

**SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA, RAČUNARSTVA  
I ELEKTROTEHNIKE**

**DIPLOMSKI RAD**

**RAZVOJ WEB APLIKACIJE ZA  
ADMINISTRACIJU AUKCIJA**

Antonio Markić

Mostar, listopad 2021.

**SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA, RAČUNARSTVA  
I ELEKTROTEHNIKE**

**DIPLOMSKI RAD**

**RAZVOJ WEB APLIKACIJE ZA  
ADMINISTRACIJU AUKCIJA**

**Mentor:**

**Doc.dr.sc.Goran Kraljević**

**Student:**

**Antonio Markić**

**Mostar, listopad 2021.**

## IZJAVA

*Izjavljujem da sam diplomski rad uradio samostalno, uz pomoć navedene literature i uz konzultacije s mentorom doc. dr. sc. Goranom Kraljevićem, kojem se posebno zahvaljujem na pruženoj pomoći i savjetima kojima me vodio tijekom izrade diplomskog rada!*

---

Antonio Markić

SVEUČILIŠTE U MOSTARU  
FAKULTET STROJARSTVA, RAČUNARSTVA I ELEKTROTEHNIKE

Diplomski studij: **Diplomski studij računarstva**

Smjer: **Programsko inženjerstvo i informacijski sustavi**

Ime i prezime: **ANTONIO MARKIĆ**

Broj indeksa: **478-RM**

ZADATAK DIPLOMSKOG RADA

Naslov: **RAZVOJ WEB APLIKACIJE ZA ADMINISTRACIJU AUKCIJA**  
( *WEB APPLICATION DEVELOPMENT FOR ADMINISTRATION OF AUCTIONS* )

Zadatak: U diplomskom radu je potrebno objasniti osnovne principe razvoja web aplikacija. Navesti i pojasniti serverske i klijentske tehnologije koje se koriste u razvoju web aplikacija (Symfony, MySQL, Docker, HTML, CSS, JavaScript). Korištenjem navedenih tehnologija , u praktičnom dijelu rada je potrebno implementirati bazu podataka te razviti web aplikaciju za administraciju aukcija.

Prijava rada:

Rok za predaju rada:

Rad predan:

Predsjednik Povjerenstva:

Mentor:

---

---

Doc.dr.sc. Goran Kraljević

**Antonio Markić**

## **Razvoj web aplikacije za administraciju aukcija**

### **Sažetak:**

U diplomskom radu predstavljena je web aplikacija za administraciju aukcija. Aplikacija omogućuje korisnicima pregled proizvoda i slanje ponuda za proizvode koji se nalaze na aukciji. Administracija podrazumjeva evidenciju svih proizvoda, svih poslanih ponuda i svih korisnika koji koriste aplikaciju. Kroz opis tehnologija, arhitekture i razvojnog okruženja je prikazan cjelokupan razvoj jedne takve aplikacije.

## **Web application development for administration of auctions**

### **Abstract:**

This thesis describes a web application for administration of auctions. The application allows users to view products and send bids for products that are up for the auction. Administration includes records of all products, all submitted offers and all users who use the application. Through the description of technologies, architecture and development environment, the overall development of one such application is presented.

# SADRŽAJ:

<b>1</b>	<b>UVOD.....</b>	<b>1</b>
<b>2</b>	<b>WEB TEHNOLOGIJE.....</b>	<b>2</b>
2.1	Web aplikacije .....	2
2.2	PHP .....	5
2.3	Symfony.....	8
2.4	MySQL baza podataka.....	10
2.5	Docker .....	11
2.6	Frontend tehnologije.....	13
2.6.1	HTML.....	14
2.6.2	CSS .....	15
2.6.3	JavaScript.....	16
<b>3</b>	<b>RAZVOJNI ALATI.....</b>	<b>17</b>
<b>4</b>	<b>ANALIZA ZAHTEJEVA .....</b>	<b>19</b>
4.1	Funkcionalni zahtjevi .....	19
4.2	Nefunkcionalni zahtjevi.....	20
4.3	Use case dijagram sustava .....	20
<b>5</b>	<b>IMPLEMENTACIJA DOCKERA .....</b>	<b>22</b>
5.1	Postavljanje Docker okruženja.....	22
<b>6</b>	<b>IMPLEMENTACIJA BAZE PODATAKA .....</b>	<b>27</b>
6.1	Relacijski model baze podataka .....	27
<b>7</b>	<b>IMPLEMENTACIJA SUSTAVA .....</b>	<b>33</b>
7.1	Konfiguracija baze podataka.....	33
7.2	Prikaz proizvoda .....	35
7.3	Login i registracija .....	37
7.4	Slanje ponuda.....	39
7.5	Servis za slanje mailova.....	42
7.6	Admin panel.....	43
7.6.1	Administracija ponuda.....	44
7.6.2	Administracija proizvoda .....	48
7.6.3	Administracija korisnika.....	52

<b>8 ZAKLJUČAK .....</b>	<b>54</b>
<b>LITERATURA .....</b>	<b>55</b>
<b>POPIS SLIKA.....</b>	<b>56</b>
<b>POPIS TABLICA .....</b>	<b>57</b>
<b>POPIS KODOVA .....</b>	<b>58</b>
<b>SKRAĆENICE .....</b>	<b>59</b>

# 1 UVOD

U ovom diplomskom radu je prikazan praktični primjer iz svijeta web aplikacija. Razvoj interneta u zadnjih 30-ak godina značajno je promijenio način odvijanja komunikacije i poslovanja u svijetu. World Wide Web (www) je postao globalna mreža povezanih dokumenata, slika, multimedije i aplikacija kojima mogu pristupiti korisnici interneta bilo gdje u svijetu. Razvijaju se načini komunikacije preko interneta od kojih je najstariji putem e-maila koji se i danas masovno koristi. Poduzeća su uvidjela koristi interneta te započinju sa internet prodajom svojih proizvoda i usluga. Time započinje i oglašavanje putem interneta bez čega je danas poslovanje gotovo nezamislivo. Zbog naglog razvoja web poslovanja došlo je i do povećane potrebe za programerima na tržištu rada.

