Uspostava kontejnerske virtualizacije primjenom Docker tehnologije na primjeru PHP aplikacije

Vasiljević, Filip

VELEUČILIŠTE U RIJECI

Filip Vasiljević

USPOSTAVA KONTEJNERSKE VIRTUALIZACIJE PRIMJENOM DOCKER TEHNOLOGIJE NA PRIMJERU PHP APLIKACIJE

(završni rad)

Rijeka, 2021

VELEUČILIŠTE U RIJECI

PREDDIPLOMSKI STRUČNI STUDIJ TELEMATIKA

USPOSTAVA KONTEJNERSKE VIRTUALIZACIJE PRIMJENOM DOCKER TEHNOLOGIJE NA PRIMJERU PHP APLIKACIJE

(završni rad)

MENTOR

izv. prof. dr. sc. Alen Jakupović, prof. v. š.

STUDENT

Filip Vasiljević

MBS: 2427000011/17

Rijeka, srpanj 2021.

VEUČILIŠTE U RIJECI
Preddiplomski stručni studij Telematika
Rijeka, 4.5.2021.

ZADATAK
za završni rad

Pristupnik Filip Vasiljević, MBS: 2427000011/17.

Studentu preddiplomskog stručnog studija Telematika izdaje se zadatak za završni rad – tema završnog rada pod nazivom:

USPOSTAVA KONTEJNERSKE VIRTUALIZACIJE PRIMJENOM DOCKER TEHNOLOGIJE NA PRIMJERU PHP APLIKACIJE

Sadržaj zadatka:
Opisati primijenjene tehnologije i alate koji su se koristili u razvoju web aplikacije, uspostavi kontejnerske virtualizacije i njezina postavljanja na produkcijsko računalo – posebno korišteni IDE, osnove HTML i PHP, te Docker tehnologije. Prikazati osnovne zahtjeve koje web aplikacija treba zadovoljiti, te njezino grafičko sučelje. Izraditi odgovarajuće modele i crteže. Detaljno opisati postavljanje baze podataka, implementacije web aplikacije te izrade Docker kontejnera. Opisati postavljanje Docker kontejnera na produkcijsko računalo na AWS platformi.

Preporuka:

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 4.5.2021. **Predati do: 15.9.2021.**

Mentor:
(izv.prof.dr.sc. Alen Jakupović, prof.v.š.)

Voditelj studija:
(Damir Malnar, predavač)

Zadatak primio dana: 4.5.2021.

Pristupnik:
(Filip Vasiljević)

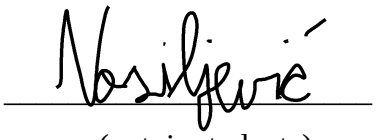
Dostavlja se:
- mentoru
- pristupniku

I Z J A V A

Izjavljujem da sam specijalistički završni rad pod naslovom Uspostava kontejnerske virtualizacije primjenom Docker tehnologije na primjeru PHP aplikacije izradio samostalno pod nadzorom i uz stručnu pomoć mentora

izv. prof. dr. sc. Alen Jakupović, prof. v. š..

Filip Vasiljević



(potpis studenta)

SAŽETAK

U ovom radu biti će opisana izrada aplikacije, dokerizacija iste aplikacije te stavljanje aplikacije na AWS. U početku rada će ukratko biti opisani programi koji su korišteni za izradu ovog sustava te će biti točno navedeno za što su se koristili. Nadalje u radu se opisuje struktura web sjedišta te koji će biti statički i koji će biti dinamički dijelovi web aplikacije. U izradi sustava će biti dijagram tijeka te izrada baze podataka. Kasnije u izradi sustava će biti opis web aplikacije tj. opis svih njezinih datoteka. U tom opisu biti će navedeno čemu služe i što rade određeni dijelovi datoteka te će također biti opisano povezivanje baze podataka sa aplikacijom. Slijedi izrada Docker spremnika tj. izrada spremnika sa web serverom i izrada spremnika sa aplikacijom te njihovo povezivanje. U stavljanju aplikacije na web biti će opisano stavljanje Docker spremnika na AWS.

Ključne riječi: Docker, AWS, web aplikacija, web sjedište, baza podataka.

SADRŽAJ

1. UVOD

U ovom radu raditi će se PHP web aplikacija dnevnika koja će biti povezana na bazu podataka ucka.veleri.hr. Web aplikacija će onda biti stavljena u Docker kontejner. Docker dio ovog završnog rada će se sastojati od dva djela. Prvi dio biti će izrada kontejnera sa web serverom, dok će drugi dio biti izrada PHP kontejnera kako bi kontejner sa web serverom mogao vidjeti aplikaciju koja je pisana u PHP-u. Nakon toga potrebno je web aplikaciju skupa s Docker datotekama staviti na web putem AWS-a. Unutar AWS-a najviše će se koristiti funkcija EC2 za objavu aplikacije na web. Kada je web aplikacija objavljenja korisnik joj mora biti u mogućnosti pristupiti preko bilo kojeg računala povezanog na Internet (računalo također mora imati Internet preglednik kao što su Google Chrome, Mozilla Firefox, Microsoft Edge, Internet Explorer, itd.). Korisnik kada pristupi aplikaciji mora se logirati i kada se logira može dodavati, pregledavati, uređivati i brisati unose u dnevnik.

U bazi podataka nalaziti će se dvije tablice, jedna za korisnike, a druga za unose u dnevnik preko kojih će se određenom korisniku prikazivati samo njegovi unosi u aplikaciju i neće moći vidjeti niti išta raditi sa unosima od drugih korisnika. Svaki korisnik kada se upiše na aplikaciju će dobiti svoj identifikacijski broj u bazi te se po tom identifikacijskom broju vide koji su unosi u dnevnik njegovi.

2. OPIS SUSTAVA

Zadatak je izraditi aplikaciju u PHP-u u kojoj se može dodavati korisnike. Korisnici pišu svoje dnevnike koji se spremaju u bazu podataka. Aplikacija će se nalaziti na webu te će se moći pristupiti preko bilo kojeg internet preglednika.
Izrada zadatka se sastoji od 3 dijela:

- Izrada aplikacije u PHP-u,
- Izrada kontejnera u Docker-u te stavljanje aplikacije u njega,
- Stavljanje kontejnera na Web uz pomoć AWS-a.

Za izradu ovog zadatka korišteno je:

- Sublime text 3,
- Docker,
- HTML,
- Microsoft Visual Studio Code
- PHP,
- AWS,
- XAMPP,
- Microsoft Command prompt,
- MySQL baza podataka,
- PuTTYgen,
- FileZilla.

2.1. Sublime text 3

Sublime text 3 je trenutna verzija programa Sublime text. Sublime text je program za uređivanje koda. Izvorno podržava puno programskih jezika, a one koje ne podržava mogu se dodati putem *plug-inova*. Sublime text 3 je besplatna aplikacija te je napravljena u Pythonu i C++. Sublime text 3 se u ovom zadatku koristi za izradu aplikacije u PHP-u te za spajanje baze podataka sa aplikacijom. (Sublime text (software), 2021)

2.2. Docker

Docker je otvorena platforma za razvoj, slanje i pokretanje aplikacija. Docker vam omogućuje odvajanje aplikacija od vaše infrastrukture kako bi ste mogli brzo i lagano isporučiti softver. Pomoću Docker-a možete upravljati svojom infrastrukturom na isti način na koji upravljate svojim aplikacijama. Koristeći prednost Docker-ovih metodologija za brzu isporuku, testiranje i implementaciju koda možete znatno povećati brzinu između pisanja koda i njegovog pokretanja u proizvodnji. Docker omogućava pakiranje i pokretanje aplikacije u izoliranom okruženju koje se naziva spremnik. Izolacija i sigurnost omogućavaju pokretanje više spremnika na istome domaćinu (*host*). Spremnici zauzimaju malu količinu memorije te sadrže sve što je potrebno za pokretanje aplikacije tako da se ne morati oslanjati na softver koji je instaliran na računalu na kojem trenutno radite. Dok radite možete jednostavno dijeliti spremnike i biti sigurni da svi s kojima radite dobivaju isti spremnik koji funkcionira na isti način. Docker pruža alat i platformu za upravljanje životnim ciklusom vaših spremnika:

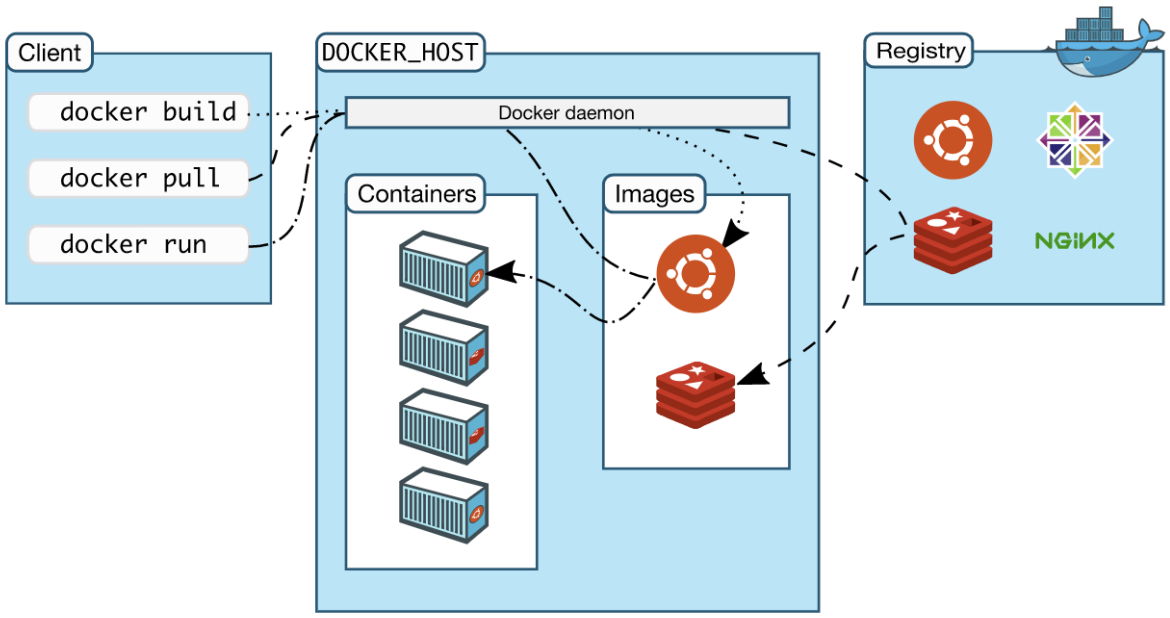
- Razvijanje aplikacije i njezinih pratećih komponenti pomoću spremnika,
- Spremnik postaje jedinica za distribuciju i testiranje aplikacije,
- Pri završetku razvoja aplikacije raspoređuje se u produkcijsko okruženje kao spremnik. To funkcionira na isti način ako je proizvodno okruženje vaš lokalni centar, cloud ili hibrid između dvoje.

Docker se može koristiti za:

- Brzu i konzistentnu isporuku aplikacije,
- Brzi „*deployment*“ aplikacija,
- Pokretanje više aplikacija na istome hardveru.

Docker koristi klijent-poslužitelj arhitekturu. Docker klijent razgovara s Docker-ovim *daemon* koji gradi, pokreće i distribuira Docker spremnike. Docker klijent i *daemon* mogu raditi na isto sustavu ili se Docker klijent može povezati s udaljenim *daemon*-om. Docker klijent i *daemon* komuniciraju putem REST API-ja putem UNIX utičnica ili mrežnog sučelja. Docker klijent može bit Docker-compose koji vam omogućuje rad s aplikacijama koje se sastoje od više spremnika.

