    $('#upladTrainingTextId').html($("#fileTrainingId").val());
});
```

Slika 9. Slušanje DOM elemenata, izvor: izradio autor

```
public function setDataToDB(Request $request)
{

    $file = $request->file('file');
    $image = $request->file('image');
    $title =  $request->title;
    $description = $request->description;
    $price = $request->price;
    $currency = $request->curencyValue;
    $owner_or_uploader_id =  Auth::user()->id;

    if(filesize($file)>20971520)
    {
        return view('upload_training')->with('fileToBig',true);;
    }
    else if(filesize($image)>10485760)
    {
        return view('upload_training')->with('pictureToBig',true);
    }
    else
    {

        $imagePath = Storage::disk('do_spaces')->putFile('public/training_images',$
            image,'public');
        $filePath = Storage::disk('do_spaces')->putFile('public/training_files',$file,'
            public');

        $training = new CommercialTraining;
        $training->user_id = $owner_or_uploader_id;
        $training->description = $description;
        $training->title = $title;
        $training->price = $price;
        $training->currency = $currency;
        $training->file = $filePath;
        $training->image = $imagePath;
        $training->save();

         return view('upload_training')->with('sucess',true);
    }
}
```

Slika 10. Funkcija za pohranu podataka na server , izvor: izradio autor

3.7 Registracija računa korisnika

Za novčane transakcije koristi se Stripe servis. Usluga Stripe-a omogućuje da se provode transakcije na odgovornost samoga servisa i pritom omogućuje *developerima* da modificiraju funkcije plaćanja, tj. nude dinamične usluge naplate. Ono što je bitno za ovaj projekt jest da Stripe dopušta protok transakcija na više stranaka. Budući da se jedan dio plaćanja provodi putem modela B2B (eng. ”*Bussines to bussines*”). Stripe je uveo Stripe Connect tip usluge koji pruža snažan API putem kojeg omogućuje provjeru i plaćanje prodavačima, izvođačima, davateljima usluga i još mnoge druge mogućnosti.
Prvi korak kod integracije Stripe funkcija jest registracija platforme. Unutar API sučelja korisnik upisuje sve važne podatke vezano za aplikaciju koju koristi, odnosno web stranicu. Postavke koje se unose u ovome koraku koriste se kod preusmjeravanja korisnika s vlastite platforme na Stripe servis. Iz toga razloga potrebno je upotpuniti parametre, poput adrese na koju se korisnik vraća nakon korištenja Stripe-a i logo. Nakon uspješne registracije platforme, sljedeći korak uključuje kreiranje Connect računa. Kako bi korisnici mogli primati uplate na svoj račun, potrebno je da registriraju stripe račun, odnosno da ispune Stripe formu. Gumb za registraciju treba sadržavati ”client_id”, broj koji identificira platformu, ”response_type” – rezultat koji vraća (u ovom slučaju identifikacijski stripe kod) i doseg – označava koje mogućnosti platforma ima s računom.

```
https://connect.stripe.com/oauth/authorize?
response_type=code&client_id=ca_CIc4spDBSBgusI8DbRhiJlJ8rQnEdBs1&scope=read_write
```

Slika 11. Adresa za registraciju korisnika na Stripe platformu , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 01.05.2018

Nakon što korisnik izvršava uspješnu registraciju na platformu, Stripe šalje POST zahtjeve s lozinkom pomoću koje se dohvaća korisnikov Stripe jedinstveni ključ pomoću kojeg platforma kontrolira transakcije. Po primitku koda, potrebno je izvršiti curl POST zahtjev koji vraća objekt s podacima korisnika unutar kojeg se nalazi Stripe identifikator. Identifikator se sprema u bazu podataka za kasnija korištenja.

```
$ curl https://connect.stripe.com/oauth/token \
>    -d client_secret=sk_test_TpyHJ2L9w0Ya21GkcpAoZibV \
>    -d code="{AUTHORIZATION_CODE}" \
>    -d grant_type=authorization_code
```

Slika 12. Adresa za registraciju korisnika na Stripe platformu , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 01.05.2018

3.8 Plaćanje i smjer transakcija

Nakon uspješne registracije korisničkih Stripe računa, moguće je izvoditi transakcije primanja novčanih iznosa. U ovom poglavlju bit će opisane funkcije za izvršavanje uplate na račun korisnika platforme. Connect podržava tri pristupa za obradu troškova putem platforme. Kod korištenja Standardnih računa, preporučuje se stvaranje izravnog plaćanja preko Connect računa. Koristeći ovaj pristup, Connected račun je odgovoran za troškove naknade za Stripe, povrate sredstava i storniranja uplate.

Za neke korisnike računa tipa Express i Custom, poželjno je izrađivati izravne troškove na Connected računu. U tom slučaju, iako je povezani račun početno odgovoran za naknade Stripe-a, povrate i stornirane uplate, račun na platformi ima krajnju odgovornost pokriti sve gubitke.

Stvaranje izravnih troškova na Connected računu posebno je prikladno za platforme koje omogućuju e-trgovinu svojim korisnicima.

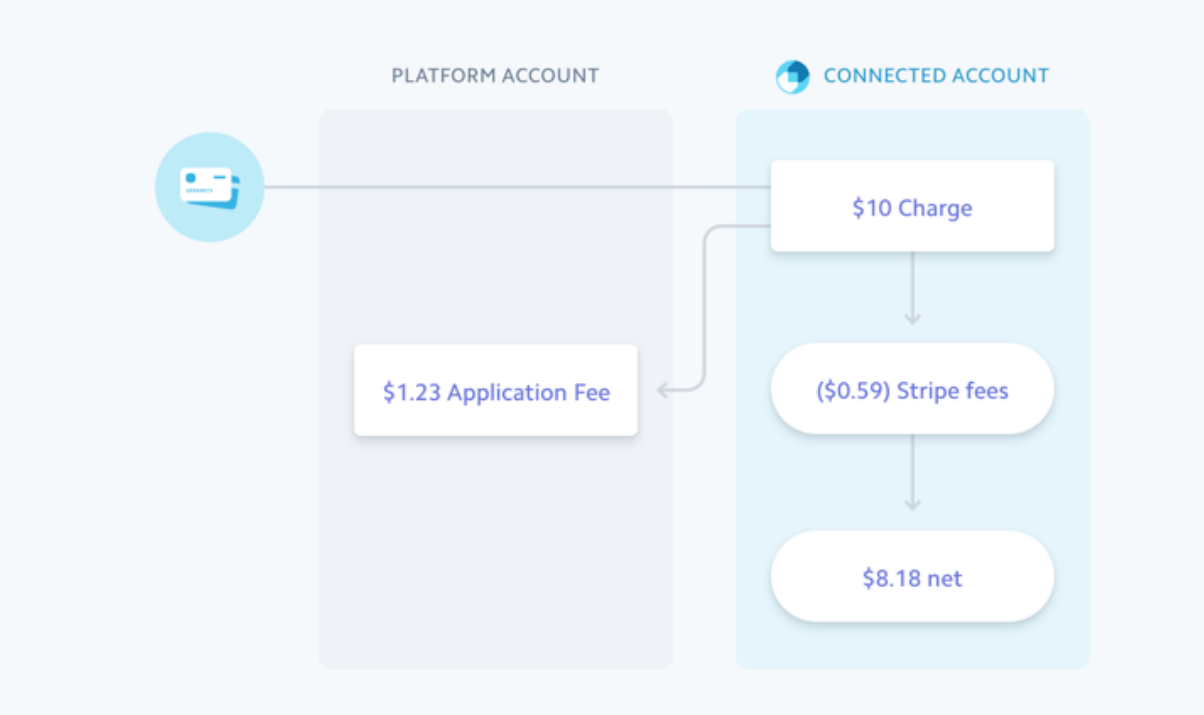
U slučaju da korisnik A kupi proizvod ili uslugu od korisnika B, novčani iznos se dijeli u 3 djela. Osoba koja prodaje (B) dobiti će iznos prodaje umanjen za 9%. To je postotak koji odlazi platformi. Nakon izdvajanja transakcija, platforma plaća troškove vezane uz transakciju (2,9 % + 30 centa za kreditne i debitne kartice, internacionalne kartice – 1%, ACH 0.8 – 5%). Za uzimanje provizije dovoljno je koristiti funkciju za direktnu naplatu i umanjiti iznos koji se šalje korisniku. Kao npr.: ako je naplata transakcije 100$, a klijentu se pošalje 80$, platforma direktno prihoduje 20$ - troškovi transakcije.

curl Ruby Python PHP Java Node Go

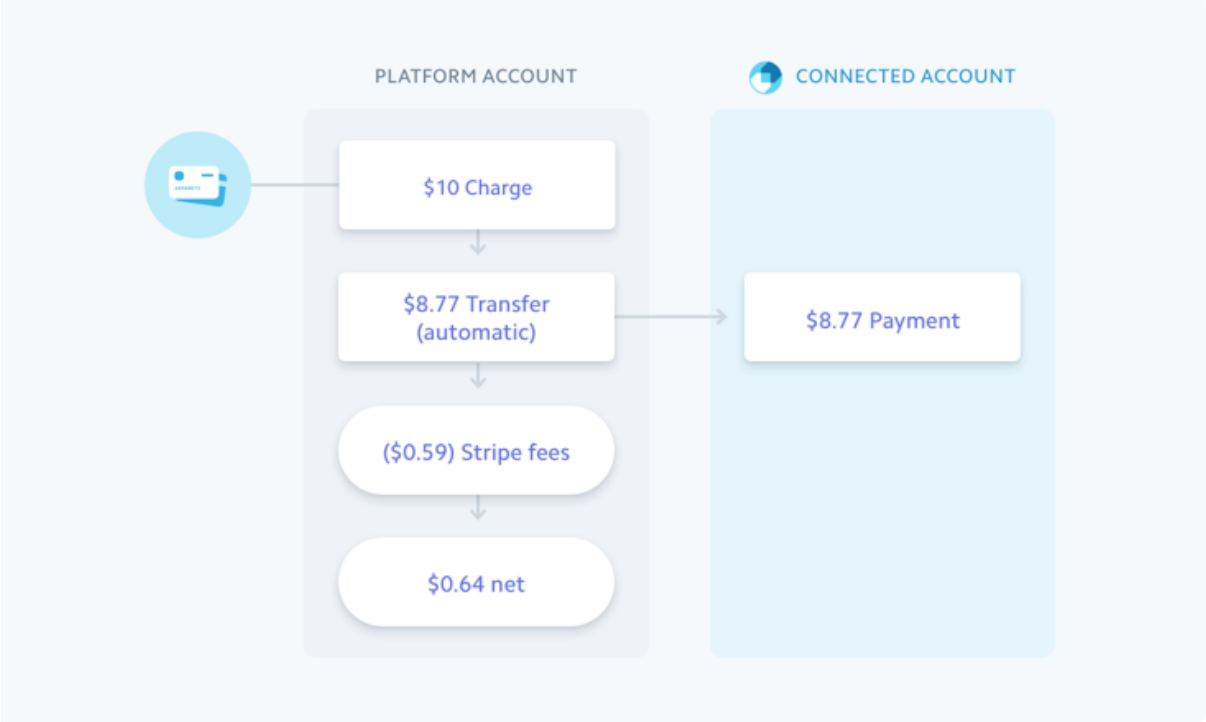
```php
// Set your secret key: remember to change this to your live secret key ir
// See your keys here: https://dashboard.stripe.com/account/apikeys
\Stripe\Stripe::setApiKey("sk_test_TpyHJ2L9w0Ya21GkcpAoZibV");