Zadatak i cilj ovog rada je implementacija Web aplikacije za administraciju aukcija. Rad je podijeljen u osam poglavlja i to tako, da nakon uvodnog slijedi objašnjenje koncepta web aplikacija i detaljno objašnjenje tehnologija korištenih za izradu ove web aplikacije. U trećem poglavlju pažnja je posvećena alatima korištenima za razvoj aplikacije, dok se u četvrtom analiziraju zahtjevi koje aplikacija mora ispuniti. Peto poglavlje opisuje implementaciju Docker okruženja za razvoj aplikacije, a šesto implementaciju relacijskog modela baze podataka. Sedmo i najopširnije poglavlje opisuje implementaciju frontend i backend dijelova aplikacije. Rad završava zaključnim osmim poglavljem.



## 2 WEB TEHNOLOGIJE

### 2.1 Web aplikacije

Web aplikacija je računalni program pohranjen na udaljenom serveru, a korisnik mu pristupa preko interneta koristeći web preglednik. Poduzeća širom svijeta koriste internet kao najisplativiji način komunikacije. Internet im omogućuje brzu razmjenu informacija sa tržištem ali i obavljanje sigurnih transakcija. Učinkovito korištenje interneta je moguće samo ako su tvrtke sposobne prikupiti i pohraniti sve podatke koji su im potrebni te ako posjeduju načine njihove obrade i prezentacije korisniku. Web aplikacije koriste kombinaciju jezika na strani poslužitelja koji služe da bi spremali i dohvaćali podatke sa servera, te jezika na strani klijenta da bi prezentirali sadržaj korisnicima. Sve to omogućuje korisnicima komunikaciju sa poduzećima koristeći online obrasce, sustave za upravljanje sadržajem još mnogo toga. Web aplikacije koje ne zahtijevaju obradu podataka na strani servera nazivaju se statičke web aplikacije. Osim njih postoje i web aplikacije koje zahtijevaju obradu na strani servera i nazivaju se dinamičke web aplikacije. Za rad web aplikacije potreban je web server koji će upravljati zahtjevima klijenata, aplikacijski server za izvršavanje traženih zadataka, te često i baza podataka za pohranjivanje informacija. Primjer tipičnog tijeka rada web aplikacije:

- Korisnik šalje zahtjev web serveru putem internet
- Web server prosljeđuje zahtjev aplikacijskom server
- Aplikacijski server izvršava traženi zadatak, kao što je obrada podataka

ili upit u bazi podataka a zatim generira rezultate traženih podataka

- Aplikacijski server šalje rezultate na web server sa traženim podacima
- Web server odgovara klijentu sa traženim podacima koji se zatim

pojavljuju na korisnikovom ekranu. <sup>1</sup>

---

<sup>1</sup> <https://www.indeed.com/career-advice/career-development/what-is-web-application>

Počeci razvoja web aplikacija započinju još ranih 1990-ih kada su se pojavile prve statičke web stranice koje su samo prikazivale tekstualne dokumente. Ubrzo je postalo moguće dodati i slike, video zapise i audio zapise na web stranice. Želja da web stranice postanu dinamičnije, 1995.g. je dovela do stvaranja programskog jezika na strani klijenta pod nazivom JavaScript. JavaScript je omogućio dodavanje interaktivnih elemenata kao što su animacije, a ubrzo su postale popularne i interaktivne video igre. Veliki skok sa statičkih na dinamičke web aplikacije dogodio se 2005.g. razvojem Ajaxa koji je omogućio novi pristup responzivnom dizajnu web aplikacija, što je omogućilo programerima da rade brže i kvalitetnije.

2014.g. objavljen je HTML5 koji je unaprijedio postojeće HTML standarde. HTML5 je pružio podršku novim vrstama multimedije te omogućio stvaranje web aplikacija koje su neovisne o pretraživačima i platformama. 2015.g. pojavio se pojam "progresivne web aplikacije", a opisane su kao "web stranice koje su uzele sve prave vitamine". Drugim riječima, web aplikacije imaju sve funkcionalnosti izvorne aplikacije te korisnik ne može vidjeti nikakvu razliku između izvorne aplikacije i progresivne web aplikacije.<sup>2</sup>

Glavna prednost web aplikacija je da njihov rad ne ovisi o operacijskom sustavu korisnika. Programeri ne moraju pripremati različite verzije iste aplikacije za Microsoft Windows, Mac OS, Linux itd. Aplikacija se stvara samo jednom za sve platforme i sve operacijske sustave. Upravo zbog toga su postale izuzetno popularne tijekom 1990-ih i 2000-ih godina. No zbog različitih realizacija HTML-a, CSS-a, DOM-a i drugih sučelja u preglednicima može uzrokovati problem tijekom razvoja web aplikacija i njihove daljnje podrške. Web aplikacija može raditi nepravilno jer korisnik može promijeniti postavke svoga web preglednika onako kako želi.