Slika 1. Docker arhitektura



Izvor: Docker Overview, 2021

Docker *deamon* sluša za Docker API zahtjeve i upravlja Docker objektima kao što su slika, spremnik itd.. *Daemon* može komunicirati sa drugim *daemon*-ovima radi upravljanja Docker-ovim uslugama. Docker klijent je primarni način za komunikaciju s Docker-om. Kada se koriste naredbe kao: docker run; klijent šalje te naredbe *daemon*-u koji ih onda izvede. Docker naredbe koriste Docker API-ja. Korisnik može komunicirati sa više *daemon*-a od jednom. Docker hub je javni registrator kojim se mogu koristiti svi korisnici Docker-a. Slike su predlošci sa uputama za stvaranje Docker spremnika. Slike su često bazirane na drugim slikama npr. izradimo sliku koja se bazira na slici sustava Ubuntu, ali na to još dodamo Apache web server i našu aplikaciju. Spremnici su pokrenute verzije slika. Spremnici se mogu pokrenuti, zaustaviti, premjestiti i obrisati koristeći Docker API ili CLI. U ovom radu Docker se koristi za stavljanje aplikacije u spremnik sa web serverom. (Docker Overview, 2021 i Docker (software), 2021)

2.3. HTML

Hypertext Markup Language (HTML) je standardni jezik za dokumente dizajnirane za prikaz u web-pregledniku. Mogu mu pomoći tehnologije poput „kaskadnih tablica stilova“ (CSS) i programskih jezika, kao što je JavaScript. Web-preglednici primaju HTML

dokumente s web poslužitelja ili iz lokalne pohrane i dokumente pretvaraju u multimedijske web stranice. HTML opisuje semantičku strukturu web stranice i izvorno sadrži znakove izgleda dokumenta. HTML elementi su sastavni dijelovi HTML stranica. Pomoću HTML konstrukcija, slike i drugi objekti, poput interaktivnih oblika, mogu biti ugrađeni u prikazanu stranicu. HTML pruža način za stvaranje strukturiranih dokumenata označavanjem strukturne semantike za tekst kao što su naslovi, odlomci, popisi, poveznice, citati i druge stavke. HTML elementi razgraničeni su oznakama, napisani uglatim zagradama. Oznake poput <img/> i <input/> izravno uvode sadržaj na stranicu. Ostale oznake poput <p> okružuju i pružaju informacije o tekstu i mogu sadržavati druge oznake kao podelemente. Preglednici ne prikazuju HTML oznake, ali ih upotrebljavaju za tumačenje sadržaja stranice. HTML se u ovom zadatku koristi za kreiranje web stranice i za izradu vizualnih dijelova aplikacije. (HTML, 2021)

2.4. Microsoft Visual Studio Code

Visual Studio Code je program za uređivanje koda koji je razvio Microsoft za Windows, Linux i macOS. Uključuje podršku za uklanjanje pogrešaka, ugrađenu Git kontrolu i GitHub, isticanje sintakse, inteligentno dovršavanje koda, isječke i preuređivanje koda. Vrlo je prilagodljiv, omogućavajući korisnicima da promijene temu, prečace na tipkovnici, postavke i instaliraju proširenja koja dodaju dodatnu funkcionalnost. Izvorni kod je besplatan, a pušta se pod dozvoljenom MIT licencom. Sastavljene binarne datoteke su besplatne za privatnu ili komercijalnu upotrebu. Visual Studio Code temelji se na Electron-u, okviru koji se koristi za razvoj Node.js aplikacija za radnu površinu koja radi na mehanizmu za *Blink layout*. Iako koristi okvir Electron-a, softver ne koristi Atom i umjesto toga koristi istu komponentu uređivača (kodnog naziva "Monaco") koja se koristi u Azure DevOps. U Anketi za razvoj programera Stack Overflow 2019 Visual Studio Code ocjenjen je kao najpopularniji alat za razvojno okruženje za programere, a 50,7% od 87.317 ispitanika je tvrdilo da ga koriste. Visual Studio Code se koristi za uređivanje koda u raznim jezicima, uključujući:

- Java,
- JavaScript,
- Go,
- Node.js,

- Python,
- C++,
- PHP,
- Itd.

Umjesto projektnog sustava korisnicima, omogućuje otvaranje jednog ili više direktorija koji se potom mogu spremiti u radne prostore za buduću upotrebu. To omogućuje rad u bilo kojem jeziku. Mnoge naredbe Visual Studio Code-a nisu izložene putem izbornika ili korisničkog sučelja, ali se mogu dodati putem proširenja (*Plug-ins*). Proširenja uključuju dodatke za uređivanje teksta, jezičnu podršku, u ovom slučaju dodatak Docker-a, itd.. Jedna od važnijih značajka je mogućnost stvaranja proširenja koja dodaju podršku za nove jezike, teme i programe za otklanjanje pogrešaka, izvođenje statičke analize koda i dodavanje poveznica koda. Visual Studio Code uključuje više proširenja za FTP što omogućava korištenje softvera kao besplatne alternative za web razvoj. Kod se može sinkronizirati između Visual Studio Code-a i web poslužitelja bez preuzimanja dodatnog softvera. Visual Studio Code se u ovom projektu koristi za izradu konfiguracijskih Docker datoteka te za upravljanje Docker spremnicima pomoću proširenja. (Visual Studio Code, 2021)

2.5. PHP

PHP je skriptni jezik posebno pogodan za web razvoj. Izvorno ga je napravio programer Rasmus Lerdorf 1994. godine. PHP je izvorno bila skraćenica za *Personal Home Page*, ali sada predstavlja rekurzivnu kraticu PHP: *Hypertext Preprocessor*. PHP kod se obično obrađuje na web poslužitelju od strane PHP *interpreter*-a koji kod interpretira kao modul, *daemon* ili kao *Common Gateway Interface* (CGI). Na poslužitelju, rezultat interpretiranog i izvedenog PHP koda, koji može biti bilo koja vrsta podataka kao npr. HTML ili binarni slikovni podatci, činiti će cjeloviti ili dio HTTP odgovora u pregledniku. Uz to PHP se može koristiti za programske zadatke izvan konteksta internetskih aplikacija poput samostalnih grafičkih aplikacija ili robotske kontrole dronova. PHP kod se može direktno izvršiti iz naredbenog retka. U ovom radu PHP se koristi za izradu funkcionalnih dijelova web aplikacije. (PHP, 2021)

2.6. AWS

Amazon Web Services (AWS) je Amazonova podružnica koja pruža platforme i API-je za obavljanje zadataka u cloudu na zahtjev pojedinaca i tvrtka na osnovi plaćanja po zahtjevu. Te web usluge u cloudu pružaju raznovrsnu tehničku strukturu, blokove i alate za pomoć. Jedna od tih usluga je Amazon Elastic Compute Cloud (EC2) koja dopušta korisnicima da na raspolaganju imaju virtualnu nakupinu računala koja je cijelo vrijeme dostupna preko interneta. AWS-ova verzija virtualnih računala oponaša većinu atributa fizičkog računala uključujući procesore, grafičke kartice, RAM memoriju, memoriju na tvrdom disku ili na SSD disku, izbor operativnih sustava, umrežavanje i unaprijed učitani softver kao što su web poslužitelji, baze podataka i aplikacija za upravljanje odnosima sa kupcima (*customer relationship management*). U ovom radu najviše će se koristiti funkcija EC2 tj. funkcija za rad s instancama. (Amazon Web Services, 2021)

EC2 (Amazon Elastic Compute Cloud) je web usluga koja pruža siguran i promjenjiv računalni kapacitet u oblaku. Dizajniran je kako bi olakšao programiranje za Internet za programere. Jednostavno sučelje vam pruža potpunu kontrolu nad vašim računalnim resursima i omogućuje vam rad na Amazonovom računalnom okruženju. (Amazon EC2, 2021)

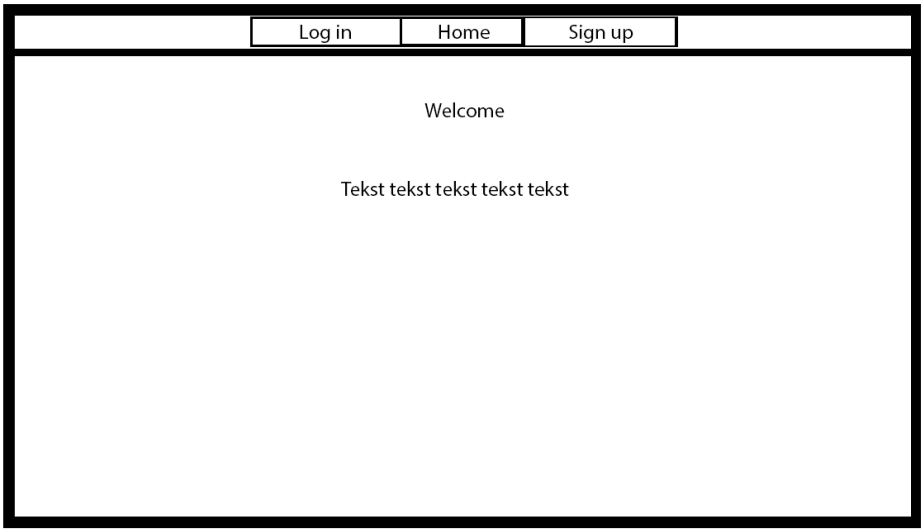
2.7. Ideja za web aplikaciju

Ideja za web aplikaciju došla je od profesora koji je predložio da se za taj dio završnog rada napravi dnevnik aplikacija. Dalje se ideja širila da se napravi login sustav povezan sa bazom podataka te da svaki korisnik može pisati svoj dnevnik te ga uređivati i brisati. Web sjedište trebalo bi biti jednostavno i lako za korištenje te korisnici koji ga po prvi put koriste ne bi trebali imati nikakvih problema sa navigacijom kroz aplikaciju.

2.8. Dizajn

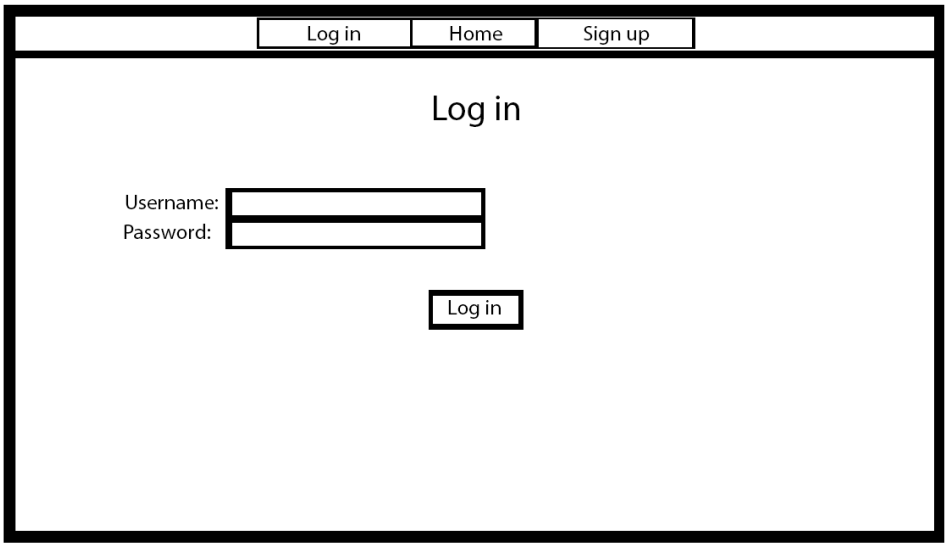
Dizajn aplikacije trebao bi biti jednostavan sa navigacijskom trakom na vrhu te u bojama lagodnim za oči.