$charge = \Stripe\Charge::create(array(
  "amount" => 1000,
  "currency" => "usd",
  "source" => "tok_visa",
  "destination" => array(
    "amount" => 877,
    "account" => "{CONNECTED_STRIPE_ACCOUNT_ID}",
  ),
));
```

Slika 13. Naplata iznosa i uzimanje provizije , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno:05.05.2018



Slika 14. Tijek transakcije kod direktne naplate korisnika , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 15.05.2018



Slika 15. Tijek transakcije kod preuzimanja provizije na transakciju između dva korisnika, izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 16.05.2018

3.9 Integracija komunikacijskih kanala

U mnogim suvremenim internetskim aplikacijama, WebSocket-i se koriste za implementiranje korisničkih sučelja u stvarnom vremenu, ažuriranja i sl. Kada se neki podaci ažuriraju na poslužitelju, poruka se obično šalje preko WebSocket-a na obradu klijentu.
U sljedećem poglavlju biti će opisane integracije Push obavijesti koje se koriste za informiranje korisnika u realnome vremenu. U konkretnom slučaju koriste se za izradu konzole za dopisivanje između korisnika, ali i primanje ostalih obavijesti.

3.10 Postavke komunikacijskih kanala

Sve konfiguracije emitiranja događaja aplikacije pohranjene su u konfiguracijskoj datoteci config/broadcasting.php. Laravel podržava nekoliko mrežnih upravljačkih programa: Pusher, Redis i program loga za lokalni razvoj i ispravljanje pogrešaka. Osim toga, uključen je *null driver* koji omogućuje potpuno isključivanje emitiranja. Primjer konfiguracije uključen je za svaki od tih upravljačkih programa u konfiguracijskoj datoteci config/broadcasting.php.
Prije emitiranja bilo kakvih događaja potrebno je registrirati App\Providers\BroadcastServiceProvider. U novonastalim Laravel aplikacijama nužno je isključiti ovaj davatelj usluga u polju davatelja (eng. ”*providers*”) konfiguracijske datoteke config/app.php. Za ovaj projekt bit će korišten Pusher, koji se može instalirati putem Composera. Sljedeći korak je konfiguracija Pusher credentials-a u konfiguracijskoj datoteci config/broadcasting.php. Primjer konfiguracije već je uključen u datoteku, omogućujući brzo određivanje Pusher ključa, tajnog ključa i ID aplikacije. Konfiguracija unutar config/broadcasting.php omogućuje dodatne opcije, kao što je klaster.

```
'options' => [
    'cluster' => 'eu',
    'encrypted' => true
],
```

Slika 16. Konfiguracijska datoteka unutar Pusher-a, izvor: https://laravel.com/docs/5.6/broadcasting, pristupljeno:12.06.2018

Također potrebno je postaviti *broadcast* unutar Echo-a u resursima/assets/js/bootstrap.js datoteci:

```
import Echo from "laravel-echo"

window.Pusher = require('pusher-js');

window.Echo = new Echo({
    broadcaster: 'pusher',
    key: 'your-pusher-key'
});
```

Slika 17. Konfiguracija Echo broadcastera, izvor: https://laravel.com/docs/5.6/broadcasting, pristupljeno:14.06.2018

3.11 Postavke privatnih kanala

U primjeru projekta potrebno je definirati komunikaciju između dva korisnika, u smislu privatne komunikacije, i onemogućiti ostalim korisnicima pristup komunikaciji između dva korisnika. U projektu nije moguće voditi višesmjerne razgovore, kao ni slanje višestrukih obavijesti. Iz toga razloga, Laravel omogućuje privatne kanale koji će biti opisani u ovom poglavlju.

Budući da korisnici moraju biti ovlašteni za slušanje na privatnim kanalima, potrebno je definirati pravila za autorizaciju kanala u ruti/kanali.php. U ovom primjeru može se potvrditi da je svaki korisnik koji pokušava slušati na privatnom kanalu.1, zapravo stvaratelj narudžbe:

```php
Broadcast::channel('App.User.{id}', function ($user, $id) {
    return (int) $user->id === (int) $id;
});

//owner must listen to his chanel
Broadcast::channel('message.{userId}', function ($user) {
   // return true;
   return Auth::check();
});
```

Slika 18. Konfiguracija privatnog broadcast kanala, izvor: izradio autor

Metoda kanala prihvaća dva argumenta: naziv kanala i povratni poziv koji vraća potvrdu ili negaciju, što odgovara na pitanje: da li je korisnik ovlašten slušati na kanalu.

Svi povratni pozivi za autorizaciju primaju trenutno ovjerene korisnike kao svoj prvi argument i sve dodatne parametre zamjenskog znaka kao njihove kasnije argumente. U ovom primjeru koriste se dva tipa kanala. Jedan služi za izmjenu poruka (“message.{userId}“) a drugi za obavijesti (”App.User.{id}).

Slijedi postavljanje slušača u JavaScriptu. Prvo se upotrebljava privatna metoda za pretplatu na privatni kanal. Zatim se koristi način slušanja kako bismo slušali događaj. Na slici 17. prikazano je korištenje Laravel Echo-a zajedno sa bibliotekom Vue.js pri korištenju slušača. Funkcija ”created” poziva se pri kreaciji DOM objekta unutar kojeg se poziva identifikacijski broj skripte. Prva funkcija koja se izvodi je ”axios.post” na PHP kontroler koji pritom vraća identifikacijski broj trenutno prijavljenog korisnika. Podatak se sprema u Vue.js objekt i pokreće se ”Echo.private” funkcija koja prima parametar tipa String, koji je ujedno i privatni kanal. Unutar Echo funkcije *listen*, koja je ujedno i *callback* funkcija, postavljena su 3 uvjeta koja diferenciraju tip poruke i prema tome pozivaju različite funkcije. Prvi uvjet poziva se kada se korisnik nalazi unutar pogleda za dopisivanje i komunikaciju. Pri primitku poruke, poruka se dodaje u polje sa svim ostalim porukama i osvježuju se liste korisnika. Drugi uvjet označava obavijest korisniku u kupovini njegove usluge na platformi, te posljednji uvjet jest

slučaj gdje korisnik dobiva poruke od ostalih korisnika s kojima trenutno nije u dopisivanju.

```
created()
{
    // get logged in user id
    axios.post('/getLoggedInUserId')
      .then(response =>
      {

        app.loggedInUserId = response.data;
                  console.log('**app.loggedInUserId**',app.loggedInUserId);

         Echo.private('message.' + app.loggedInUserId)

        .listen('MessageSent', (e) =>
        {

         // here i revice the single message from
             if(app.conID == e.messageData[0].conversation_id )
             {

                 console.log('update_message ');
                 app.singleMsgs.push(e.messageData[0]);
                 this.scrollToEnd();
                 this.reorderListOfChats();
             }

             else if(app.loggedInUserId == e.messageData[0].coach_id)
             {
                 console.log('update_notifcation ',e.messageData[0].coach_id);
                 this.updateNotification(e.messageData[0].coach_id);
                 app.userBoughtCoachingData.push(e.messageData[0]);
             }
             else
             {
                  console.log('update_notification_only');
                  this.updateMessageNotification(e.messageData[0].conversation_id);
                  this.reorderListOfChats();
             }

        });
    })
    .catch(function (error)
     {
        console.log(error); // run if we have error
    });
```

Slika 19. Slušači unutar JavaScript funkcija, izvor: izradio autor

Ovakav način diferenciranja poruka postignut je tako da se kao poruka šalje objekt s informacijama. Objekt koji se šalje kreira se unutar kontrolera funkcijom ”event” koja prima parametar *broadcast* objekt sa parametrima jedinstvenog ključa korisnika i objekta koji sadržava informacije iz upita na bazu. Unutar klase ”*MessageSent*” koristi se funkcija ”*broadcastOn*” koja dohvaća kanal putem kojeg se emitira događaj.