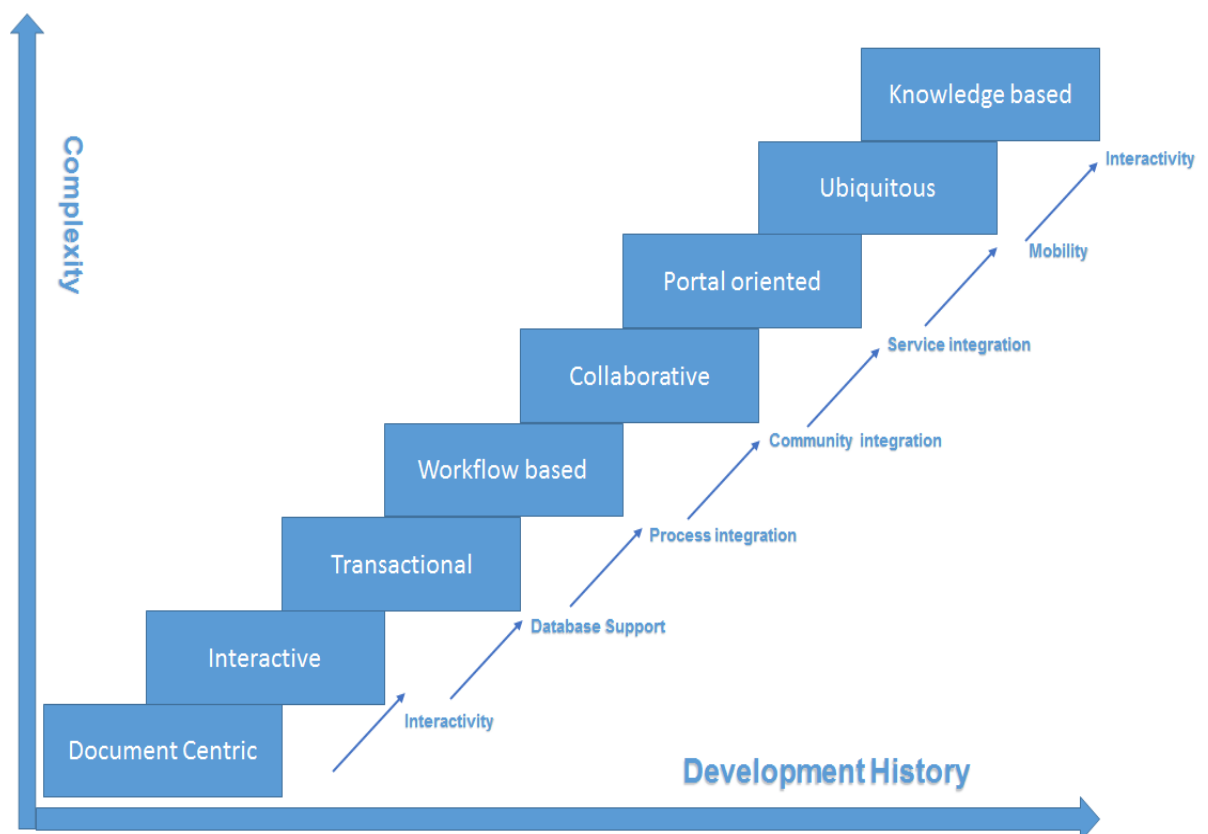
Na samom početku programeri su nailazili na velike probleme u načinu rada web aplikacija. Svaka je aplikacija imala svoj predkompajlirani klijentski program i morala se zasebno instalirati na svako računalo korisnika. Sve komponente klijenta i poslužitelja bile su čvrsto vezane za operacijski sustav i arhitekturu računala. Zbog toga je prijenos aplikacija na druge sustave bio izuzetno skup i nepraktičan. Kod prvih statičkih web stranica prilikom bilo kakve promjene sadržaja trebalo je proći određeno vrijeme da se uspostavi komunikacija sa serverom i da se stranica ponovno učita.<sup>3</sup>

---

<sup>2</sup> <https://aplextor.medium.com/a-brief-history-of-web-app-50d188f30d>

<sup>3</sup> <https://www.devsaran.com/blog/history-web-application-development>

Samo su se neumornim radom programera i pronalaskom radikalnih rješenja uspjeli riješiti postojeći problemi. U relativno kratkom razdoblju od 30 godina prešao se veliki put od prvih statičkih web stranica, do interaktivnih stranica nastalih korištenjem Java Scripta, korištenje baza podataka za rad sa većim količinama podataka, provođenje transakcija, b2b rješenja, društvenih mreža, prodaje preko interneta, sveprisutnih usluga za određivanje lokacija i razvoja umjetne inteligencija za pomoć pri donošenju odluka.<sup>4</sup>



Slika 2.1 Povijest razvoja web aplikacije

<sup>4</sup> <https://msatechnosoft.in/categories-of-web-applications-characteristics-of-web-applications/>

## 2.2 PHP

PHP ("Hypertext Preprocessor", a ranije "Personal Home Page Tools") je skriptni jezik na strani servera koji služi za programiranje dinamičkih web aplikacija. Razvoj PHP-a započeo je 1994.g. kada je Rasmus Lerdorf napisao nekoliko "Common Gateway Interface" programa u programskom jeziku C koji su služili za održavanje osobne web stranice. Zatim je omogućio da rade s web obrascima te da mogu komunicirati sa bazom podataka. Tu implementaciju nazvao je "Personal Home Page/Forms Interpreter" ili PHP/FI, a koristila se samo za izradu veoma jednostavnih dinamičkih web aplikacija. Lerdorf je 1995.g. objavio poboljšanu verziju 1.0 koja je imala varijable nalik Pearlu, rukovanje formama i mogućnost ugradnje u HTML. Taj programski jezik je s vremenom rastao te je 1997.g. objavljena nova verzija 2.0. Sam Lerdorf je rekao da mu nikad nije ni bila namjera napisati novi programski jezik nego je samo radio sljedeći logičan korak.