Slika 2. Grubi dizajn web sjedišta



Izvor: izradio autor

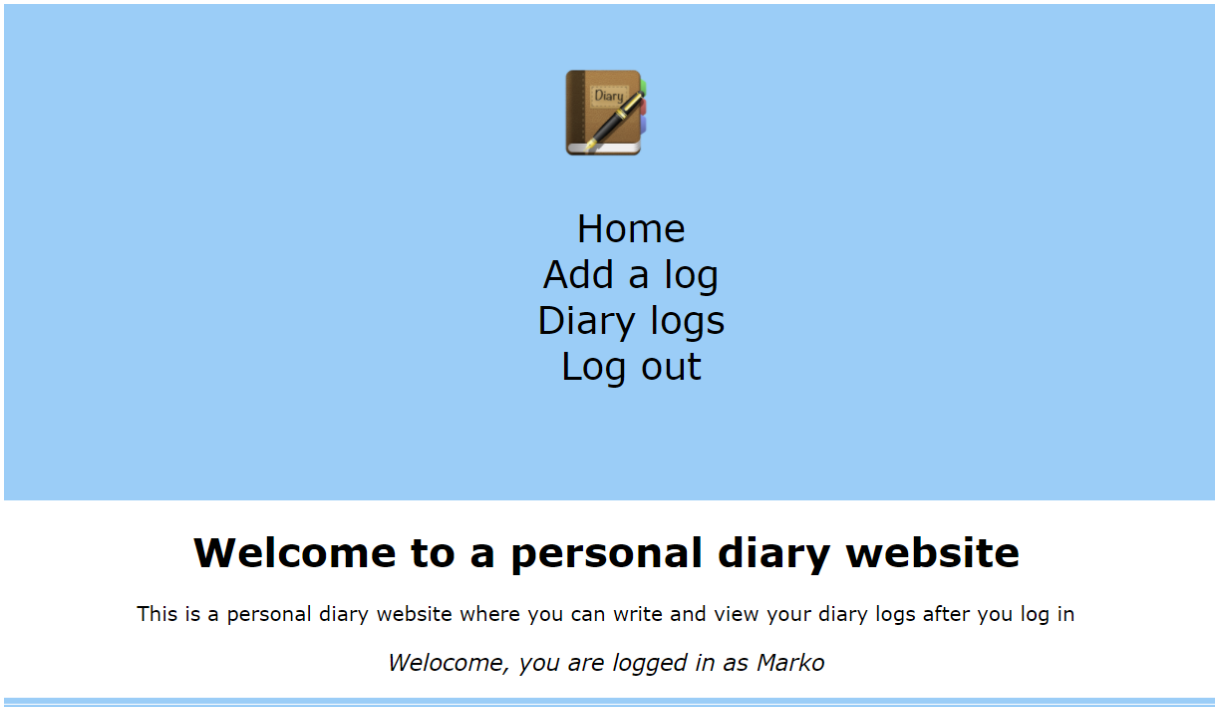
Slika 3. Grubi dizajn log in sustava



Izvor: izradio autor

Kako možemo vidjeti na slici 4. dizajn web sjedišta je ispao drugačije i čak jednostavnije.

Slika 4. Završni dizajn web sjedišta



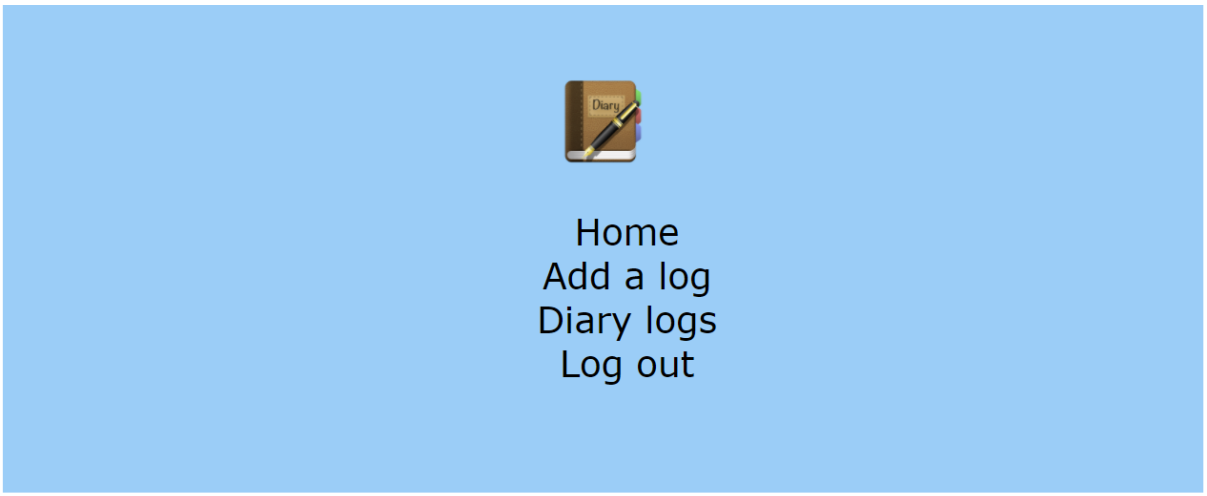
Izvor: izradio autor

Home stranica se sastoji od tipki *Log in* i *Sign up* te od teksta koji se nalazi ispod tipki. *Sign up* stranica se sastoji od četiri kučice za unos teksta i *Sign up* tipke. Prva je unos imena i prezimena, druga za unos korisničkog imena (*username*), treća za unos lozinke i zadnja za potvrdu lozinke. Ako se unese nešto krivo dobijemo odgovarajuću poruku. Ako ne unesemo istu lozinku dobijemo poruku „Passwords are not matching!“, ako nešto ne unesemo dobijemo „Fill in all fields!“ te ako unesemo korisničko ime koje već postoji u bazi podataka dobijemo poruku „Username is taken!“. Kada kreiramo korisnika moramo otići na *Log in* stranicu kako bi se ulogirali. *Log in* stranica se sastoji od dvije kučice za unos teksta i *Log in tipke*. U jednu se unosi korisničko ime (*unsername*) a u drugu se unosi lozinka. Ako unesemo nešto krivo javlja se poruka upozorenja „Wrong login information!“ ispod *Log in* tipke. Kada se ulogirate nestanu *Log in* i *Sign up* tipke i pojave se *Add a log* i *Diary logs* te se na dnu pojavi ime koje ste unijeli kada ste proši kroz *Sign up* stranicu.

Stranica *Add a log* se sastoji od dvije kučice za unos teksta, jedne za odabir datuma i od tipke *Submit Diary Notes*. Prva kučica za unos teksta je naslov za današnji dan i druga je

za pisanje svega što smatrate da se treba napisati u dnevnik. U kučici za odabir datuma samo biramo datum na kalendaru. Ako nešto ne unesemo javlja se poruka „Fill in all fields!“. *Diary logs* se sastoji od tablice koja u sebi ima stupce: naslov, datum i bilješke te od tipka *edit* i *delete*. Bilješke su ono što ste upisivali ranije. Klikom na tipku *edit* dobijemo ekran u kojem možemo promijeniti naslov i bilješke. Klikom na tipku *delete* dobijemo odabir ako sigurno želimo obrisati taj unos u dnevnik sa opcijama da i ne. *Log out* stranica ispisuje korisnika te ga vraća na *Home* stranicu.

Slika 5. Izgled Diary logs stranice



My diaries

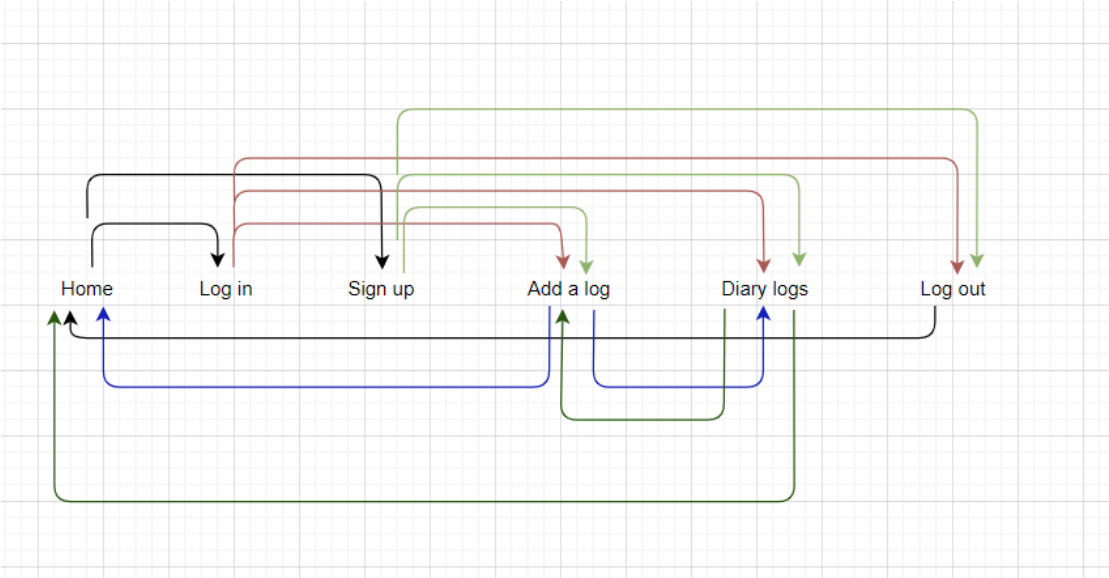
Title	Date	Notes		
Hello	2021-04-14	Danas je dan	EDIT	DELETE
Naslov	2021-04-28	ngdnb	EDIT	DELETE

Izvor: izradio autor

3. STRUKTURA WEB SJEDIŠTA

Web sjedište sastoji se od stranica: Home, Log in, Sign up, Add a log, Diary logs i Log out. Stranicama Add a log, Diary logs i Log out može se pristupiti samo kada je korisnik prijavljen na aplikaciju.

Slika 6. Karta web sjedišta



Izvor: izradio autor

Na karti web sjedišta se vidi odnos stranica. Prije nego što se ulogiramo sa *Home* stranice možemo pristupiti stranicama *Log in* i *Sign up*. Kada napravimo račun ili kada se ulogiramo možemo pristupiti stranicama *Add a log*, u kojoj dodajemo unose u dnevnik, i *Diary logs* u kojoj možemo vidjeti prijašnje unose te ih možemo urediti i obrisati. Pritiskom na tipku *log out* vračamo se na home stranicu te više ne možemo pristupiti stranicama *Add a log* i *Diary logs* dok se ponovo ne ulogiramo.

3.1. Statički i dinamički dijelovi sjedišta

Izrađeno web sjedište sastoji se od statičkih i dinamičkih dijelova. Pod statičkim dijelovima smatraju se svi dijelovi sjedišta koji su izrađeni i ne mogu se mijenjati kao na primjer tekst na stranici. Dinamičkim dijelovima se smatraju oni dijelovi sjedišta u kojima korisnici mogu nešto raditi, tj. mogu unositi, izmjenjivati, dodavati i brisati podatke.