```
$userMsg = DB::table('messages')
->join('users', 'users.id','messages.user_from')
->where('messages.id',$sendM)
->orderBy('created_at_time', 'asc')
->get();

 DB::table('conversation')
->where('id',$conID)
->update(['created_at'=>$time,'last_message'=>$msg]);

 event(new MessageSent($userToSendId,$userMsg));
```

Slika 20. Kreiranje događaja za slanje poruka, izvor: izradio autor

3.12 Postavke vanjskih diskova za pohranu podataka

Budući da neke od opcija unutar projekta sadrže mogućnosti pohrane podataka, potrebno je zakupiti prostor na serveru. Za početnu fazu izrade projekta koristio se lokalni disk za pohranu podataka, no u produkciji je potrebno je koristiti usluge poslužitelja na *web* servisu. Budući da je projekt pogonjen VPS-om koji nije namijenjen za pohranu, korišteni su vanjski diskovi poslužitelja DigitalOcean, tzv. Spaces.

DigitalOcean Spaces dizajniran je kako bi olakšao pohranu podataka na ekonomičan način. Vrlo je jednostavan te je za kreiranje potrebno nekoliko sekundi i spremno je za upotrebu bez konfiguracije. Prijenos podataka automatski se osigurava HTTPS-om, a dostupni kapacitet pohrane neprekidno se mijenja (dinamički kapacitet).

Prostori su idealni za pohranu statičnih, nestrukturiranih podataka kao što su audio, video i slike, kao i velike količine teksta. Za strukturirane ili dinamičke podatke kao što su baze podataka ili aplikacije kreirane na stranicama poslužitelja, potrebna je lokalna pohrana ili blokovska pohrana.

Blokovska usluga skladištenja pruža tradicionalni uređaj za pohranu blokova - poput tvrdog diska - preko mreže. *Cloud provideri* često imaju proizvode koji mogu pružiti uređaj za pohranu blokova bilo koje veličine i priložiti je na virtualni stroj.
Od tamo se tretira kao normalan disk. Može se formatirati s datotečnim sustavom i pohranjivati datoteke na njemu, kombinirati više uređaja u RAID niz ili konfigurirati bazu podataka za izravno pisanje na blok uređaj, izbjegavajući nadgradnju datotečnog sustava. Osim toga, mrežni uređaji za pohranu blokova često imaju neke jedinstvene prednosti pred normalnim tvrdim diskovima:

- Izrada preslike svih podataka u realnom vremenu u svrhu backup-a.
- Blok uređaji za pohranu mogu se mijenjati tako da zadovoljavaju sve veće potrebe.

Stoga, jasno je da je blokovska pohrana fleksibilno rješenje koje može biti korisno za bilo koje vrste aplikacija. Neke prednosti pohrane blokova su:

- Blok uređaji dobro su podržani. Svaki programski jezik može se lako čitati i pisati datoteke,
- Dozvole za datotečni sustav i kontrole pristupa poznate su i dobro su razumljive,
- Blok uređaji za pohranu nude nisku latenciju IO (*Input/Output*), tako da su prikladni za korištenje s bazama podataka.

Nedostaci skladišta blokova su:

- Pohrana je vezana za jedan poslužitelj odjednom,
- Blokovi i datotečni sustavi imaju ograničene *metapodatke* o informacijama koje pohranjuju (datum izrade, vlasnik, veličina). Sve dodatne informacije o onome što se sprema riješit će se na razini aplikacije i baze podataka, što je dodatna složenost za razvojnog programera,
- Potrebno je platiti blok prostor za pohranu koja je dodijeljena, čak i ako ga se ne koristi,
- Moguće je pristupiti pohrani bloka samo putem pokretnog poslužitelja.

Zahvaljujući brzim IO karakteristikama, usluge blok skladištenja pogodne su za pohranu podataka u tradicionalnim bazama podataka. Osim toga, mnoge naslijeđene aplikacije koje zahtijevaju normalnu pohranu datoteka sustava trebaju koristiti blok uređaj za pohranu.

Ako oblak server ne nudi uslugu skladištenja blokova, moguće je pokrenuti vlastitu upotrebu programa OpenStack Cinder, Ceph ili ugrađene iSCSI usluge dostupne na NAS (*Network attached storage*) uređajima.
U suvremenom svijetu *cloud computinga*, pohrana predmeta je pohrana i dohvaćanje nestrukturiranih blokova podataka i metapodataka pomoću HTTP API-ja. Umjesto da se razvrstaju datoteke na blokove za pohranu na disk pomoću datotečnog sustava, bave se cjelovitim objektima pohranjenim preko mreže. Ti objekti mogu biti slikovna datoteka, zapisnici, HTML datoteke ili bilo koji samostalni *byte*. Oni su nestrukturirani jer nema posebne sheme ili formata koje trebaju slijediti.
Budući da se API sastoji od standardnih HTTP zahtjeva, biblioteke su brzo razvijene za većinu programskih jezika. Spremanje podataka bloka postalo je lako kao HTTP zahtjev za pohranu predmeta. Preuzimanje datoteke i metapodataka je normalan zahtjev za GET. Nadalje, većina objekata za pohranu predmeta također može javno posluživati datoteke svojim korisnicima, uklanjanjem potrebe za održavanjem web poslužitelja.
Povrh toga, usluge pohrane predmeta naplaćuju se samo za prostor za pohranu koji se upotrebljava (neki se naplaćuju po HTTP zahtjevu i za propusnost prijenosa).

Pohrana objekta nije pravi izbor za svaku situaciju. Neke prednosti skladištenja objekata su:

- Jednostavan HTTP API, s klijentima dostupnim za sve poznatije operacijske sustave i programske jezike,
- Struktura troškova - plaća se samo ono što se iskoristi,
- Ugrađeno javno posluživanje statičkih sredstava znači manje poslužitelja za pokretanje,
- Neki servisi pohrane objekata nude ugrađenu integraciju CDN-a, čime se spremaju predmeti u cijelom svijetu kako bi preuzimanje i učitavanje stranica bilo brža za korisnike,
- Omogućuju preuzimanje stare verzije za povrat informacija u slučaju nezgode ili slučajno obrisanih podataka,
- Usluge pohrane objekata mogu se lako skalirati od skromnih potreba do zaista intenzivnih slučajeva upotrebe bez da razvojni programer mora pokrenuti više resursa ili restrukturirati arhitekturu za obradu opterećenja,

- Korištenje servisa za pohranjivanje objekata znači da nije potrebno održavati tvrde diskove i RAID polja jer ih obrađuje davatelj usluga,
- Biti u stanju pohraniti dijelove metapodataka uz podatke blob može dodatno pojednostaviti arhitekturu aplikacija.

Neki nedostaci pohrane objekata su:

- Ne mogu se koristiti usluge pohrane objekata za pohranu tradicionalne baze podataka zbog visoke latencije takvih usluga,
- Pohrana predmeta ne dopušta da se mijenja samo dio bloka podataka, potrebno je čitati i napisati cijeli objekt odjednom. Na primjer, na datotečnom sustavu moguće je jednostavno dodati jednu liniju na kraj datoteke. Na sustavu za pohranu predmeta potrebno je preuzeti objekt, dodati novu liniju i napisati cijeli objekt natrag. Iz toga razloga pohrana objekata nije pogodna kod dinamičnih objekata,
- Operativni sustavi ne mogu lako montirati objekt kao normalan disk. Postoje neki klijenti i prilagodnici koji pomažu u tome, ali općenito, pohrana objekata nije jednostavna funkcija kao običan prolazak kroz datoteke.

Zbog tih svojstava, pohrana predmeta korisna je kod statičkih datoteka, spremanje korisničkih sadržaja kao što su slike i filmovi, pohrana sigurnosnih kopija i spremanje zapisa.

Postoje četiri načina za izradu i upravljanje Spaces-ima.

- Upravljačka ploča DigitalOcean-a: Pomoću upravljačke ploče može se započeti s upotrebom Spaces-a izravno u web pregledniku bez ikakve konfiguracije,
- Third Party grafički alati: grafički klijenti omogućuju komunikaciju sa Spaces-om na slične načine kao na upravljačkoj ploči. Oni izbjegavaju ograničenja preglednika, ali zahtijevaju konfiguraciju za povezivanje. Third Party CLI tools: alati komandne linije kao što su s3cmd (Linux, Mac, Bash na Windowsu) olakšavaju i osobnu upotrebu i automatizaciju,
- API Spaces: API Spaces omogućuje programsko kreiranje Spaces-a i upravljanje njima. To je preporučeni način automatizacije korištenja prostora, ali zahtijeva dodatne tehničke vještine.

Moguće je koristiti jedan ili više alata istovremeno u radu sa Spaces-om.
Upravljačka ploča, kao i mnogi Third Party klijenti, predstavlja datoteke u prostoru pomoću metafora mapa.

Spremnici objekata, međutim, nisu hijerarhijski. Umjesto korištenja brancha, strukture temeljene na direktorijima, objekti su pohranjeni sustavi ključeva/vrijednosnih datoteka. To daje objektu svoju karakterističnu skalabilnost, redundanciju i pouzdanost, ali također znači da se mape mogu ponašati drugačije kada je u pitanju dohvaćanje informacija o sustavu.

Kod početka korištenja Spaces-a putem upravljačke ploče, potrebno je odabrati naziv za prostor.
Ime mora biti jedinstveno među svim korisnicima u svim regijama. Imena moraju biti mala i započeti slovom ili brojem. Mogu biti između 3 i 63 znaka i mogu sadržavati crtice.
Nakon odabira imena, odabire se regija te područje podatkovnog centra za pohranu. Regija će postati dio krajnje točke URL-a. Također postavlja se tip privatnosti – privatno ili javno. Privatnost utječe na vidljivost datoteka. U slučaju privatnog prostora, datoteke neće biti vidljive svima. To je uglavnom korisno za određene zahtjeve API-ja i aplikacije koje koriste Spaces izravno kao *backend*. Nakon stvaranja prostora, postavke se mogu promijeniti na kartici Postavke u odjeljku Dozvole.

3.13 Postavke diskova unutar Laravel okvira

Spaces ima S3 kompatibilan API, što znači da se njime mogu koristiti postojeći klijenti s Amazon S3. Laravel omogućuje jednostavno korištenje postojećeg S3 *driver-a* za spajanje na Spaces.
Prvi korak je prijava na DigitalOcean Spaces i generiranje pristupnog ključa.
Nakon toga provjerava se prisutnost *drivera* s3.