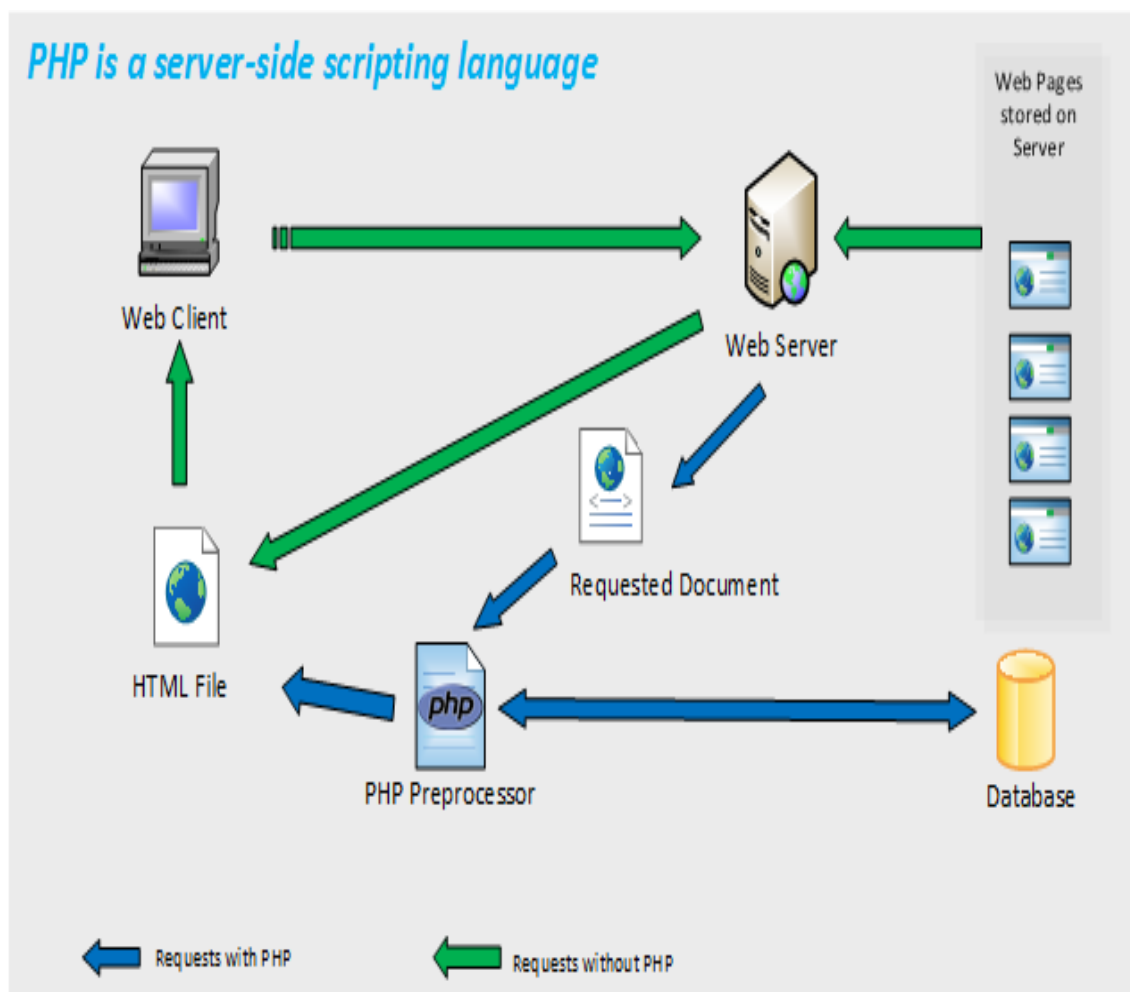
Zeev Suraski i Andi Gutmans su prepravili sintaksu 1997.g. čime su stvorili bazu za PHP 3 koji su javno objavili 1998.g., a koji je postao izuzetno popularan. PHP 4 je objavljen 2000.g., ažurirao se sve do 2008.g., a danas se više ne koristi. 2004.g. objavljen je PHP 5 koji je uključivao nove značajke kao što su podrška za objektno orijentirano programiranje, PDO proširenje i mnoga druga poboljšanja. Verzija 6 nikada nije objavljena, a verzija 7 izašla je 2015.g. Svrha PHP 7 bila je optimizirati performanse PHP-a preoblikovanjem Zend Engine-a uz zadržavanje gotovo potpune jezične kompatibilnosti. PHP 8 je objavljen 2020.g. a nove značajke uključuju: Just-in-time kompilaciju što znači da će PHP 8 kompajler pružiti značajno poboljšanje performansi u nekim slučajevima, „match“ izraz koji je sličan switch izjavi, novi „static“ povratni tip i novi „mixed“ tip.

U najvažnije razloge popularnosti PHP-a ubraja se lakoća usvajanja za početnike i sličnost sintakse programskom jeziku C. Svatko tko poznaje HTML vrlo brzo može naučiti raditi s PHP-om. PHP je jezik koji oprašta mnoge nepravilnosti, nema deklaracije tipova varijabli, a ista varijabla se može koristiti za spremanje različitih vrsti vrijednosti, a sve to ga čini primamljivim početnicima. Ne postoje točno određena pravila kako se grade funkcije pa programeri imaju prostora za inovacije. PHP se redovno održava i ima veliku podršku programerske zajednice.

Fleksibilnost kao najveća snaga PHP-a je ujedno i njegova najveća slabost zato što zna biti i previše tolerantan prema greškama. Budući da nema jasno definiranih pravila neiskusni programeri često znaju doći do loših rješenja za neke jednostavne probleme. Problem može predstavljati i pronalaženje grešaka koje postaje sve teže s rastom aplikacije.

Princip rada PHP-a :

- Pokreće skripte na web poslužitelju
- Korisnički zahtjev ispunjava se pokretanjem skripti izravno na web poslužitelju i generiranjem dinamičkih HTML stranica koje se zatim šalju u preglednik klijenta
- Obrada se događa na računalu poslužitelja
- Omogućuje interaktivne web aplikacije koje povezuju bazu podataka ili druge izvore podataka pohranjene na poslužitelju <sup>5</sup>



Slika 2.2 PHP kao skriptni jezik na strani poslužitelja

<sup>5</sup> <https://medium.com/laravel-power-devs/web-programming-with-php-b6c96d187070>

PHP služi za izvršavanje funkcija sustava, za izvođenje CRUD (create, read, update, delete) operacija nad podacima u sustavu, za obradu obrazaca, za pristup varijablama kolačića i postavljanje kolačića, za ograničavanje pristupa određenim dijelovima web stranica, za šifriranje podataka itd. PHP se danas najviše koristi za izradu funkcionalnosti na strani poslužitelja. Tijekom vremena došlo je do razvoja mnogih okvira (eng. framework) kao što su Symfony, Laravel, CodeIgniter, Yii Framework, CakePHP i mnogi drugi koji uvelike olakšavaju rad sa PHP-om. Odlično se integrira sa svim popularnim bazama podataka uključujući MySQL, Postgres, Oracle i MS Sql Server.

Kada želimo usporediti PHP s drugim programskim jezicima najčešće se prvo sjetimo Jave. Oba su jezika objektno orijentirani i otvorenog koda. Laki su za naučiti te su izuzetno popularni. Java se može koristiti za dinamičke web aplikacije, ali je to ipak kompajlirani (a ne interpretirani) programski jezik za opću namjenu. Javina glavna ideja je pisanje koda samo jednom za njegovu upotrebu na različitim platformama. Kod se može pokrenuti na bilo kojem uređaju na kojem je instaliran Java Virtual Machine (JVM). Kada usporedimo PHP i Javu možemo zaključiti da je PHP kodiranje nešto manje optimizirano zbog čega je kod kraći i potrebno je manje vremena za njegovo pisanje. Java se općenito smatra najsigurnijim programskim jezikom dok se PHP smatra nešto manje sigurnim.<sup>6</sup>



Slika 2.3 PHP logo

---

<sup>6</sup> <https://www.ideamotive.co/blog/php-vs-java-the-best-choice-for-web-development>