Statički dio stranice je Home stranica u kojoj ne možemo ništa osim preći u druge stranice kao *Log in* i *Sign up.*

Dinamički dijelovi stranice su :

- *Log in* stranica u kojoj možete unositi podatke za login ako ste već u bazi podataka
- *Sign up* stranica u kojoj unosite podatke koji će se dodati u bazu kao što su: ime i prezime, *username* i korisničku lozinku
- *Add a log* stranica u kojoj su unose novi unosi u dnevnik
- *Diary logs* stranica u kojoj se mogu brisati i uređivati prijašnji unosi

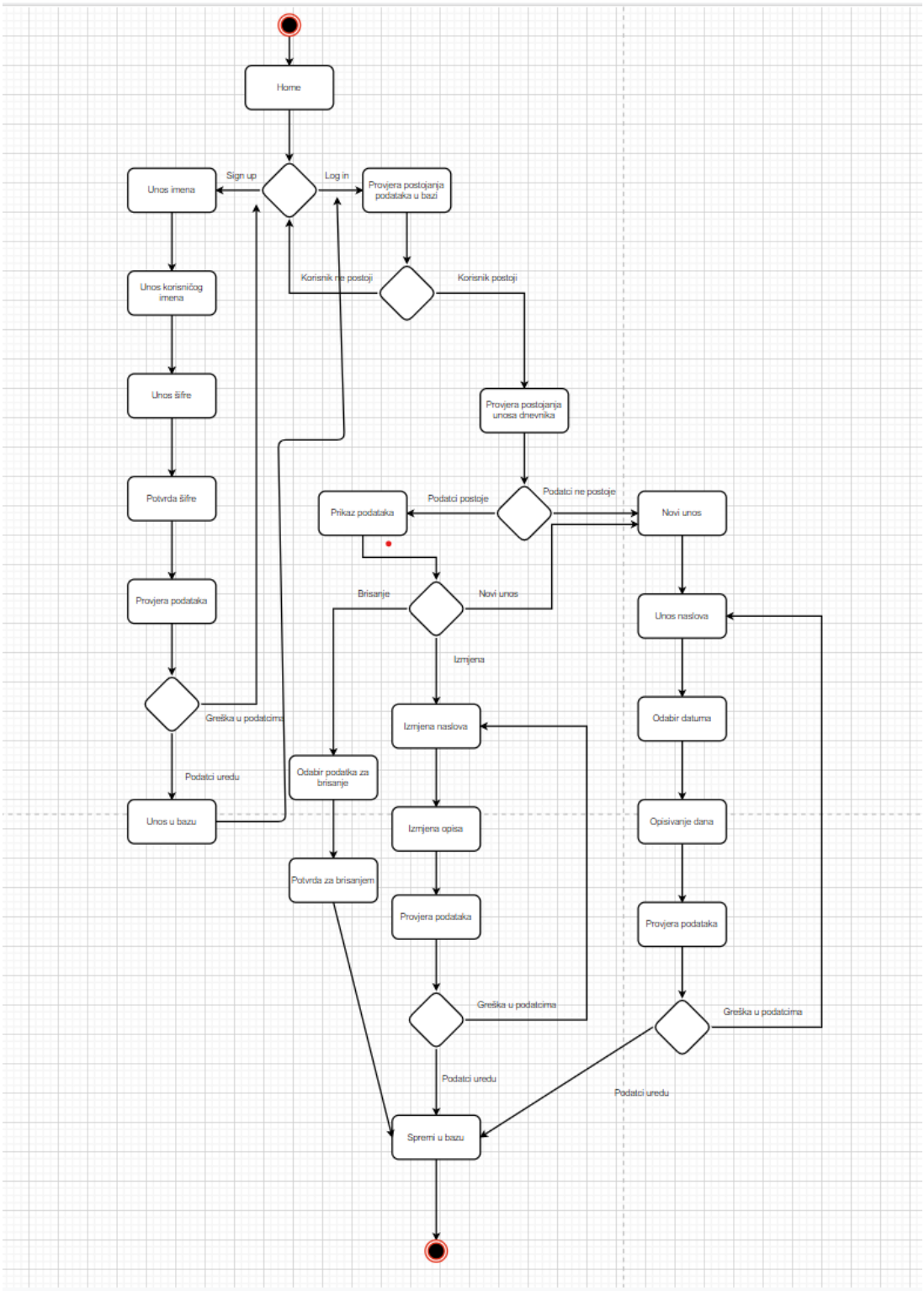
4. IZRADA SUSTAVA

U ovom poglavlju biti će prikazano i opisano dijagram tijeka, modeliranje podataka, izrada baze podataka, izrada aplikacije i dokerizacija te iste aplikacije.

4.1. Dijagram tijeka

Dijagram tijeka prikazuje korake te sva moguća tzv. raskrižja kroz koja korisnik prolazi kada dođe na web aplikaciju.

Slika 7. Dijagram tijeka



Izvor: izradio autor

Kada korisnik pristupi aplikaciji otvori mu se *Home* stranica te on na njoj može birati *Sign up* ili *Log in*. Ako odabere *Sign up* aplikacija ga traži da unese puno ime, korisničko ime, lozinku i da potvrdi lozinku. Ako je sve u redu onda se korisnik vrača na Home stranicu. Ako

nije onda ga se traži da ponovo unese podatke. Ako korisnik odabere *Log in* aplikacija ga traži da unese korisničko ime i lozinku. Ako je sve u redu korisnik se uspješno prijavi na web aplikaciju. Nakon što se uspješno prijavio može unositi nove podatke ili pregledavati stare. U slučaju da korisnik odabere novi unos aplikacija ga traži da unese naslov, odabere datum i unese što želi u dnevnik. Ako je sve u redu unos se sprema u bazu podataka i prikazuje se u unosima. Unosi se mogu brisati i uređivati.

4.2. Modeliranje podataka

Baza podataka se sastoji od dvaju povezanih tablica. Prva tablica je „users“ koja sadrži podatke o korisniku a druga je „diaries“ koja sadrži podatke o unosima u dnevnik.
Tablica „users“ se sastoji od idućih polja:

- userID – koristi se kao identifikacijski broj korisnika te ga dodjeljuje baza podataka (primarni ključ),
- username – korisnici upisuju svoje ime i prezime,
- userUsername – korisnici upisuju svoje korisničko ime te se uz korisničku lozinku koristi za pristup na stranicu,
- userPwd – korisnici unose svoju lozinku te se s njom pristupa stranici.

Tablica „diaries“ se sastoji od idućih polja:

- id – koristi se kao identifikacijski broj unosa u bazu podataka te ga dodjeljuje sama baza (primarni ključ),
- userId – vanjski ključ koji je povezan s tablicom „users“ te se preko njega može pročitati koji korisnik je upisao svoj unos u bazu podataka,
- title – naslov unosa u bazu te ga dodjeljuje korisnik koji upisuje podatke,
- date – datum unosa kojeg korisnik sam može izabrati,
- notes – glavni dio koji korisnik unosi tzv. log, tu korisnik može pisati što god želi.

4.3. Izrada baze podataka

Prilikom izrade baze podataka koristimo se znanjem iz kolegija baze podataka. Tablica „users“ radi se pomoću iduće skripte:

```
CREATE TABLE `users` (
```

```
  `userID` int(5) NOT NULL,
  `username` varchar(128) NOT NULL,
  `userUsername` varchar(128) NOT NULL,
  `userPwd` varchar(128) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

ALTER TABLE `users`
  ADD PRIMARY KEY (`userID`);

ALTER TABLE `users`
  MODIFY `userID` int(5) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=4;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

U prvom odlomku smo stvorili tablicu s podatcima te smo definirali sva polja i koje strukture će biti ta polja. „userID“ je tipa *integer* te se sastoji od 5 znamenki tj. sprema se kao peteroznamenkasti broj (ako je „userID“ 1 u bazu se sprema kao 00001). „username“ je tipa *varchar* te se može sastojati od maksimalno 128 znakova. „userUsername“ je također tipa *varchar* i isto može imati 128 znakova maksimalno. „userPwd“ je također tipa *varchar* i također može imati 128 maksimalno znakova. U drugom odlomku smo dodali primarni ključ na polje „userID“. U trećem odlomku smo dodali auto inkrement na polje „userID“ te smo u programu napravili zadnje tri linije koje onemogućavaju korisnicima koji posjete bazu podataka da vide lozinke od drugih ljudi već vide nešto kao: $2y$10$c8nxqKtt6JNzusntJkh0JecYDJaPb3N13Ti463y9srmjQcqt2Jigi. To služi u slučaju da se dogodi pokušaj krađe podataka sa baze kako bi lozinke ostale sigurne.

Tablicu „diaries“ radimo po idućoj skripti:

```
 CREATE TABLE `diaries` (
  `id` int(100) NOT NULL,
  `userId` int(100) NOT NULL,
  `title` varchar(250) NOT NULL,
  `date` date NOT NULL,
  `notes` varchar(500) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
ALTER TABLE `diaries`
  ADD PRIMARY KEY (`id`);

ALTER TABLE `diaries`
  MODIFY `id` int(100) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
```

U prvom odlomku stvaramo tablicu te definiramo kojeg tipa će biti polja u njoj. „id“ je tipa *integer* te se sastoji od 100 znamenki. „userId“ je tipa *integer* te se sastoji od 100 znamenki te je on poveznica sa bazom „users“. „title“ je naslov unosa te se on može sastojati od maksimalno 250 znakova. „date“ je datum i „notes“ je tekst koji unosimo te se on može sastojati od maksimalno 500 znakova. U drugom odlomku dodajemo primarni ključ na polje „id“ te u trećem auto inkrementaciju kako bi se polje „id“ automatski ispunjavalo.

Bazu podataka implementiramo u program pomoći datoteke zvane „dbh.inc“ te se ona sastoji od idućeg koda:

```
<?php

$serverName = "ucka.veleri.hr";
$dbUsername = "fvasiljevic";
$dbPassword = "11";
$dbName = "fvasiljevic";

$conn = mysqli_connect($serverName, $dbUsername, $dbPassword, $dbName);

if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
```

Datotka je pisana u PHP-u te se zato na početku otvara PHP dio. „serverName“ je naziv servera, u ovom slučaju koristimo ucka.veleri.hr, u kojem se nalazi baza podataka. „dbUsername“ je korisničko ime na koje se spajamo na serveru. „dbPassword“ je lozinka za pristup bazi i „dbName“ je naziv baze podataka koju ćemo koristiti unutar servera. Preko „conn“ ćemo pristupiti ugrađenoj funkciji u PHP-u a to je mysqli_connect koja se korisit za povezivanje s bazom podataka, te ćemo njoj dati parametre koje smo prijašnje definirali. U

slučaju da konekcija ne upije gasimo proceduru i funkcija mysqli_connect_error() nam ispisuje koji je bio problem za uspostavu veze s bazom.

4.4. Izrada aplikacije

Aplikacija će se sastojati od stila, slike i PHP datoteka.

Index datoteka je početna datoteka koje će se prva otvoriti kada otvaramo aplikaciju u web preglednike. Ona se sastoji od gornjeg djela PHP koda, HTML-a i donjeg djela PHP koda. Gornji dio PHP koda služi, kao i na svim drugim PHP datotekama, kako bi učitali *header* datoteku tj. da ne moramo svaki put ponovo pisati uzglavlje stranice. U HTML djelu se nalazi tekst na stranici. Donji dio PHP-a se sastoji od dvije oznake PHP-a. One služe da nam u slučaju da korisnik nije ulogiran pokazuje tipke na dnu stranice, iste kao i na uzglavlju a u slučaju da je ulogiran na dnu stranice piše: Dobrodošli, ulogirani ste kao %%; gdje %% ispisuje ime korisnika koji se ulogirao.

Header datoteka se sastoji od uvodnog PHP djela koji služi za pokretanje sesije tj. za povezivanje s bazom podatka, HTML djela koji služi za prikazivanje tipki i slike na stranici. Donji dio PHP-a služi za funkcionalnost tipki. U slučaju da korisnik nije ulogiran prikazuju se tipke *Home*, *Log in* i *Sign up*. U slučaju da je korisnik ulogiran prikazuju se tipke *Home, Add a log*, *Diary logs* i *Log out* (kod prikazan ispod).