```
$ composer require league/flysystem-aws-s3-v3
```

Slika 21. Instalacija drivera za spremanje datoteka, izvor: https://laravel.com/docs/5.6/filesystem, pristupljeno: 15.06.2018

Zatim, unutar projekta potrebno je dodati novi disk koji koristi S3 *drivere*, ali također daje parametar krajnje točke u konfiguraciji diska u config/filesystems.php:

```
'spaces' => [
    'driver' => 's3',
    'key' => env('DO_SPACES_KEY'),
    'secret' => env('DO_SPACES_SECRET'),
    'endpoint' => env('DO_SPACES_ENDPOINT'),
    'region' => env('DO_SPACES_REGION'),
    'bucket' => env('DO_SPACES_BUCKET'),
],
```

Slika 22. Instalacija drivera za spremanje datoteka, izvor: https://laravel.com/docs/5.6/filesystem, pristupljeno: 16.06.2018

Također. potrebno je dodati nove varijable u .env datoteku projekta:

```
DO_SPACES_KEY=SNVOPFLWFZ43RVV4D7TZ
DO_SPACES_SECRET=NaUM3XyZrtQeDfasdfdsafYPZkapwFt9jVzdaPp698rQ0bMXbZQU
DO_SPACES_REGION=ams3
DO_SPACES_BUCKET=fitnessbookspace
DO_SPACES_ENDPOINT=https://ams3.digitaloceanspaces.com
```

Slika 23. Konfiguracija Spaces-a, izvor: Izradio autor

```
if($request->picture != null) // we just update the pic
{

        $imagePath = Storage::disk('do_spaces')->putFile('public/' . $owner_or_uploader_id ,$
          image,'public');

        // $imageName = substr($imagePath,7);
         $imageName = $imagePath;

         DB::table('user_pictures')->where('image',$request->picture)
        ->update
        (
            ['user_id' => $owner_or_uploader_id, 'image' => $imagePath]
        );