## 2.3 Symfony

Kako bismo mogli razumjeti što je to okvir (framework) za početak zamislimo jednu planinsku avanturu. Razvoj aplikacije je sličan penjanju po stijeni: na početku smo na dnu i moramo doći do vrha tj. moramo izraditi funkcionalnu aplikaciju. Ako se nitko prije nas nije popeo na tu planinu, morat ćemo sve proći sami: testiranje ruta, povremeno vraćanje unatrag kako se ne bismo zaglavili u zavoju itd. Nasuprot tome, ako je planinski vrh već netko osvojio, oni koji su prošli prije nas već su obavili ovaj posao, uočili greške, otvorili moguće staze (okvir) i postavili alate koji će vam olakšati uspon (naš rad). U principu, okvir se sastoji od skupa alata i metodologije. Skup alata je skup montažnih, brzo integriranih softverskih komponenti. Skup alata podrazumijeva pisanje manje koda s manje rizika od pravljenja greški kao i veću produktivnost. Metodologija je nacrt sklapanja aplikacije. Taj strukturirani pristup se u početku može činiti ograničavajućim. No, u praksi omogućuje programerima da učinkovito i djelotvorno rade na najsloženijim aspektima zadatka, a upotreba najboljih praksi jamči stabilnost, održivost i nadogradnju aplikacija koje razvijamo.<sup>7</sup>



Slika 2.4 Symfony framework logo

Symfony je PHP okvir i skup PHP komponenata za višekratnu upotrebu. Objavljen je kao besplatni softver 2005. pod licencom MIT, a bio je inspiriran Java frameworkom pod nazivom Spring. Symfony ima za cilj ubrzati stvaranje i održavanje web aplikacija i zamijeniti ponavljajuće zadatke kodiranja. Također je usmjeren na izgradnju robusnih aplikacija u kontekstu poduzeća i ima za cilj pružiti programerima potpunu kontrolu nad konfiguracijom: od strukture direktorija do stranih knjižnica, gotovo sve se može prilagoditi. Kako bi odgovarao smjernicama za razvoj poduzeća, Symfony se isporučuje s dodatnim alatima koji pomažu programerima u testiranju, uklanjanju pogrešaka i dokumentiranju projekata. Symfony koristi 600 000 programera iz 120 zemalja svijeta.

---

<sup>7</sup> <https://symfony.com/at-a-glance>

U odnosu na druge PHP razvojne okvire (engl. framework) Symfony je iznimno popularan. Intenzivno koristi postojeće PHP projekte otvorenog koda kao dio frameworka, kao što su:

- Propel ili Doctrine kao objektno-relacijski slojevi mapiranja
- PHPUnit – okvir za testiranje
- Twig – template engine
- Swift Mailer – knjižnica za slanje e-mailova

U glavne prednosti korištenja Symfonyja ubrajamo veliku brzinu i malu pohlepu. U IT svijetu nisu rijetki slučajevi da se ljudi brinu zbog izvedbe aplikacije nakon što je sve osmišljeno i na funkcionalnoj i na tehnološkoj razini. Symfony je od početka zamišljen kao brz, s jakim naglaskom na performanse. Usporedbe radi, Symfony je najbrži PHP okvir. Druga prednost je neograničena fleksibilnost. Bez obzira na potrebe programera, Symfony će biti prilagodljiv. To je 3 u 1 framework jer se može koristiti za: Full Stack (cjelovita verzija) ako želite razviti složenu aplikaciju i trebaju vam brojne funkcionalnosti, cigla po cigla ako svoj okvir gradite prema funkcionalnostima koje će vam trebati i mikrookvir koji se samostalno također može koristiti za razvijanje određene funkcionalnosti u jednom od projekata. Trajnost je također nešto što se odnosi na dugoročnu podršku. Profesionalnu podršku za Symfony pruža SensioLabs, ali postoji i čitav ekosustav koji je izrastao oko Symfonyja od njegovog pokretanja (Slack, Stack Overflow, itd.).

Treća prednost Symfonyja je njegova proširivost. Od najmanje cigle sve je predstavljeno kao "paket" u Symfonyju. Svaki je paket namijenjen dodavanju funkcionalnosti okvira, a može se ponovno upotrijebiti u drugom projektu ili podijeliti s ostatkom zajednice. Četvrta prednost je stabilnost i održivost. Postupak održavanja Symfonyja osigurava kompatibilnost između svih manjih verzija i pruža trogodišnju podršku za glavne verzije Symfonyja. Kao izuzetno funkcionalno okruženje, Symfony također jamči određenu razinu uživanja za programere. Među Symfonyjevim alatima dizajniranim da život programera uvelike olakšaju, tu je alatna traka za web uklanjanje pogrešaka, kao i izvorna podrška za razvojna okruženja i detaljne stranice s pogreškama. Možda i najvažnija prednost je jednostavnost korištenja. Obilna dokumentacija, podrška zajednice i profesionalna podrška omogućuju početnicima da se vrlo brzo osjećaju ugodno uz Symfony.<sup>8</sup>

---

<sup>8</sup> <https://symfony.com/six-good-technical-reasons>



## 2.4 MySQL baza podataka

MySQL je sustav za upravljanje relacijskim bazama podataka otvorenog koda (RDBMS). Bazu podataka definiramo kao zasebnu aplikaciju koja pohranjuje zbirku podataka. Svaka baza podataka ima jedan ili više različitih API-ja za stvaranje, pristup, pretraživanje, upravljanje i repliciranje podataka koje posjeduje. U teoriji bi se mogle koristiti i druge vrste spremišta podataka, kao što su velike raspršene tablice u memoriji ili datoteke u datotečnom sustavu, ali dohvaćanje i pisanje podataka ne bi bilo tako brzo i jednostavno s takvim vrstama sustava. Danas najčešće koristimo relacijske sustave za upravljanje bazama podataka (RDBMS) za pohranu i upravljanje ogromnom količinom podataka. Nazivaju se relacijske baza podataka jer se svi podaci pohranjuju u različite tablice, a odnosi se uspostavljaju pomoću primarnih ključeva ili drugih ključeva poznatih kao strani ključevi. Relacijski sustav upravljanja bazom podataka (RDBMS) softver je koji:

- Omogućuje implementaciju baze podataka s tablicama, stupcima i indeksima
- Jamči referentni integritet između redova različitih tablica
- Automatski ažurira indekse
- Tumači SQL upit i kombinira podatke iz različitih tablica

MySQL je jednostavan za korištenje, prilično je brz i koriste ga kako male tako i velike tvrtke. Više je čimbenika zbog kojih je MySQL postao jako popularan:

- MySQL je objavljen pod licencom otvorenog koda što znači da je besplatan za korištenje
- Koristi standardni oblik jezika SQL
- Radi veoma brzo i učinkovito i s velikim skupovima podataka
- Radi na mnogim operativnim sustavima i na mnogim jezicima, uključujući PHP, Javu, C, C# itd.
- Moguće ga je instalirati ručno, ali dolazi i u paketu u programima kao što su XAMPP, WAMP i LAMP<sup>9</sup>

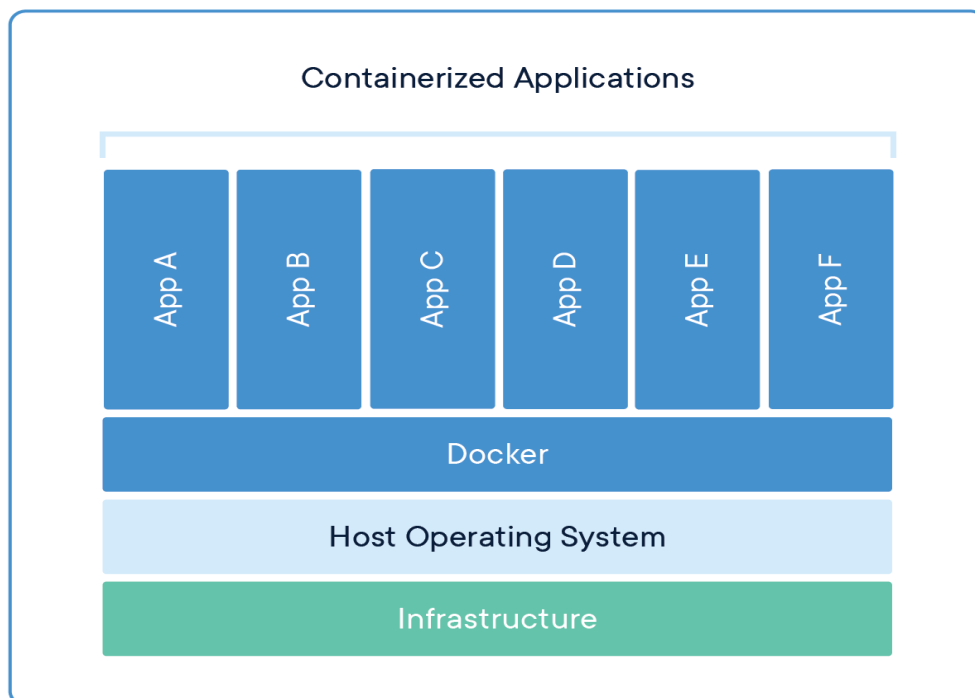
---

<sup>9</sup> <https://www.tutorialspoint.com/mysql/mysql-introduction.htm>

## 2.5 Docker

Docker je usluga upravljanja kontejnerima. Cijela ideja Docker je da programeri lakše razvijaju aplikacije, šalju ih u kontejnere koji se zatim mogu pokrenuti bilo gdje. Prvo izdanje Dockera objavljeno je u ožujku 2013.g. i od tada je postao jedan od najvažnijih elemenata programiranja današnjice.

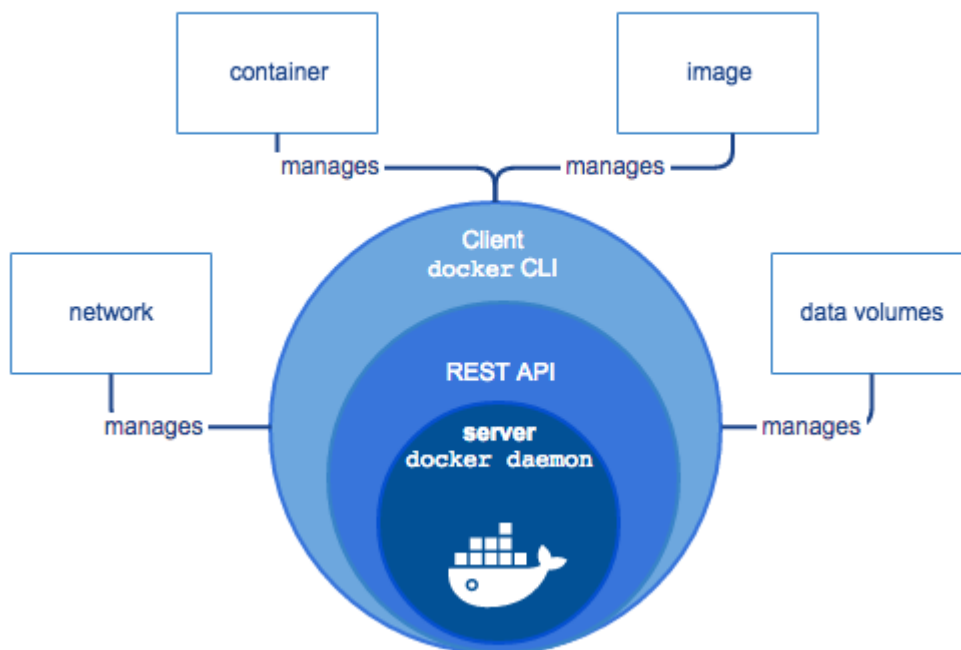
Docker kontejneri imaju zadatak izolirati aplikaciju i sve pripadajuće komponente aplikacije u kompletne jedinice koje mogu biti pokrenute bilo gdje. Isto tako smanjuju potrebu za hardver komponentama tako da omogućuju paralelno izvršavanje više zasebnih jedinica na jednom računalu čime povećavaju efikasnost računala. Docker kontejneri imaju određene sličnosti sa virtualnim strojevima. Imaju vlastita mrežna sučelja i vlastite IP adrese, mogu postavljati datotečne sustave i slično. Kontejneri su po veličini manji od virtualnih strojeva, uglavnom oko nekoliko desetaka megabajta i može ih se pokrenuti puno više na istom računalu. Kontejneri dijele jezgru operativnog sustava način da se da se svaki od njih izvršava kao zasebni proces unutar vlastitog korisničkog prostora. Docker spremnike možemo postaviti bilo gdje, na bilo kojem fizičkom i virtualnom stroju, pa čak i u oblaku. Budući da su Docker kontejneri prilično lagani, vrlo ih je lako prilagoditi.