```
<?php
   if (isset($_SESSION["userName"])) {
   echo "<li><a href='diary.php'>Add a log</a></li>";
   echo "<li><a href='diaries.php'>Diary logs</a></li>";
   echo "<li><a href='includes/logout.inc.php'>Log out</a></li>";
   }
   else {
   echo "<li><a href='login.php'>Log in</a></li>";
   echo "<li><a href='signup.php'>Sign up</a></li>";
   }
?>
```

Prva linija koda nakon otvaranja PHP oznaka ima funkcionalnost da utvrdi ako je korisnik ulogiran tj. ako na sesiji imamo korisničko ime. Ako ga imamo onda su tipke *Home, Add a*

log, Diary logs i *Log out.* U slučaju da ga nemamo tj. ako nitko nije ulogiran onda su tipke *Home, Log in* i *Sign up.*

Login datoteka se također sastoji od dviju oznaka PHP-a i od HTML-a u sredini tj. druga PHP oznaka se nalazi unutar HTML oznake. Funkcija prve PHP oznake je da uključi *header* datoteku te u slučaju da smo ulogirani na stranicu da nas vrati na index datoteku tj. da ne možemo pristupiti login datoteci. HTML oznake definiraju kučice za pisanje korisničkog imena i lozinke i tipku *Log in*. U slučaju da je sve dobro ispunjeno tipka *Log in* poziva *login.inc.php* datoteku. PHP oznake unutar HTML-a ima svrhu ispisivanja grešaka. U slučaju da je greška „*emptyinput*“ tj. ako nismo sve ispunili dobijemo poruku da ispunimo sva polja (*Fill in all fields!*) a u slučaju krivog login-a („*wronglogin*“) tj. ako korisnik ne postoji dobijemo poruku krivi podatci za prijavu („Wrong login information!“).

```
if (isset($_GET["error"])) {
            if ($_GET["error"] == "emptyinput") {
                echo "<p>Fill in all fields!</p>";
            } else if ($_GET["error"] == "wronglogin") {
                echo "<p>Wrong login information!</p>";
            }
        }
```

Signup datoteka se sastoji od PHP oznake i od HTML oznake u kojoj se nalazi još jedna PHP oznaka. Gornja PHP oznaka ima svrhu uključivanja *header* datoteke. HTML oznake definiraju kučice za pisanje, imena, korisničkog imena, lozinke i potvrda lozinke, i tipku *Sign up*. Ako je sve u redu onda se pritiskom na tipku *Sign up* poziva datoteka *signup.inc.php*. PHP oznake unutar HTML oznaka ima ulogu *error handler*-a.

- greška: „*emptyinput*“ – ispis: „ispunite sva polja!“ („*Fill in all fields!*“),
- greška: „*invalidusername*“ – ispis: „Odaberite korisničko ime koje sadržava slova od A do Z i brojeve od 0 do 9!“ („*Choose a username that contains letters A-Z and numbers 0-9!*“),
- greška: „*passwordsdontmatch*“ – ispis: „Lozinke se ne podudaraju!“ („*Passwords are not matching!*“),
- greška „*usernametaken*“ – ispis: „Korisničko ime je zauzeto!“ („*Username is taken!*“),
- greška „*stmtfailed*“ – ispis: „Nešto je pošlo po zlu, pokušajte ponovo!“ („*Something went wrong, try again!*“),

- greška „*none*“ (nema greške) – ispis: „Prijavili ste se!“ („*You have signed up!*“).

```
if (isset($_GET["error"])) {
            if ($_GET["error"] == "emptyinput") {
                echo "<p>Fill in all fields!</p>";
            } else if ($_GET["error"] == "invalidusername") {
                echo "<p>Choose a username that contains letters A-
Z and numbers 0-9!</p>";
            } else if ($_GET["error"] == "passwordsdontmatch") {
                echo "<p>Passwords are not matching!</p>";
            } else if ($_GET["error"] == "usernametaken") {
                echo "<p>Username is taken!</p>";
            } else if ($_GET["error"] == "stmtfailed") {
                echo "<p>Something went wrong, try again!</p>";
            } else if ($_GET["error"] == "none") {
                echo "<p>You have signed up!</p>";
            }
        }
```

Diary datoteka se sastoji od PHP skripte koja uključuje *header* datoteku, od HTML djela u kojem se definiraju kučice, za pisanje naslova i teksta,, datuma i tipke za spremanje u bazu podataka. U datumu se otvaraju PHP oznake kako bi se automatski odabrao današnji datum:

`<?php echo date("Y-m-d"); ?>.` Također u HTML-u se nalaze PHP oznake za upravljanje greškama:

```
if (isset($_GET["error"])) {
            if ($_GET["error"] == "emptyinput") {
                echo "<p>Fill in all fields!</p>";
            } else if ($_GET["error"] == "diaryExists") {
                echo "<p>Sorry, you already saved diary for this date!</p>";
            }
```

U slučaju da je greška „*emptyinput*“ tj. ako korisnik nije ispunio sva polja onda se pojavljuje tekst da se treba ispuniti sve („*Fill in all fields!*“). U slučaju da je greška „*diaryExists*“ tj. ako je korisnik već opisao taj dan onda mu se pojavljuje poruka da je taj dan već unesen („*Sorry, you already saved diary for this date!*“).

Diaries datoteka na početku ima PHP oznake u kojima se nalazi linija koda koja uključuje *header* datoteku i koja zahtjeva *dbh.inc* i *functions.inc* (dbh.inc datoteka je za

povezivanje s bazom podataka, opisana u djelu izrada baze podataka, functions.inc datoteka biti će opisana u kasnijem odlomku) datoteke. Također se u oznakama nalazi dodjeljivanje vrijednosti varijablama tj. userID dobiva svoju vrijednost što znači (i što piše u idućoj liniji koda) da korisnik mora biti upisan i uzimanje podataka o upisima u dnevnik iz baze podataka:

```
<?php
include_once 'header.php';
require_once 'includes/dbh.inc.php';
require_once 'includes/functions.inc.php';

$userID = $_SESSION['userid'];
if (!isset($userID)) {
    header('Location: index.php?error=loginRequired');
}
$diaries = fetchUserDiaries($conn, $userID);
?>
```

U HTML dijelu datotke se nalazi definiranje tablice u kojoj se nalaze naslov, datum i bilješke te se oni podatci uzimaju iz baze (prikazano u kodu ispod) i smještaju u tablicu.

```
<table border="1">
                <tr>
                    <th>Title</th>
                    <th>Date</th>
                    <th>Notes</th>
                </tr>
                <?php
                foreach ($diaries as $diary) {
                    echo "<tr>";
                    echo "<td>" . $diary[2] . "</td>";
                    echo "<td>" . $diary[3] . "</td>";
                    echo "<td>" . $diary[4] . "</td>";
                    echo "<td> <a href=diaryActions.php?a=edit&id=" . $diary[0
] . ">EDIT</a></td>";
                    echo "<td> <a href=diaryActions.php?a=del&id=" . $diary[0]
 . ">DELETE</a></td>";
                    echo "</tr>";
                }
                ?>
```

```
            </table>
```

Ispod toga se nalazi kod za greške:

- greška „*emptyinput*“ – ispis: „Ispunite sva polja!“ („*Fill in all fields!*“),
- greška „*diaryExists*“ – ispis: „Unos u drvenik za ovaj datum već postoji!“ („*Sorry, you already saved diary for this date!*“),
- greška za sve ostalo – ispis: „Pojavila se greška!“ („*Sorry, an error occured!*“).

```
if (isset($_GET["error"])) {
            if ($_GET["error"] == "emptyinput") {
                echo "<p>Fill in all fields!</p>";
            } else if ($_GET["error"] == "diaryExists") {
                echo "<p>Sorry, you already saved diary for this date!</p>";
            } else {
                echo "<p>Sorry, an error occured!</p>";
            }
        }
```

Ispod koda za greške nalazi se kod za potvrdu i otkazivanje brisanja.

```
if (isset($_GET["delete"])) {
            if ($_GET["delete"] == "canc") {
                echo "<p>Delete canceled!</p>";
            } else if ($_GET["delete"] == "succ") {
                echo "<p>Delete successful</p>";
            }
        }
```

Slika 8. Prikaz tablice

My diaries

Title	Date	Notes		
Hello	2021-04-14	Danas je dan	EDIT	DELETE
Naslov	2021-04-28	ngdnb	EDIT	DELETE

Izradio: izradio autor

Datoteka *diaryActions* se sastoji od gornjih PHP oznaka, HTML oznaka i PHP oznaka unutar HTML-a. U gornjoj PHP oznaci se nalazi zahtjev da se uključi datoteka *header* i potreba (*require*) za dbh.inc i functions.inc datotekama. Također se nalazi dodjeljivanje vrijednosti varijabli *id* sa vrijednostima iz baze podataka, također iz stupca *id* i povlačenje odabranog unosa u dnevnik. HTML dio opisuje izgled uređivanja odabranog unosa ili brisanja odabranog unosa. U svakom se nalaze PHP oznake koje imaju svrhu ili povlačenja nekog podatka iz baze podataka ili upravljanje pogreškama. Dolje je prikazan gornji PHP dio.

```
<?php
include_once 'header.php';

require_once 'includes/dbh.inc.php';
require_once 'includes/functions.inc.php';

$id = $_GET['id'];
$diary = fetchDiary($conn, $id);

if (isset($_GET)) {
    if ($_GET['a'] == "edit") {
?>
```

Datoteke sa .inc nastavkom su datoteke koje u sebi sadržavaju samo PHP kod te se preko njih obavlja većina funkcija. Includes datoteka se sastoji od *dbh.inc.php, deleteDiary.inc.php, diary.inc.php, editDiary.inc.php, functions.inc.php, login.inc.php, logout.inc.php, signup.inc.php* datoteka te će biti opisane sve koje već nisu.

Datoteka deleteDiary.inc ima svrhu brisanja unosa koji je zapisan u bazu podataka. On zahtjeva datoteku dbh.inc.