        Storage::disk('do_spaces')->delete($request->picture);
}
```

Slika 24. Instalacija drivera za spremanje datoteka, izvor: Izradio autor

Nakon što su konfigurirani svi uvjeti za Spaces, korištenje funkcije Storage::disk, s parametrom tipa string koji označava ime prostora unutar Spaces-a, datoteke se šalju na server.

4. Sustav za sportsko mentoriranje

Sustav koji se opisuje u ovome radu obuhvaća više programskih jezika, razvojnih okvira, biblioteka, razvojnih okruženja, alata i servisa. Prvenstveno naglasak je na sustavu koji se odvija na mreži, ali uključuje i usluge i pristup s mobilnih uređaja. Upravo iz tog razloga, najveći fokus je na mrežnim tehnologijama i razvojnim alatima, kao i mrežnim servisima. Jedan od bitnih stavki jest komunikacija između različitih sustava i spremanje podataka.

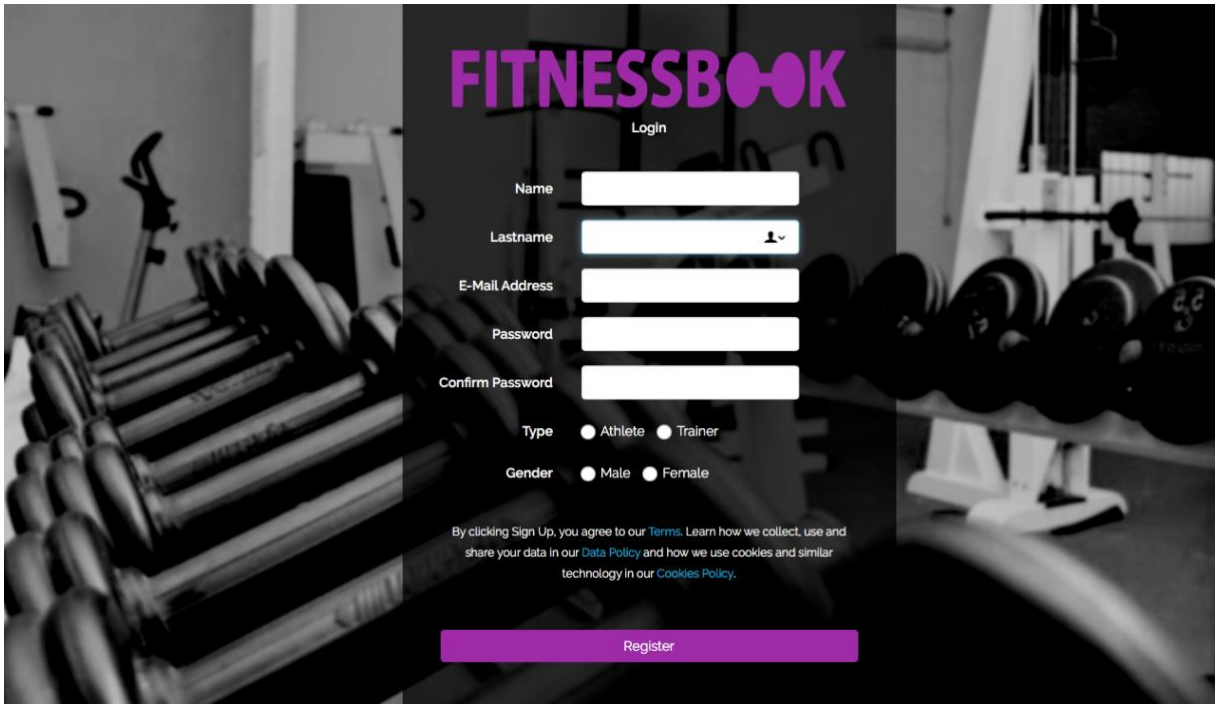
4.1 Opis osnovne funkcionalnosti sustava

Sustav obuhvaća funkciju registracije korisnika u bazu podataka. Korisnik upisuje svoje podatke i pristaje na uvjete korištenja sustava. Sustav razlikuje dva tipa korisnika, sportaša, koji koristi sve usluge sustava, ali nema mogućnosti mentoriranja drugih, te trenera/mentora, koji ima dodatne pogodnosti koje mu pomažu pri mentoriranju. Nakon registracije, slijedi verifikacija. Sustav šalje e-mail s verifikacijskom poveznicom na unesenu adresu pri registraciji. Korisnik ne može pristupiti web sustavu putem prijave sve dok ne potvrdi autentičnost posjedovanja e-mail adrese. Nakon što je verifikacija uspješno prošla, korisnik putem prijave ulazi u sustav. Sada se korisnik nalazi na svojoj upravljačkoj ploči gdje ima grafički prikaz na svoj tjelesni napredak, kao i prikaz na mogućnosti koje sustav nudi.
Korisniku je omogućeno da sprema i vizualizira tjelesne mjere te napredak u snazi. Podaci se spremaju u bazu podataka na serveru te se grafički vizualiziraju kako bi dali široku sliku stanja kroz vrijeme. Korisnik može kreirati svoje vlastite programe vježbanja, ili u slučaju da je korisnik i mentor, postoji mogućnost slanja istih klijentima. Sustav je tako da služi korisnicima kao baza podataka koja nudi pronalaženje željenog mentora, ali i mogućnosti da sami pohranjuju vlastite programe u bazu ili ih po želji komercijaliziraju. Svaki od korisnika može ocijeniti pojedini program vježbi ocjenama od 1 do 10, jednako kao i mentore. Sustav ocjenjivanja korisnika i programa vježbanja pomoći će kod filtriranja neozbiljnih korisnika. Svaki korisnik ima mogućnost predstaviti sebe vizualno kroz 17 fotografija. Mentori fotografijama mogu prezentirati sebe i svoj rad. Sustav sadrži i funkcije pretraživanja i filtriranja rezultata pri traženju ostalih korisnika ili programa za vježbanje. Jedna od stavki koja olakšava komunikaciju između korisnika jest integrirani sustav prenošenja poruka i obavijesti. Korisnici mogu međusobno komunicirati u realnom vremenu, kao i primat određene

obavijesti, npr.: mentor može primiti obavijest o novom zahtjevu za mentorstvo. Nekoliko funkcija koje sustav nudi korisnicima koji se deklariraju kao mentori, jesu praćenje aktivnosti svojih korisnika i slanje programa kreiranih unutar sustava klijentima. Sustav ima cilj motivirati korisnike na tjelesnu aktivnosti i zdraviji život. Jedan od načina na koji to postiže je prikaz BMI-a (Body mass index), omjera mišića i masti u tijelu u kilogramima i ljestvicu, tj. postotak korisnika koji su postigli bolji ili gori rezultat kod tjelesne kompozicije.

4.2 Registracija korisnika

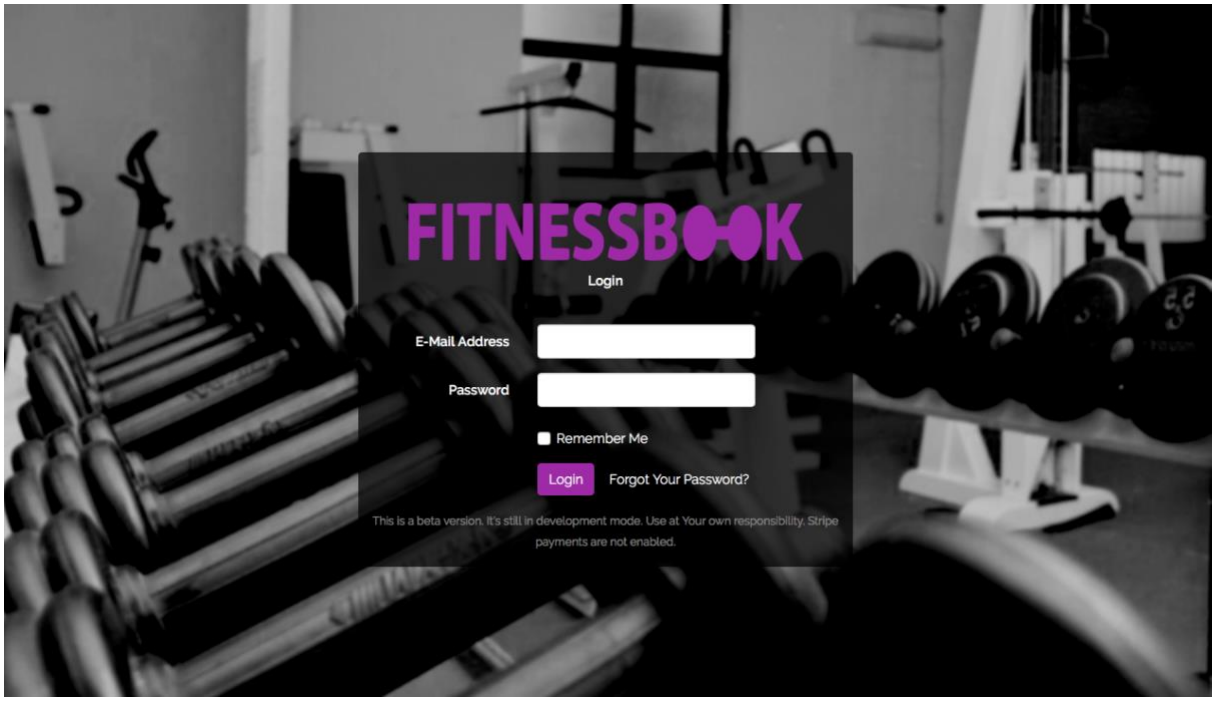
Pogled registracije traži od korisnika unos 7 tipova podataka kako bi se mogao registrirati unutar sustava. Ime i prezime koje služe kao identifikacija korisnika unutar sustava i vidljivo je ostalim sudionicima unutar sučelja te elektronska pošta i lozinka koji služe za identifikaciju korisnika. Nakon toga korisnik se identificira kao sportaš ili trener te odabire spol. Ovisno o ovoj početnoj konfiguraciji "sportaš/trener", korisniku se prikazuju dodatne opcije unutar sustava. Odabir spola utječe na određene kalkulacije kao npr. kod indexa tjelesne mase.



Slika 25. Registracija korisnika, izvor: Izradio autor

4.3 Prijava korisnika

Korisnici se nakon uspješne potvrde svoga računa mogu prijaviti i pristupiti sustavu. Prijava se odvija na pogledu Login, gdje korisnik upisuje elektronsku poštu i lozinku. U slučaju da je prijava uspješna, korisnika se prosljeđuje u sustav, točnije, na pogled za vizualizaciju napretka.



Slika 26. Prijava korisnika, izvor: Izradio autor

4.4 Vizualizacija napretka korisnika

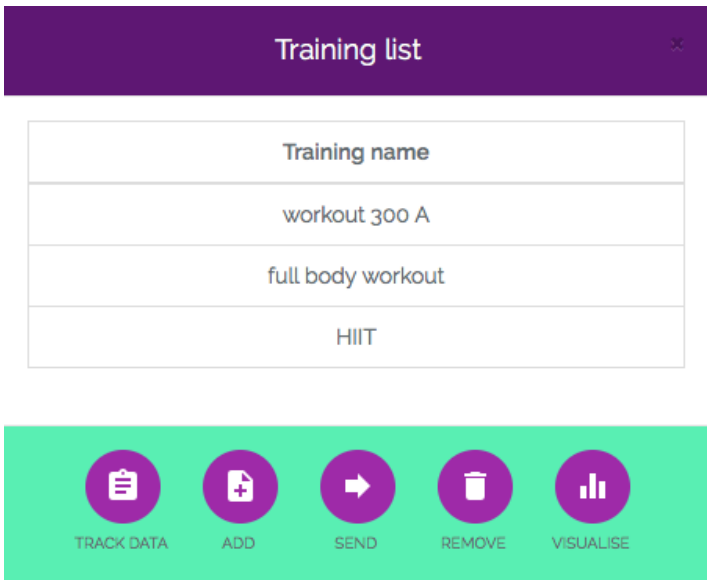