Slika 2.5 Shema Docker kontejnera

Na slici 3. vidimo kako je operativni sustav domaćina stavljen na raspolaganje svim kontejnerima. Svaki kontejner ima svoj vlastiti korisnički prostor zbog čega možemo pokretati više zasebnih kontejnera na jednom računalu domaćinu.

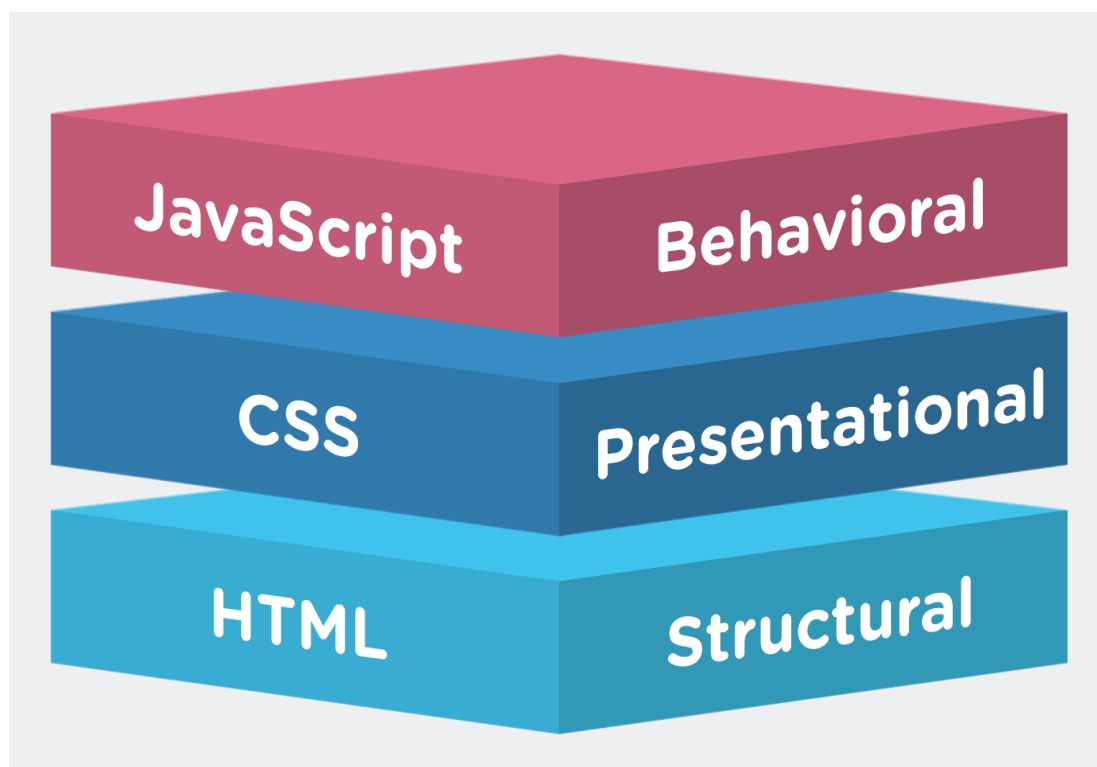
Da bi mogli pokrenuti Docker kontejnere moramo imati Docker pokretač (eng. Docker engine). To je sloj na kojem se Docker izvršava. Radi se o klijent – server tipu aplikacije koja upravlja kontejnerima, slikama kontejnera, izdanjima, mrežom i ostalim elementima koji se koriste pri pokretanju Docker kontejnera. Docker pokretač se sastoji od tri glavne komponente. Prva komponenta je server koji se izvršava na računalu domaćinu pod nazivom Docker pozadinski proces. Druga komponenta je REST aplikacijsko programsko sučelje koje sadrži specificirana sučelja putem kojih ostale aplikacije mogu komunicirati s pozadinskim procesima i izdavati upute istom. Treća i posljednja komponenta je Docker klijent odnosno 'docker' naredba koja se koristi u Terminalu preko upravljačke linije. Radi se o klijentu uz pomoć kojega komuniciramo s Docker pozadinskim procesima kako bi izvršili naredbe.



Slika 2.6 Docker pokretač

## 2.6 Frontend tehnologije

Razvoj frontenda ili razvoj na strani klijenta je proces stvaranje HTML-a, CSS-a i JavaScripta za web aplikaciju tako da ih korisnik može vidjeti i s njima komunicirati. Alati i tehnike za razvoj frontenda stalno napreduju pa programeri stalno moraju biti u toku sa razvojem tehnologije. Cilj dizajniranja web stranice je osigurati da korisnici kada otvore web stranicu vide informacije u formatu koji je jednostavan za čitanje i relevantan. To je dodatno komplicirano činjenicom da korisnici sada koriste veliki broj uređaja s različitim veličinama zaslona i rezolucijama, pa dizajneri moraju uzeti u obzir ove aspekte prilikom projektiranja web stranice. Moraju osigurati ispravno pojavljivanje svoje web stranice u različitim preglednicima, različitim operativnim sustavima i različitim uređajima, što zahtijeva pažljivo planiranje od strane programera.<sup>10</sup>



Slika 2.7 Frontend tehnologije : HTML, CSS, JavaScript

<sup>10</sup> <https://frontendmasters.com/books/front-end-handbook/2018/what-is-a-FD.html>