```
<?php
require_once 'dbh.inc.php';

if (isset($_POST["submit"])) {
    if ($_POST['confirm'] == "no") {
        header("location: ../diaries.php?delete=canc");
        exit();
    }
    $sql = "DELETE FROM diaries WHERE id = ?;";
```

```
    $stmt = mysqli_stmt_init($conn);
    if (!mysqli_stmt_prepare($stmt, $sql)) {
        header("location: ../diaries.php?error=stmtfailed");
        echo "errot";
        exit();
    }
    mysqli_stmt_bind_param($stmt, "s", $_POST['id']);
    mysqli_stmt_execute($stmt);

    $resultData = mysqli_stmt_get_result($stmt);

    if ($row = mysqli_fetch_assoc($resultData)) {
        header("location: ../diaries.php?delete=succ");
        exit();
    } else {
        header("location: ../diaries.php?delete=succ");
        exit();
    }
    mysqli_stmt_close($stmt);
}
```

Na početku imamo prvi *if statement* koji nam govori da je korisnik odabrao da želi brisati datoteku. Odmah ispod imamo *if statement* u kojem je korisnik ipak odabrao da ne želi obrisati taj unos te se izlazi iz funkcije. Nakon toga definiramo varijablu sql kojoj dodjeljujemo vrijednost teksta za brisanje podataka iz baze podataka (DELETE FROM diaries WHERE id = ?). *Diaries* je tablica u bazi podataka koja sadržava unose i na kraju te izjave biti će dodijeljen *id* od unosa kojeg smo odabrali i želimo da obrisati. Varijabli stmt dodjeljujemo vrijednost funkcije mysqli_stmt_init($conn). U idućem *if statementu* provjeravamo ako su spremne i jedna i druga varijabla i u slučaju da nisu program će nam izbaciti grešku. Nakon toga se obavlja proces brisanja podataka iz baze i zatvara se funkcija ove datoteke.

Datoteka diary.inc služi kako bi uzela spremila podatke koje smo upisali u bazu podataka.

```
<?php

if (isset($_POST["submit"])) {
```

```
    $title = $_POST["title"];
    $date = $_POST["date"];
    $notes = $_POST["notes"];

    session_start();
    $userID = $_SESSION['userid'];
    if (!isset($userID)) {
        header('Location: index.php?error=loginRequired');
    }

    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';

    if (emptyDiary($title, $date, $notes) !== false) {
        header("location: ../diary.php?error=emptyinput");
        exit();
    }
    saveDiary($conn, $userID, $title, $date, $notes);
} else {
    header("location: ../login.php");
    exit();
}
```

Na vrhu vidimo if statement u kojem piše da ako je korisnik pritisnuo tipku „submit“ onda varijable title, date i notes dobivaju vrijednosti koje smo mi upisali u odgovarajuće kučice i onda se program veže za bazu podataka. U slučaju da korisnik nije ulogiran program izbacuje grešku te se korisnik mora ulogirati. Program onda traži funkcije datoteka dbh.inc i functions.inc. U slučaju da nešto nije upisano onda na izbacuje grešku „*emptyinput*“ a u slučaju da je sve u redu onda sprema unos u dnevnik na bazi podataka. Krajnja *else* izjava nam govori da se vratimo na stranicu login.

Datoteka editDiary.inc služi kako bi se ažurirali postojeći unosi u dnevnik.

```
<?php

if (isset($_POST["submit"])) {
```

```
    $id = $_POST["id"];
    $title = $_POST["title"];
    $notes = $_POST["notes"];

    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';

    if (emptyDiary($title, 'update', $notes) !== false) {

        header("location: ../diaryActions.php?a=edit&id=". $id ."error=emptyin
put");
        exit();
    }
    updateDiary($conn, $id, $title, $notes);
    header("location: ../diaries.php");
    exit();
} else {
    header("location: ../login.php");
    exit();
}
```

Na početku imamo uvjet ako je korisnik pritisnuo tipku onda se uzimaju vrijednosti id (korisnika koji unosi podatke u dnevnik), naslov unosa podataka u dnevnik i sam unos podataka te se oni dodjeljuju varijablama istog imena. Onda skripta zahtjeva datoteke dbh.inc kako bi se povezala na bazu podataka i functions.inc kako bi koristila funkcije. U slučaju da korisnik koji želi ažurirati nešto a nema ništa upisano ili nema neko polje upisano onda se javlja greška „*emptyinput*“. U slučaju da je sve u redu onda se dnevnik ažurira i korisnika vraća na stranicu na kojoj se pokazuju svi unosi u dnevnik (*diaries*). Na kraju skripte vidimo da ako korisnik nije pritisnuo tipku onda ga se vraća na login stranicu.

Datoteka login.inc prijavljuje korisnika na web aplikaciju.

```
<?php

if (isset($_POST["submit"])) {

    $username = $_POST["username"];
    $password = $_POST["password"];
```

```
    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';

    if (emptyInputLogin($username, $password) !== false) {
        header("location: ../login.php?error=emptyinput");
        exit();
    }

    loginUser($conn, $username, $password);
} else {
    header("location: ../login.php");
    exit();
}
```

U slučaju da je korisnik pritisnuo tipku vrijednosti koje je upisao u kučice se dodjeljuju varijablama istog imena te se onda zahtijevaju datoteke dbh.inc i functions.inc. U slučaju da nešto nije ispisano web aplikacija daje grešku „*emptyinput*“, inače se korisnik upisuje. Na kraju vidimo da ako korisnik nije pritisnuo tipku onda ostaje na stranici.

Datoteka logout.inc je najjednostavnija datoteka te ona služi za ispisivanje korisnika sa web aplikacije.

```
<?php

session_start();
session_unset();
session_destroy();

header("location: ../index.php");
exit();
```

Datoteka logout.inc to radi tako da pokrene sesiju, *unseta* ju te ju onda uništi. Korisnika onda vrati na index stranicu i završi svoj posao.

Signup.inc datoteka upisuje korisnika na web aplikaciju po prvi put.

```
<?php

if (isset($_POST["submit"])) {
```

```
    $name = $_POST["name"];
    $username = $_POST["username"];
    $password = $_POST["password"];
    $confirmpassword = $_POST["confirmpassword"];

    require_once 'dbh.inc.php';
    require_once 'functions.inc.php';

    if (emptyInputSignup($name, $username, $password, $confirmpassword) !== fa
lse) {
        header("location: ../signup.php?error=emptyinput");
        exit();
    }
    if (invalidUsername($username) !== false) {
        header("location: ../signup.php?error=invalidusername");
        exit();
    }
    if (passwordMatch($password, $confirmpassword) !== false) {
        header("location: ../signup.php?error=passwordsdontmatch");
        exit();
    }
    if (usernameExists($conn, $username) !== false) {
        header("location: ../signup.php?error=usernametaken");
        exit();
    }
    createUser($conn, $name, $username, $password);
} else {
    header("location: ../signup.php");
    exit();
}
```

U slučaju da je korisnik pritisnuo tipku onda se vrijednosti iz kučica dodjeljuju varijablama istog imena te se traže datoteke dbh.inc i functions.inc. Nakon toga se proilazi kroz sve moguće scenaije i provjeravaju se sve moguće greške:

- U slučaju da nešto nije upisano je greška „*emptyinput*“,
- U slučaju da nešto nije u redu sa korisničkim imenom je greška „*invalidusername*“,

- U slučaju da se lozinke ne preklapaju je greška „*passwordsdontmatch*“,
- U slučaju da je korisničko ime već postojeće u bazi podataka je greška „*usernametaken*“.

Ako nema grešaka onda se korisnika ubacuje u bazu podataka.

Datoteka functions.inc ima u sebi sadržava sve funkcije koje program koristi. Funkcija „*emptyInputSignup*“ definira novu varijablu *result* te provjerava ako je bilo koja od varijabli poslana njoj prazna (npr. korisničko ime može biti prazno). U slučaju da je neka varijabla prazna tj. bez dodijeljene vrijednosti varijabla *result* prima vrijednost da (true) a ako su sve varijable sa svojim vrijednostima onda prima vrijednost ne (false). Na kraju procesa ta varijabla se vraća u program.

```
function emptyInputSignup($name, $username, $password, $confirmpassword)
{
    $result;
    if (empty($name) || empty($username) || empty($password) || empty($confirm
password)) {
        $result = true;
    } else {
        $result = false;
    }
    return $result;
}
```

Funkcija „*invalidUsername*“ definira novu varijablu *result* te provjerava ako je korisničko ime validno. Korisničko ime je validno ako sadržava slova „A-Z“, „a-z“ i brojeve „0-9“. U slučaju da korisničko ime ima neke druge znakove varijabla *result* prima vrijednost da (true) a u slučaju da korisničko ime sadržava samo definirane znakove onda prima vrijednost ne (false). Na kraju funkcije varijabla *result* se vrača u glavni dio programa.

```
function invalidUsername($username)
{
    $result;
    if (!preg_match("/^[a-zA-Z0-9]*$/", $username)) {
        $result = true;
    } else {
        $result = false;
```

```
    }
    return $result;
}
```

Funkcija „*passwordMatch*“ također definira novu varijablu *result* te provjerava ako su lozinke jednake. Provjerava se ako su jednaka polja *password* i *confirmpassword*. U slučaju da su različiti *result* prima vrijednost da (true) a u slučaju da su jednaki prima vrijednost ne (false). Na kraju funkcije se vrača u glavni program.

```
function passwordMatch($password, $confirmpassword)
{
    $result;
    if ($password !== $confirmpassword) {
        $result = true;
    } else {
        $result = false;
    }
    return $result;
}
```

Funkcija „*usernameExists*“ definira dvije nove varijable, *sql* i *stmt*, te provjerava ako korisničko ime postoji u bazi podataka. Varijabla *sql* prima vrijednost „SELECT * FROM users WHERE userUsername = ?;“. Pomoću nje će se proći kroz bazu podataka i vidjeti ako postoji to ime već. Varijabla *stmt* prima vrijednost „mysqli_stmt_init($conn)“ što nam govori da je to varijabla koja će započeti konekciju sa bazom podataka. U slučaju da se iz nekog razloga ne može nešto od toga izvršiti program izbacuje grešku „*stmtfailed*“. Ako je sve u redu šalje se korisničko ime u bazu podataka i ako korisničko ime već postoji program nam kaže da već postoji a ako ne se upisuje u bazu podataka.

```
function usernameExists($conn, $username)
{
    $sql = "SELECT * FROM users WHERE userUsername = ?;";
    $stmt = mysqli_stmt_init($conn);
    if (!mysqli_stmt_prepare($stmt, $sql)) {
        header("location: ../signup.php?error=stmtfailed");
        exit();
    }

    mysqli_stmt_bind_param($stmt, "s", $username);
```

```
    mysqli_stmt_execute($stmt);

    $resultData = mysqli_stmt_get_result($stmt);

    if ($row = mysqli_fetch_assoc($resultData)) {
        return $row;
    } else {
        $result = false;
        return $result;
    }

    mysqli_stmt_close($stmt);
}
```

Funkcija „*createUser*“ definira dvije nove varijable *sql* i *stmt*. Funkciji se šalju parametri *conn, name, username* i *password*. Varijabla *sql* prima vrijednost „INSERT INTO users (username, userUsername, userPwd) VALUES (?, ?, ?);“ a varijabla *stmt* prima vrijednost inicijacije veze s bazom podataka. U slučaju da nešto nije u redu sa varijablama stmt i sql izbacuje se greška „*stmtfailed*“. Nakon toga definirana je još jedna varijabla hashedPwd koja maskira korisničku lozinku tako da u slučaju pokušaja krađe podataka iz baze lozinke su sigurne. Nakon toga se podatci ubacuju u bazu podataka, prekida se veza i ne vraća se nikakva greška.