Slika 26. prikazuje prvi pogled koji korisnik ima kada ulazi u sustav. Isti ujedno označava i upravljačku ploču jer korisniku omogućuje da se kreće putem navigacijskog bara. Korisnik ima pregled na sliku koja predstavlja ljudsko tijelo. Klik na pojedini dio ljudskoga tijela stvara formu za unos mjere i spremanje u bazu podataka. Svaki dio tijela ima svoj graf, što znači da postoji 13 polja koja označavaju dijelove tijela za odabir te visinu i težinu kao odvojeno područje. Ispod slike prikaza tijela nalazi se linearni graf koji prikazuje postotak korisnika koji su bolji od korisnika profila. Na slici 27., korisniku linearni graf prikazuje 100% jer trenutno nema korisnika

koji imaju bolju tjelesnu kompoziciju unutar sustava. Nakon toga slijedi Pie graf koji prikazuje omjer mišićne mase i masnih naslaga korisnika.

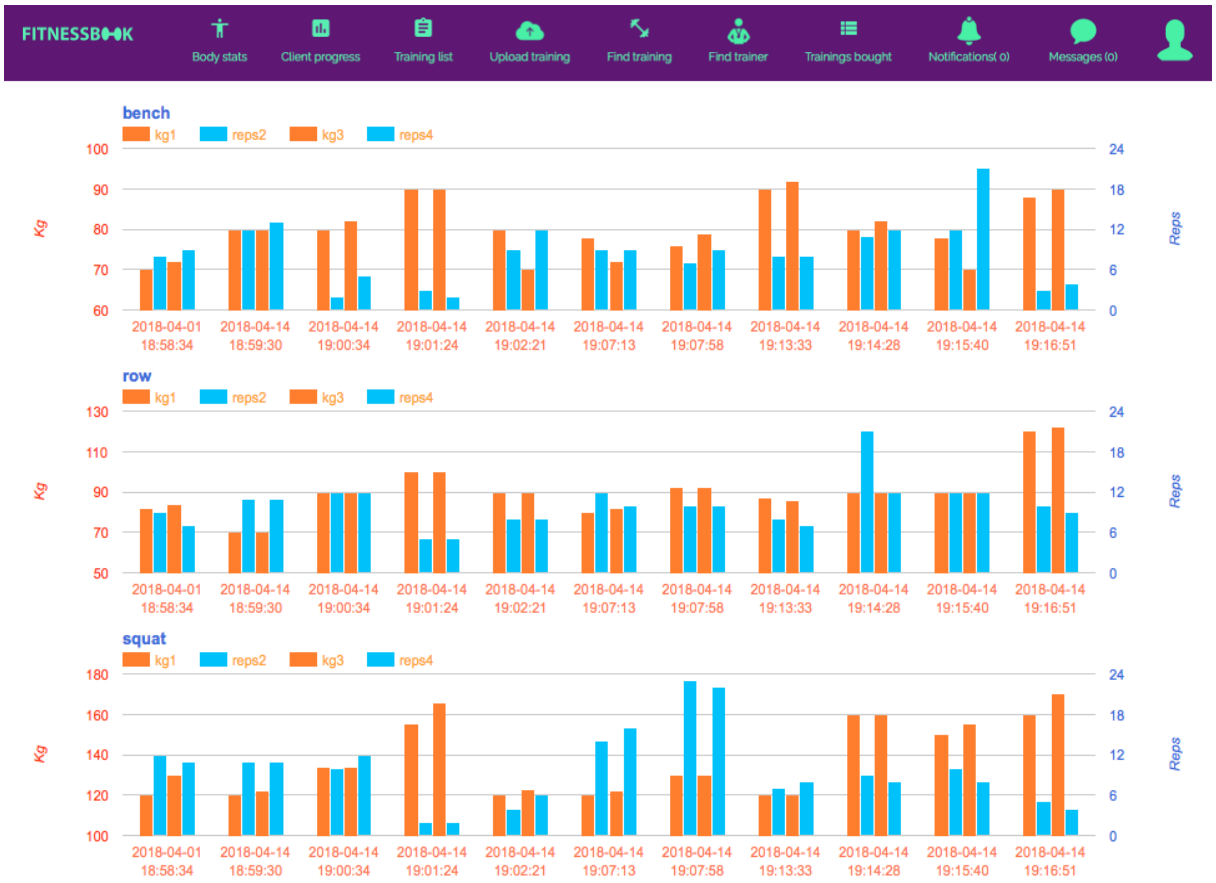
Slika 27. Tjelesne mjere korisnika, izvor: Izradio autor

4.5 Osobne vježbe – kreiranje, korištenje i vizualizacija

Klikom na gumb ”training list” pojavljuje se prozorčić s listom osobnih rutina vježbanja. Gumb ”track data” otvara prozor za izabranu rutinu u koji se unose podaci o obavljenom treningu. Na gumb ”Add” otvara se prozor za kreaciju vlastitog treninga gdje korisnik upisuje podatke o vježbama, serijama i sl. Gumb ”send” imaju samo korisnici koji su i treneri. Oni mogu putem gumba slati vlastoručno izrađen trening svojim klijentima. Pritiskom na taj gumb otvara se lista klijenata te korisnik pri odabiru šalje trening. Klijent će tada moći vizualizirati primljeni trening u svojoj listi. ”Remove” briše trening s liste i baze podataka. Odabirom ”visualise” opcije, prikazat će se grafovi zapisa izabranog treninga.



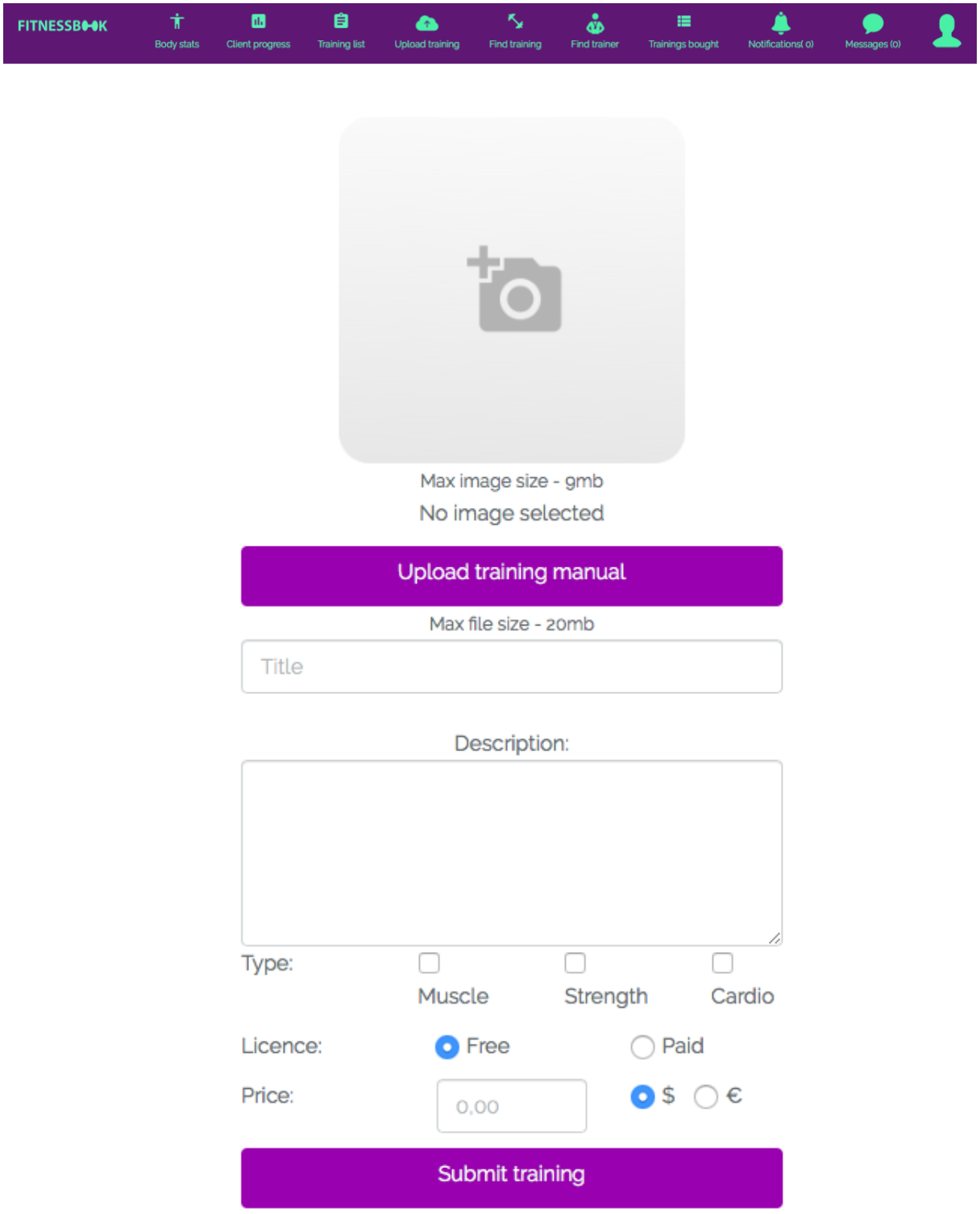
Slika 28. Prikaz liste treninga i dodatne opcije, izvor: Izradio autor



Slika 29. Prikaz povijesti treninga korisnika , izvor: Izradio autor

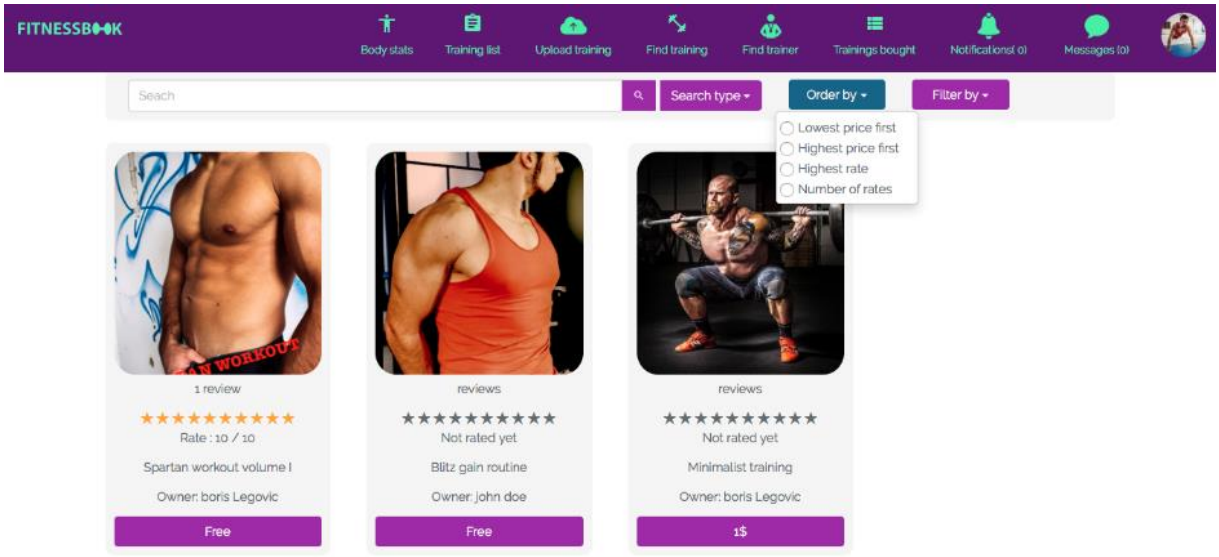
4.6 Dijeljenje rutina za vježbanje

Klik na gumb ”upload training” odvodi na pogled za pohranu podataka o treningu unutar kojega korisnik može pohraniti vlastite datoteke o vježbanju i dijeliti ih besplatno ili naplaćivati. Korisnik u pogledu za pohranu treninga mora pohraniti fotografiju koja predstavlja njegovu datoteku o vježbanju, naslov kao ime rutine i opis koji korisnicima služi za uvid u detalje o programu koji se nudi. Sljedeće što korisnik mora postaviti je licenca, koja može biti besplatna ili se može naplaćivati. U slučaju da se naplaćuje, korisnik mora unijeti i cifru te izabrati valutu.



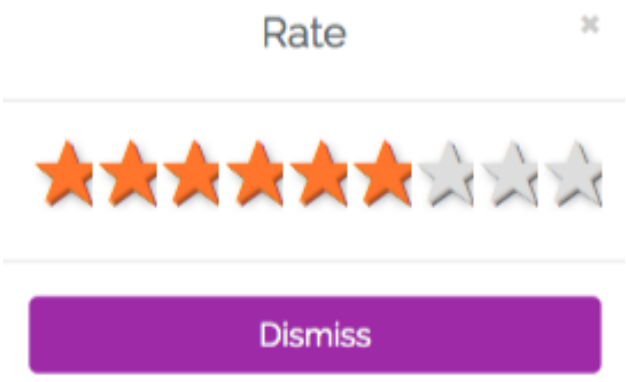
Slika 30. Prikaz pogleda za pohranu treninga, izvor: Izradio autor

Rutine koje su pohranjene tako moguće je otvoriti klikom na gumb ”find training” koji otvara pogled na listu svih treninga. Tu je moguće filtrirati i pretraživati treninge tražilicom koja se nalazi pri vrhu pogleda, iznad liste.

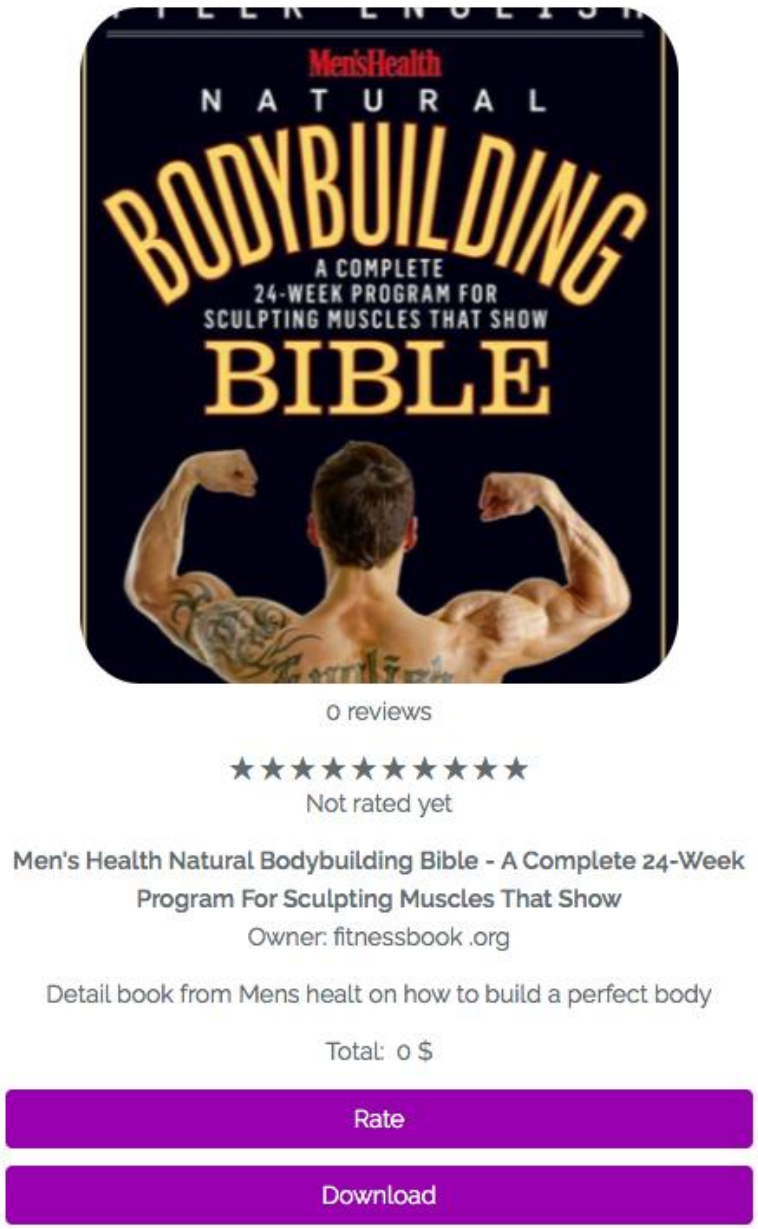


Slika 31. Pregled programa s vježbama, izvor: Izradio autor

Korisnik klikom na željeni trening otvara novi pogled u kojem se nalazi detaljan opis treninga, mogućnost za ocjenjivanje treninga i preuzimanje. Kod treninga koji se naplaćuju potrebno je unijeti broj kartice i kod uspješnog plaćanja, treninge je moguće preuzeti. Korisnik jednom kupljeni trening uvijek može besplatno preuzeti ponovo. U slučaju da se trening želi ocijeniti, gumb ”rate” otvara preglednik s mogućnošću davanja ocjene od 1 do 10. Korisnik uvijek može mijenjati svoju ocjenu te može ocijeniti samo besplatne i treninge koje je kupio. To znači da nije moguće dati ocjenu za trening koji se naplaćuje, a nije preuzet od korisnika.



Slika 32. Ocjenjivanje programa vježbi, izvor: Izradio autor

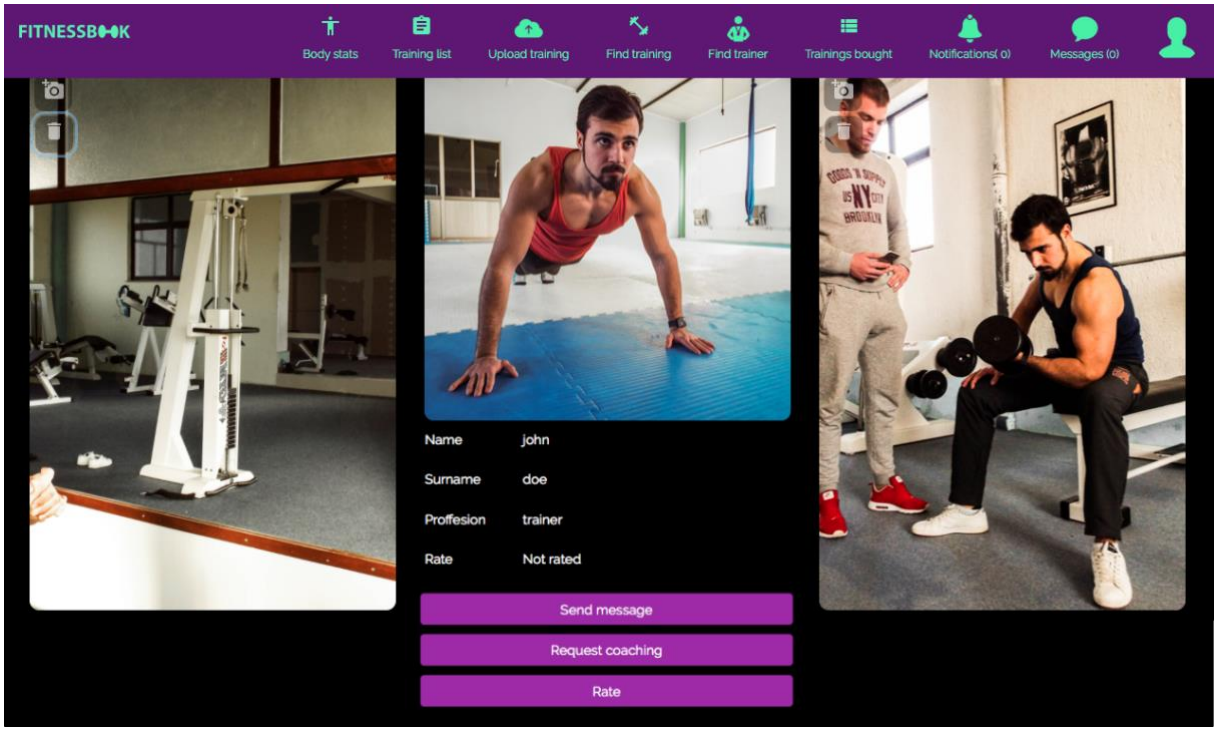


Slika 33. Detaljni pregled izabranog programa za vježbanje, izvor: Izradio autor

4.7 Mentoriranje

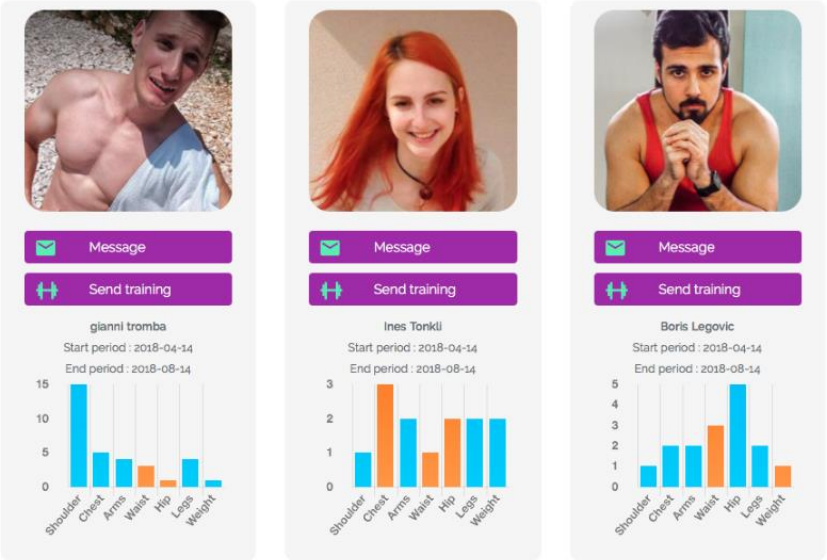
Korisnici gumbom ”find trainer” ulaze u pogled s listom svih trenera. Unutar pogleda za pretragu, mogu se koristiti funkcije filtriranja i pretraživanja. Filtriranje nudi mogućnost pretrage od najviše do najniže cijene kao i obrnuto, najvećim ocjenama ili brojem ocjena. Unutar ovoga pogleda moguće je pretraživati i obične korisnike. Korisnik putem klika na profil otvara pogled koji vodi na profil korisnika. Unutar tog

pogleda, moguće je ocijeniti trenera, poslati poruku ili zatražiti mentorstvo. Gumb za potražnju mentorstva odvodi na pogled za plaćanje mentoriranja. Unutar njega korisnik unosi broj kartice i vrši plaćanje mentoru. U slučaju da mentorstvo još uvijek traje, korisniku će se prikazati obavijest o roku trajanja mentoriranja i neće biti u mogućnosti uplatiti novo mentorstvo dok trenutno ne istekne.



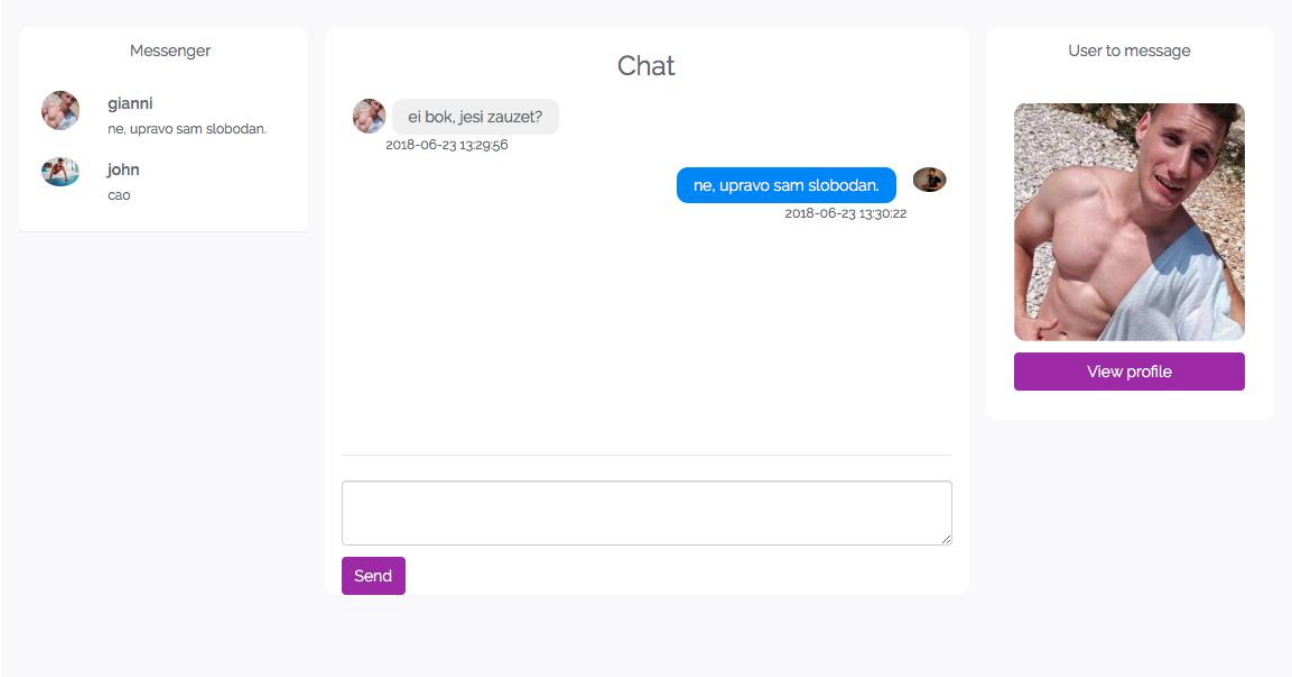
Slika 34. Pregled profila trenera iz perspektive korisnika, izvor: Izradio autor

Trener nakon uplate biva obaviješten *notifikacijom* o novom mentorstvu i može pregledati svoje klijente klikom na ”Client progress” unutar navigacijskog bara, što će otvoriti pogled s listom klijenata.



Slika 35. Pregled klijenata iz perspektive trenera, izvor: Izradio autor