```
function createUser($conn, $name, $username, $password)
{
    $sql = "INSERT INTO users (username, userUsername, userPwd) VALUES (?, ?,
?);";
    $stmt = mysqli_stmt_init($conn);
    if (!mysqli_stmt_prepare($stmt, $sql)) {
        header("location: ../signup.php?error=stmtfailed");
        exit();
    }

    $hashedPwd = password_hash($password, PASSWORD_DEFAULT);

    mysqli_stmt_bind_param($stmt, "sss", $name, $username, $hashedPwd);
    mysqli_stmt_execute($stmt);
```

```
    mysqli_stmt_close($stmt);

    header("location: ../signup.php?error=none");
    exit();
}
```

Ostale funkcije su:

- Funkcija „*emptyInputLogin*“ provjerava ako su sva polja u loginu ispunjena
- Funkcija „*loginUser*“ logira korisnika na web aplikaciju
- Funkcija „*emptyDiary*“ provjerava ako je korisnik ispunio sva polja kada piše dnevnik
- Funkcija „*diaryExists*“ provjerava ako postoji unos u dnevnik od tog korisnika za taj datum
- Funkcija „*saveDiary*“ sprema unos u dnevnik u bazu podataka
- Funkcija „*updateDiary*“ ažurira unos u dnevnik tj. sprema izmjene za taj dan
- Funkcija „*fetchDiary*“ uzima određeni unos u dnevnik kako bi mogao biti ažuriran pomoću funkcije „*updateDiary*“
- Funkcija „*fetchUserDiaries*“ uzima sve unose od određenog korisnika kako bi mogli biti prikazani u tablici na web aplikaciji

4.5. Izrada Nginx spremnika

Početi ćemo s izradom web servera te ćemo za to iskoristiti službenu Nginx sliku. Pošto ćemo za konfiguraciju spremnika koristiti docker-compose.yml datoteku, stvoriti ćemo je i u nju napisati:

```
version: '2'
services:
    web:
        image: nginx:latest
        ports:
            - "8080:80"
```

Te ćemo ju pokrenuti pomoću docker-compose up. Sada ako pokrenemo localhost:8080 (jer smo za port napisali da je 8080) na internet pregledniku trebali bi dobiti zadani Nginx ekran.

Slika 9. Nginx zadani ekran

Izvor: izradio autor

Sada kada je server napravljen na njega moramo dodati aplikaciju i kod. Upisujemo docker-compose down kako bi se zatvorili svi kontejneri i kako bi dalje mogli ažurirati docker-compose.yml datoteku. Prvi korak je ažuriranje docker-compose.yml datoteke. U nju treba dodati lokaciju lokalnog direktorija. Stvaramo folder te ažuriramo docker-compose.yml da sada izgleda ovako:

```
version: '2'
services:
    web:
        image: nginx:latest
        ports:
            - "8080:80"
        volumes:
            - ./code:/code
```

Sljedeći korak je obavijestiti Nginx da taj folder postoji. Za to stvaramo *site.conf* konfiguracijsku datoteku te u njegu stavljamo:

```
server {
    listen 80;
    index index.html;
    server_name localhost;
    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root /code;
}
```

index.html je zadana stranica na kojoj se otvara aplikacija. localhost je naziv servera te ćemo njemu pristupiti ako upišemo naziv u internet preglednik. error_log i access_log definiramo

datoteke u koje će se upisivati greške i pristupi serveru. Root je lokacija u kojoj se nalazi kod i indeks.html početna stranica. Ponovno moramo ažurirati docker-compose.yml datoteku kako bi mogli aktivirati konfiguracijsku datoteku:

```
version: '2'
services:
    web:
        image: nginx:latest
        ports:
            - "8080:80"
        volumes:
            - ./code:/code
            - ./site.conf:/etc/nginx/conf.d/default.conf
```

Ta linija koda će dodati *site.conf* datoteku u direktorij kamo Nginx traži konfiguracijske datoteke. Sada ubacujemo kod od web aplikacije u code direktorij i ponovo upisujemo docker-compose up. Kako je naša aplikacija u PHP-u ako pristupimo localhostu nećemo dobiti ništa osim Nginx zadanog ekrana. Kako bi njoj mogli pristupiti moramo dodati PHP-FPM.

4.6. Izrada PHP-FPM spremnika

PHP-FPM (*FastCGI Process Manager*) je PHP implementacija sa dodanim značajkama korisnim za sve vrste stranica te se on koristi u ovom radu. (What is PHP-FPM?, 2011)

Sada kada imamo Nginx spremnik dodati ćemo PHP spremnik kako bi mogli učitati našu aplikaciju. Prvi korak je ažuriranje docker-compose.yml datoteke.

```
version: '2'
services:
    web:
        image: nginx:latest
        ports:
            - "8080:80"
        volumes:
            - ./code:/code
            - ./site.conf:/etc/nginx/conf.d/default.conf
```

```
        networks:
            - code-network
    php:
        build: .
        volumes:
            - ./code:/code
            - ./log.conf:/usr/local/etc/php-fpm.d/zz-log.conf
        networks:
            - code-network

networks:
    code-network:
        driver: bridge
```

Pošto smo u prvoj liniji koda ispod PHP-a napisali build umjesto image moramo napraviti dockerfile datoteku u kojoj ćemo definirati koju PHP sliku želimo i koje ekstenzije želimo:

```
FROM php:fpm

RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
```

Iz slike PHP:FPM dodajemo ekstenziju *mysqli* i omogućavamo njezino korištenje. Bez tih ekstenzija PHP neće razumjeti kod koji koristimo za povezivanje na bazu podataka (kod u datoteci dbh.inc) te će bacati grešku (*HTML error 500*). U ovom radu se to dogodilo i to je na kraju bio najveći problem. Na kraju je sve riješeno pomoću konzole od docker-a u kojoj je došla obavijest: „PHP message: PHP Fatal error: Uncaught Error: Call to undefined function mysqli_connect()“. Pomoću te obavijesti se otkrio uzrok problema i uspješno se popravio problem.

Dodali smo službenu PHP sliku te smo joj rekli da se kod nalazi u folderu *code*. Također smo dodali mrežni driver *bridge* koji će povezivati naše apliakacije putem *bridge* modale. Kako bi Nginx razumio PHP kod moramo ažurirati *site.conf* konfiguracijsku datoteku:

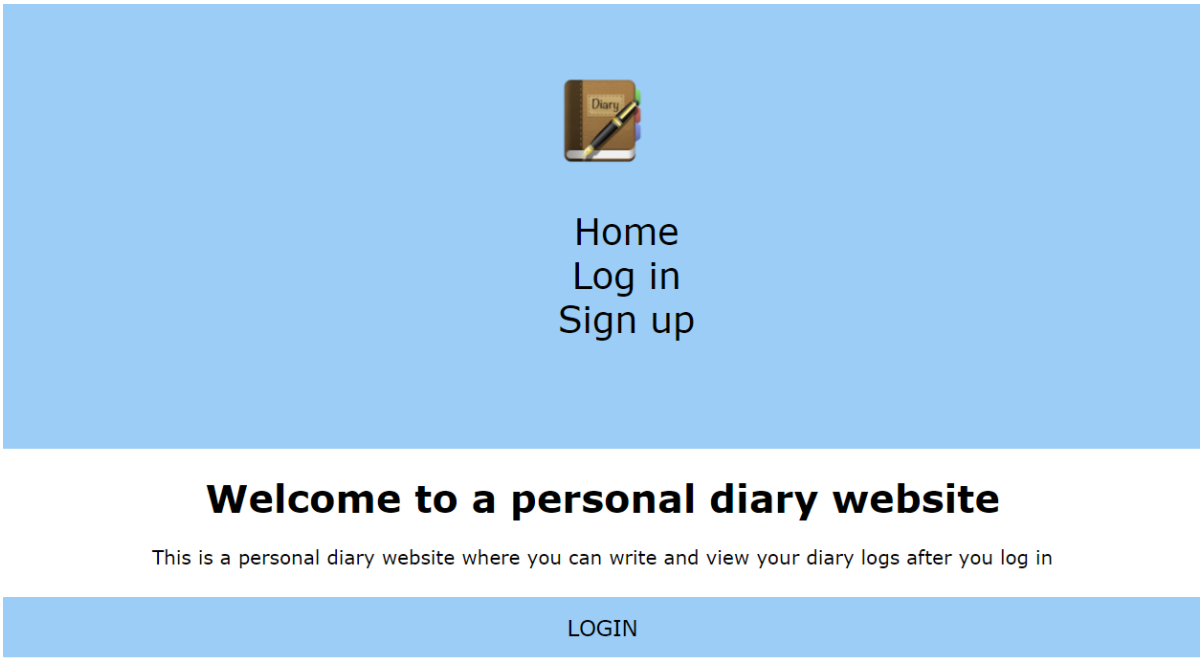
```
server {
    listen 80;
    index index.php index.html;
    server_name localhost;
```

```
    error_log  /var/log/nginx/error.log;
    access_log /var/log/nginx/access.log;
    root /code;

    location ~ \.php$ {
        try_files $uri =404;
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
    }
}
```

Kako je aplikacija samo u PHP-u možemo izbrisati indeks.html iz trećeg reda koda. Sada kada napravimo docker-compose up i pristupimo localhost:8080 vrlo vjerojatno ćemo ponovo dobiti Nginx zadani ekran (slika 2.) ali ako pristupimo localhost:8080/indeks.php onda dobivamo našu stranicu. To se dešava prvih nekoliko puta i kasnije će nas automatski usmjeriti na *indeks.php* stranicu.

Slika 10. localhost:8080/index.php



Izvor: izradio autor

5. STAVLJANJE APLIKACIJE NA WEB PUTEM AWS-A

Ovaj dio rada može se odraditi na nekoliko načina:

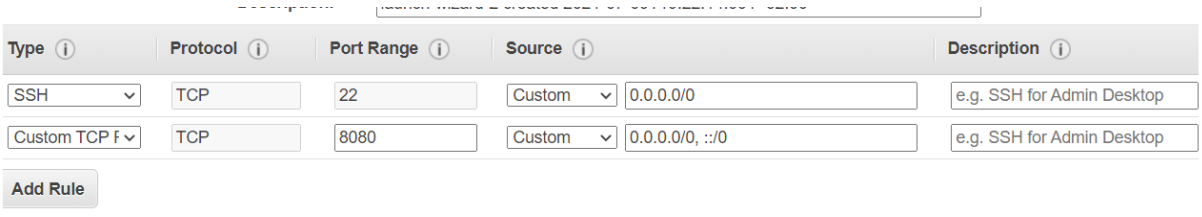
- Putem AWS CLI-ja. Na računalu se izrade kontejneri sa aplikacijom u njima i preko AWS CLI-ja se prebace na AWS,
- Putem ECS-CLI-ja. Docker se prebaci na „context“ od ECS-a te se preko njega napravi „docker compose up“ i aplikacija se onda napravi na ECS-u,
- Putem EC2 instanca na AWS-u. Napravi se instanca linux-a te se na nju instalira docker i preko toga se napravi „docker-compose up“.

Za potrebe ovog rada koristiti će se zadnja opcija.

5.1. Izrada instance Linux-a

Linux sustav se izrađuje putem EC2 usluge na AWS-u. Kada se ulogirate na AWS navigira se do usluge EC2 te se tamo odabire opcija „Launch instance“. AWS nam onda ponudi nekoliko opcija od kojih mi odabiremo Amazon Linux 2 AMI. Za tip instance odabiremo t2.micro jer je ona besplatna i ne moramo plaćati za nju a za potrebe ovog završnog rada je više nego dovoljna. Na detaljima instance sve puštamo na zadanim postavkama i nastavljamo dalje. Na memoriji također sve puštamo zadano i nastavljamo na tagove koje preskačemo. Na sigurnosnim grupama moramo dodati nekoliko pravila. Moramo dodati pravilo da na port 8080 mogu pristupiti sve v4 i v6 IP adrese. Nakon toga pokrenemo instancu te dobijemo opciju za nazvati ključeve od instance kako želimo. Ključeve moramo skinuti kako bi mogli pokrenuti instancu. Instanci na kraju možemo dati ime.

Slika 11. Pravila za pristup instanci



Izvor: izradio autor

5.2. Docker i datoteke na instanci

Novo napravljena instanca na sebi nema niti Docker instaliran niti datoteke preko kojih će biti pokrenuta web aplikacije. Kako bi instalirali Docker prvo se moramo spojiti na instancu. To radimo da odaberemo instancu i pritisnemo na tipku „connect“. AWS će nam dati ID instance, javnu IP adresu i korisničko ime. Ništa od navedenih stavki ne mijenjamo nego samo odaberemo tipku „connect“. AWS će otvoriti našu instancu te na njoj mi moramo prvo ažurirati sve pakete koje ona koristi za rad. To radimo tako da unesemo komandu: `sudo yum update -y` . Nakon što Linux ažurira sve što treba upisujemo komandu: `sudo amazon-linux-extras install docker` kako bi se instalirao Docker. Nakon toga pokrećemo Docker komandom: `sudo service docker start`. Korisniku imena ec2-user (tako nam je dodijeljeno ime pri spajanju s instancom) moramo dodjeliti ovlasti za rad s Docker-om a to radimo komandom: `sudo usermod -a -G docker ec2-user` . Nakon toga moramo zatvoriti instancu i ponovo ju otvoriti kako bi sve promjene došle do izražaja. Provjeravamo ako korisnik ec2-user ima ovlasti za korištenje Docker-a sa komandom: `docker info`. Ako je sve u redu trebao bi se prikazati ekran sličan kao na slici 12..

Slika 12. Docker info

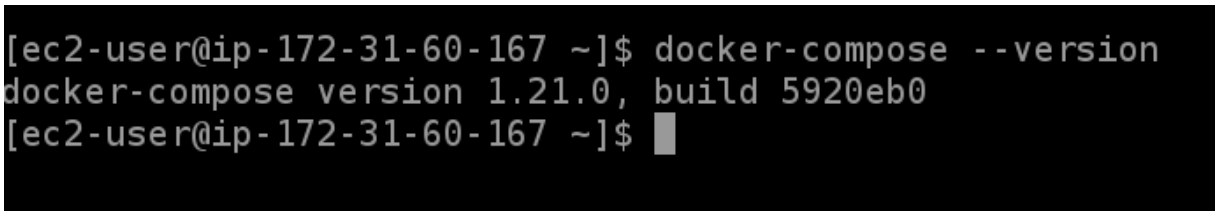
Izvor: izradio autor

Još na instanci nedostaje „docker-compose“. Njega dodajemo sa iduće četiri komande:

```
sudo curl -L
https://github.com/docker/compose/releases/download/1.21.0/docker-compose-
`uname -s`-`uname -m` | sudo tee /usr/local/bin/docker-compose > /dev/null ;
sudo chmod +x /usr/local/bin/docker-compose ;
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose ;
sudo service docker start ;
```

Zatvorimo i otvorimo instancu kako bi sustav ponovo vidio nove promjene na njemu. Provjeravamo ako je sve u redu sa komandom `docker-compose --version .` Ako je trebali bi dobiti ekran kao na slici 13..

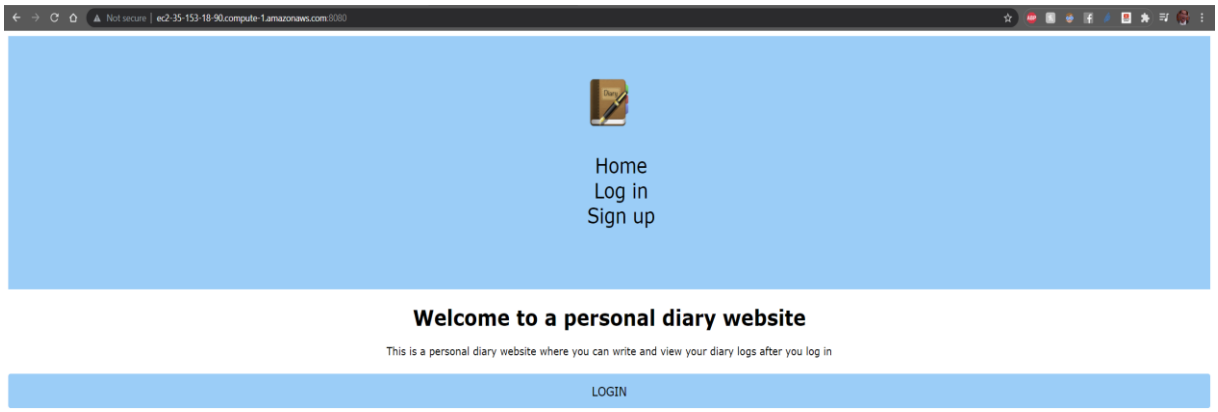
Slika 13. Verzija docker-compose-a



Izvor: izradio autor

Idući korak je prebacivanje datoteka na instancu, a to će se odraditi putem FileZilla aplikacije. Prije pokretanja FileZille moramo ključ koji smo dobili s instancom prebaciti na novi format a to radimo uz pomoć PuTTYgen-a. U PuTTYgen-u odaberemo opciju „load“ i odaberemo ključ koji smo skinuli. Pritiskom na tipku „Save private key“ ga spremamo u novom formatu. Sada možemo pokrenuti FileZillu te otvaramo Edit -> Settings -> SFTP -> Add key file. Odabiremo ključ koji smo stvorili (ključ se možda neće prikazati da je tamo ali je). Nakon dodatka ključa na vrhu ekrana upisujemo Host – javna IP adresa dodjeljena instanci, Username – ec2-user, Password – puštamo prazno i port – 22 za SFTP. Nakon što se FileZilla spoji jednostavno prebacimo datoteke sa lokalnog računala na AWS instancu. Navigiramo u direktorij koji smo prebacili i upisati komandu „docker-compose up“. Docker compose bi trebao odraditi svoje i na kraju kada dobijemo da su kontejneri gotovi možemo izaći iz instance i kopirati javni IPv4 DNS i dodamo port 8080. Trebala bi se otvoriti naša stranica.

Slika 14. Web aplikacija na AWS instanci



Izvor: izradio autor

6. ZAKLJUČAK

U ovaj završni rad se utrošilo puno vremena zato jer nisam znao ništa od koraka za izradu vrlo dobro. Morao sam se naučiti programirati u PHP-u, naučiti kako raditi sa Docker-om te naučiti kako funkcionira i kako objaviti dockeriziranu aplikaciju na AWS. PHP sam svladao najviše uz pomoć *tutorial* videa na YouTube-u i uz pomoć skripti koje sam našao na internetu. Za naučiti Docker pročitao sam puno službene Docker literature i one neslužbene. Za AWS sam također pročitao puno službene literature i dosta sam se konzultirao sa profesorom.

Najveći problem na ovom radu bi bio to što je aplikacija bila u Docker kontejneru i moglo joj se pristupiti preko *localhost*-a ali nije se moglo ulogirati u nju. Nakon puno proučavanja primijetio sam da u log-ovima na Dockerovoj aplikaciji piše da naredba „mysqli“, korištena u dbh.inc datoteci, ne postoji. PHP u Docker-ovoj izvedbi je bez dodanih naredbi i zato moramo instalirati ekstenzije koje su nam potrebne. Instalacijom ekstenzije Docker je dobio dodatak na PHP i onda se moglo logirati na stranicu. Problemi koji su još nastali bili su vezani za konfiguraciju AWS-a na lokalnom računalu. To se radi putem ključeva koji se dobiju ali iz nekog razloga moje lokalno računalo je uvijek javljalo grešku da ti ključevi nisu dobri. Takva greška se obično ne javlja i na ovom računalu je bila abnormalna. Taj dio se onda riješio putem funkcije EC2 na AWS i putem toga je sve radilo.

Putem ovog rada naučio sam puno o Docker-u i puno o AWS-u i uz sve muke drago mi je što je rad gotov i što radi kako bi trebao. Završni rad se može pogledati na idućem linku još neko vrijeme:
http://ec2-35-153-18-90.compute-1.amazonaws.com:8080/

LITERATURA

1. Amazon EC2 (2021), https://aws.amazon.com/ec2/?ec2-whats-new.sort-by=item.additionalFields.postDateTime&ec2-whats-new.sort-order=desc (09.07.2021.)
2. Amazon Web Services (2021), https://en.wikipedia.org/wiki/Amazon_Web_Services (15.05.2021.)
3. Bashir, F. (2018), How To Set Up Laravel, Nginx, and MySQL with Docker Compose, https://www.digitalocean.com/community/tutorials/how-to-set-up-laravel-nginx-and-mysql-with-docker-compose (10.06.2021.)
4. Deploy a PHP Application Using Docker-compose (2018), https://www.vultr.com/docs/deploy-a-php-application-using-docker-compose (08.05.2021.)
5. Docker (2021), https://www.docker.com/ (23.04.2021.)
6. Docker desktop (2021), https://www.docker.com/products/docker-desktop (23.04.2021.)
7. Docker For Beginners: From Docker Desktop to Deployment (2020), https://www.youtube.com/watch?v=i7ABlHngi1Q (10.05.2021.)
8. Docker Overview (2021), https://docs.docker.com/get-started/overview/ (23.04.2021.)
9. Docker (software) (2021), https://en.wikipedia.org/wiki/Docker_(software) (23.04.2021.)
10. Dockerise your PHP application with Nginx and PHP7-FPM (2016), http://geekyplatypus.com/dockerise-your-php-application-with-nginx-and-php7-fpm/ (15.05.2021.)
11. HTML (2021), https://en.wikipedia.org/wiki/HTML (23.04.2021.)
12. Lavnduski, M. (2018), How To Deploy a PHP Application Using Docker-compose, https://hostadvice.com/how-to/how-to-deploy-a-php-application-using-docker-compose/ (10.06.2021.)
13. PHP (2021), https://en.wikipedia.org/wiki/PHP (23.04.2021.)
14. PHP.NET (2021), https://www.php.net/ (23.04.2021.)
15. Sublime text (2021), https://www.sublimetext.com/ (23.04.2021.)

16. Sublime text (software) (2021), https://en.wikipedia.org/wiki/Sublime_Text (23.04.2021.)
17. Visual Studio Code (2021), https://en.wikipedia.org/wiki/Visual_Studio_Code (08.05.2021.)
18. What is PHP-FPM? (2011), https://php-fpm.org/ (10.06.2021.)

POPIS KRATICA

API – Application Programming Interface

AWS – Amazon Web Service

CLI – Command-Line Interface

EC2 – Elastic Compute Cloud

ECR – Elastic Container Registry

ECS – Elastic Container Service

HTML – Hyper Text Markup Language

IAM – Identity and Access Management

PHP – PHP: Hypertext Preprocessor

TJ. – To jest

TZV. – Tako zvano

POPIS SLIKA