Unutar ovog pogleda, trener može vizualizirati napredak svojih klijenata. Ovakav način vizualizacije omogućuje konstantno praćenje napretka, a gumb ”*Message*” omogućuje da se otvori komunikacijski kanal s korisnikom, odnosno, pošalje poruka. Trener također ima mogućnost slanja osobnih treninga klijentu koje je izradio unutar sustava. Gumb otvara listu treninga, te odabirom, trening se šalje korisniku. Korisnici također mogu komunicirati putem sustava za izmjenu poruka. Mogu međusobno izmjenjivati poruke, pritiskom na gumb ”*message*” ili otvaranjem pogleda za razgovore u navigacijskom baru. Pogled za razgovore sadrži listu svih razgovora, prozor za slanje poruka i prikaz trenutnog razgovora te fotografiju s gumbom za navigaciju na korisnički profil kao što je prikazano na slici 36.

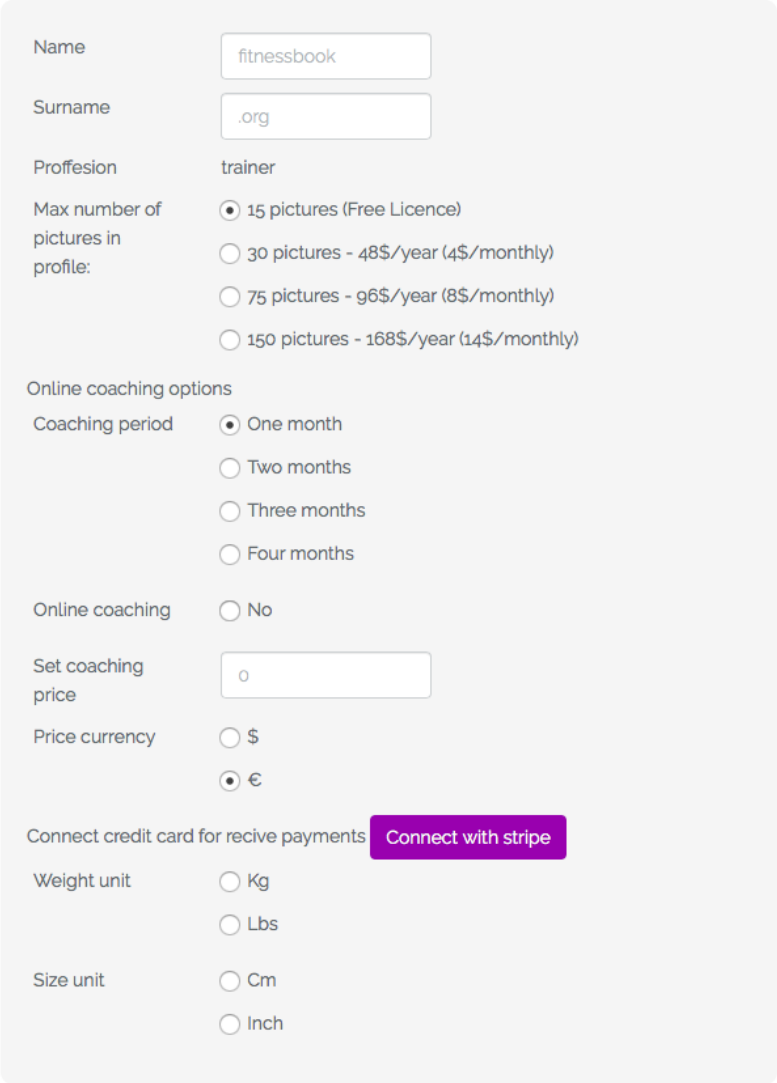


Slika 36.Prozor za razgovore, izvor: Izradio autor

4.8 Korisničke opcije i ostale funkcionalnosti

Unutar navigacijskog bara nalazi se gumb sa slikom profila korisnika. Klikom na taj gumb pojavljuje se padajući izbornik koji nudi 3 opcije korisniku – ”Profile”, ”Edit profile” i ”Logout”. ”Profile” gumb vodi k pogledu koji prikazuje osobni profil korisnika. Unutar tog pogleda, korisnik uređuje svoj profil slikama, tj. prezentira se javnosti kroz fotografije. Korisnik može brisati ili zamijeniti slike na pojedinim pozicijama klikom na gumb s fotografijom ili kantom za smeće.
”*Edit profile*” vodi korisnika u izbornik s opcijama. Ovdje korisnik može mijenjati ime i prezime u prve dvije kućice označene sa ”*name*” i ”*surname*”. Sljedeće što korisnik može izabrati jest broj slika vidljivih na profilu. Početna besplatna postavka je označena na 15 fotografija, a ako korisnik odabere opciju s više fotografija, kreirat će se polje za unos broja kartice i gumb s uplatom. Korisnik koji se deklarira kao trener, ima dodatne opcije unutar postavki. Tako postoji opcija koja definira period trajanja mentoriranja. Stavka ”*coaching period”* omogućuje odabir trajanja mentoriranja od jednog do četiri mjeseca pa zatim ”*online coaching*” koji omogućuje da mentor izabere želi li pružati usluge mentoriranja. Početne postavke nemaju omogućeno mentoriranje jer sustav ne dopušta uslugu prije nego korisnik izradi ili spoji svoj Stripe

račun na sustav, budući da se uplate ne mogu vršiti dok nema podataka o računu. Prozor ”*Set coaching price*” omogućuje korisniku da odredi cijenu mentoriranja. Zatim slijedi ”*price currency*” koji nudi odabir između eura ili dolara kao valute za uplatu.



Slika 37. Pregled opcija trenera*, izvor: Izradio autor*

Ovim nizom završene su posebne opcije koje posjeduje korisnik-trener.
Slijedi ”weight unit” koji određuje tip težine koji će korisnik koristiti tijekom korištenja sustava – kg ili lbs. Posljednja opcija je ”size unit” kojim se postavlja jedinice mjerenja - cm ili inch.
Posljednji gumb padajućeg izbornika je ”Logout” koji vraća korisnika na početni ekran stranice i odjavljuje ga iz sustava.

ZAKLJUČAK

Izuzetno brz razvoj tehnologije ulaskom u trenutno tisućljeće uveo je mnoge novine u poslovanje i svakodnevan život ljudi. Rasprostranjenost interneta i mobilne tehnologije pridonijeli su globalizaciji poslovanja, osigurali nova poslovna rješenja i radna mjesta. Iz toga razloga, početkom 2000-ih pa do danas, postoji eksponencijalni rast sjedilačkih radnih mjesta. Istraživanja potvrđuju da odrasli provode čak 70% više radnog vremena sjedeći (Sedementary behavior: emerging evidence for a new health risk, 2010). Sve prisutniji sjedilački način života pridonio je popularnosti rekreativnog bavljenja sportom. U tome ulogu imaju sportski treneri. Fitnessbook – informacijski sustav za sportsko mentoriranje omogućuje sportašima da pronađu idealni plan vježbanja ili mentora koji će ih voditi kroz proces vježbanja, a s druge strane omogućuje trenerima da pronađu svoje klijente. Tehnologija omogućuje rad na daljinu, tj. ne zahtijeva fizičku prisutnost osoba na određenoj lokaciji. Upravo je to prednost koju nosi ovaj projekt, sadržavajući funkcije koje omogućuju korisnicima da umanje mane osobnog mentorstva na daljinu.

Literatura

Knjige

[1] Mile Pavlić – Informacijski sustavi , Školska knjiga d.o.o.,2011

[2] Strahonja, V., Varga,M. Pavlić, M – Projektiranje informacijskih sustava – metodološki priručnik, ZID i INAINFO, 1992.

[3] Brodie, M. L., Mylopoulos, J., Schmidt,J.W. – On Conceptual Modelling, Springer-Verlag, 1984

[4] Cippico, V. – Faza izvedbe informacijskog sustava, Savjetovanje CASE 9, 1996.

[5] Varga, M. – Faze razvoja informacijskog sustava, Infotrend,1996

[6] Simon, J.C. – Introduction to Information Sustems, John Wiley & Sons,2001

[7] Inmon, W.H. – Information Systems Arhitecture – a System Developers Primer, Prentice Hall, 1986

[8] Hawryszkiewycz, I.T. – System Analysis and Design, Prentice-Hall, 1988.

[9] Barker, R., Longman, C. – CASE Methid FUnction and Proces Modelling. ORACLE, 1992.

[10] C. Lindley, - Jquery Succinctly, Syncfusion, 2012.

[11] Dujšin U. – Globalizacija, ekonomske integracije i Hrvatska, Zbornik Pravnog fakulteta u Zagrebu, 1999.

Web izvori

[1] https://laravel.com/docs/5.6, [Pristupljeno: 10.06.2018]

[2] https://stripe.com/docs/connect/standard-accounts [Pristupljeno:18.05.2018]

[3] https://stripe.com/docs [Pristupljeno: 15.05.2018]

[4] https://pusher.com/docs [Pristupljeno: 12.05.2018]

[5] https://www.digitalocean.com/community/tags/product-documentation/tutorials [Pristupljeno 10.05.2018]

[6] https://creately.com/diagram-type/use-case [Pristupljeno 02.05.2018]

Popis slika

Slika 1. Use-case diagram sustava, izvor: izradio autor.....16

Slika 2. Primjer Env konfiguracijske datoteke, izvor: izradio autor……17

Slika 3. Primjer Env konfiguracijske datoteke, izvor: izradio autor……18

Slika 4. Primjer kontrolera za registraciju korisnika, izvor: izradio autor……..20

Slika 5.Primjer slanja poštanske adrese registriranom korisniku, izvor: izradio autor……21

Slika 6.Primjer slanja poštanske adrese registriranom korisniku, izvor: izradio autor.....22

Slika 7. Dohvat podataka o mjerama tjela korisnika, izvor: izradio autor……...23

Slika 8.Korištenje ”Chart.js” biblioteke za prikaz grafova, izvor: izradio autor…….25

Slika 9. Slušanje DOM elemenata, izvor: izradio autor……27

Slika 10. Funkcija za pohranu podataka na server , izvor: izradio autor……..27

Slika 11. Adresa za registraciju korisnika na Stripe platformu , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 01.05.2018……28

Slika 12. Adresa za registraciju korisnika na Stripe platformu , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 01.05.2018…...….29

Slika 13. Naplata iznosa i uzimanje provizije , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno:05.05.2018……....30

Slika 14. Tijek transakcije kod direktne naplate korisnika , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 15.05.2018……...30

Slika 15. Tijek transakcije kod preuzimanja provizije na transakciju između dva korisnika , izvor: https://stripe.com/docs/connect/standard-accounts, pristupljeno: 16.05.2018……..31

Slika 16. Konfiguracijska datoteka unutar Pusher-a, izvor: https://laravel.com/docs/5.6/broadcasting, pristupljeno:12.06.2018……..…33

Slika 17. Konfiguracija Echo broadcastera, izvor: https://laravel.com/docs/5.6/broadcasting, pristupljeno:14.06.2018……..33

Slika 18. Konfiguracija privatnog broadcast kanala, izvor: izradio autor…….34

Slika 19. Slušači unutar javascript funkcija, izvor: izradio autor…….35

Slika 20. Kreiranje događaja za slanje poruka, izvor: izradio autor……...36

Slika 21. Instalacija drivera za spremanje datoteka, izvor: https://laravel.com/docs/5.6/filesystem, pristupljeno: 15.06.2018……41

Slika 22. Instalacija drivera za spremanje datoteka, izvor: https://laravel.com/docs/5.6/filesystem, pristupljeno: 16.06.2018……..41

Slika 23. Konfiguracija Spaces-a, izvor: Izradio autor…….41

Slika 24. Instalacija drivera za spremanje datoteka, izvor: Izradio autor……..42

Slika 25. Registracija korisnika, izvor: Izradio autor……44

Slika 26. Prijava korisnika, izvor: Izradio autor…….45

Slika 27. Tjelesne mjere korisnika, izvor: Izradio autor……46

Slika 28. Prikaz liste treninga i dodatne opcije, izvor: Izradio autor……..47

Slika 29. Prikaz povijesti treninga korisnika , izvor: Izradio autor……..48

Slika 30. Prikaz pogleda za pohranu treninga, izvor: Izradio autor……49

Slika 31. Pregled programa s vježbama, izvor: Izradio autor……...50

Slika 32. Ocjenjivanje programa vježbi, izvor: Izradio autor……...…….50

Slika 33. Detaljni pregled izabranog programa za vježbanje, izvor: Izradio autor…..51

Slika 34. Pregled profila trenera iz perspektive korisnika, izvor: Izradio autor…….52

Slika 35. Pregled klijenata iz perspektive trenera, izvor: Izradio autor……..……...53

Slika 36.Prozor za razgovore, izvor: Izradio autor……54

Slika 37. Pregled opcija trenera, izvor: Izradio autor……..……...55

Bitbucket adresa:

git clone https://Boris_Legovic@bitbucket.org/Boris_Legovic/fitnessbook.git

Sažetak

Zadatak ovog diplomskog rada bio je izraditi informacijski sustav za sportsko mentoriranje koji bi sportašima omogućio platformu na kojoj mogu pratiti svoj tjelesni napredak, treninge i omogućiti im široku bazu programa za vježbanje i mentora/trenera, kao i trenerima omogućiti poslovanje na daljinu i praćenje napretka svojih klijenata.

Abstract

The main assignment behind this final thesis was to develop a system for sports mentoring that would enable athletes to have a platform for track their physical progress, trainings and provide them with a broad base of exercise programs and mentors / trainers as well as trainers to enable remote business and track customer performance .