### 2.6.1 HTML

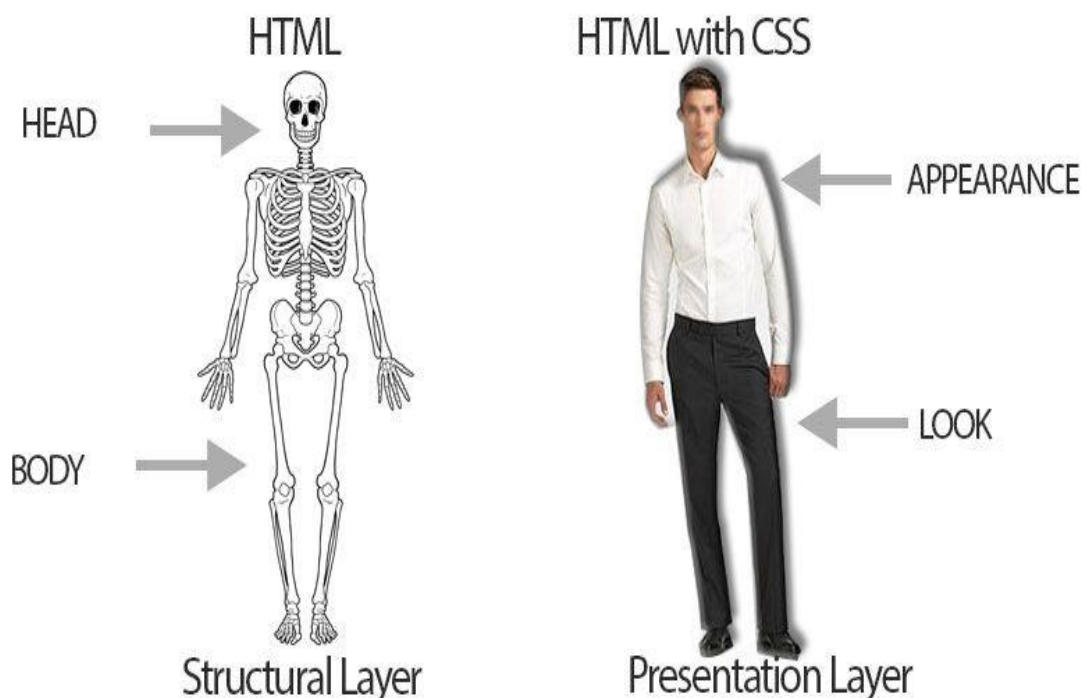
HTML (HyperText Markup Language) je prezentacijski jezik za izradu web stranica. HTML se sastoji od niza elemenata i opisuje strukturu web stranice. HTML elementi označavaju dijelove sadržaja i govore pregledniku kako prikazati sadržaj stranice. HTML je jednostavan za uporabu i lako se uči, što je jedan od razloga njegove opće prihvaćenosti i popularnosti. Od početka je bio zamišljen kao besplatan i dostupan svima. Pri prikazu HTML dokumenta želi se postići da taj dokument izgleda jednako bez obzira o kojemu je web pregledniku, računalu i operacijskom sustavu riječ. HTML datoteka mora imati ekstenziju .htm ili .html te može biti kreirana korištenjem bilo kojeg tekst editora.

Tim Berners-Lee objavio je 1991.g. dokument pod nazivom „HTML tags“ što je prvi javno objavljeni opis HTML-a. Dokument se sastoji od 20 osnovnih elemenata HTML-a. Mnoge oznake su nastale na temelju jednog od ranih jezika za formatiranje teksta pod nazivom runoff. Prva verzija HTML jezika objavljena je 1993. godine. Tada je bio još poprilično ograničen, pa nije bilo moguće čak ni dodati slike u HTML dokumente. Ubrzo je izašla verzija 2.0, no nije postala standardom. U ožujku 1995. W3C objavljuje verziju 3.0, koja donosi mogućnosti definicije tablica. Najveći i najprihvaćeniji web preglednici tada su prihvatili specifične oznake HTML-a. Nenamjerno je došlo i do stvaranja više oznaka koje su imale istu svrhu. Npr. podebljani tekst se mogao definirati oznakom `<b>`, ali i oznakom `<strong>`. U prosincu 1997.g. objavljen je HTML4. Neke oznake koje su smatrane suvišnim su uklonjene, a neke nove koje su nametnuli najveći web preglednici su dodane. HTML5 je peta i posljednja velika HTML verzija koja je preporuka konzorcija World Wide Web Consortium (W3C). Prvi put je objavljen 2008.g., te ponovno 2014.g. s velikom nadogradnjom. Cilj je bio dodati podršku za najnoviju multimediju i druge sadržaje te da jezik ostane lako čitljiv i za ljude i za računala. HTML5 omogućuje reprodukciju videozapisa bez korištenja Adobe flasha ili Microsoft Silverlighta, mogućnost upravljanja pomoću tipkovnice i opcijama za bilo koju vrstu manipulacija kao što su drag and drop, canvas itd. Za uključivanje i rukovanje multimedijским i grafičkim sadržajem dodani su novi elementi `<video>`, `<audio>` i `<canvas>`, a dodana je i podrška za sadržaj skalabilne vektorske grafike i za matematičke formule. Kako bi se obogatio semantički sadržaj dokumenata, dodaju se novi elementi strukture stranice kao što su `<main>`, `<section>`, `<article>`, `<header>`, `<footer>`, `<aside>`, `<nav>`, i `<figure>`.

### 2.6.2 CSS

CSS (Cascading Style Sheets) je stilski jezik koji se koristi za opisivanje prezentacije dokumenta napisanog u HTML-u ili XML-u. CSS je temeljna tehnologija World Wide Weba, uz HTML i JavaScript i standardiziran je u svim web preglednicima prema W3C specifikacijama. Kako se web razvijao, prvotno su u HTML ubacivani elementi za definiciju prezentacije, ali je ubrzo uočena potreba za stilskim jezikom koji će HTML osloboditi potrebe oblikovanja sadržaja. Kada su u HTML3.2 dodani tagovi poput `<font>` i `<color>` došlo je do velikih poteškoća za programere. Da bi se riješio ovaj problem, World Wide Web Consortium (W3C) stvorio je CSS.

Drugim riječima, CSS definira kako prikazati HTML elemente. Korištenjem CSS-a postalo je moguće odvojiti prezentaciju podataka i dizajn od same strukture podataka. Razdvajanje prezentacije i sadržaja je dovelo do poboljšanja pristupačnosti sadržaja, većoj fleksibilnosti i kontroli karakteristika prezentacije, te mogućnost da više stranica koristi isti stil razdvajanjem CSS-a u zasebnu .css datoteku. Samim time je i HTML kod postao pregledniji i lakši za korištenje.

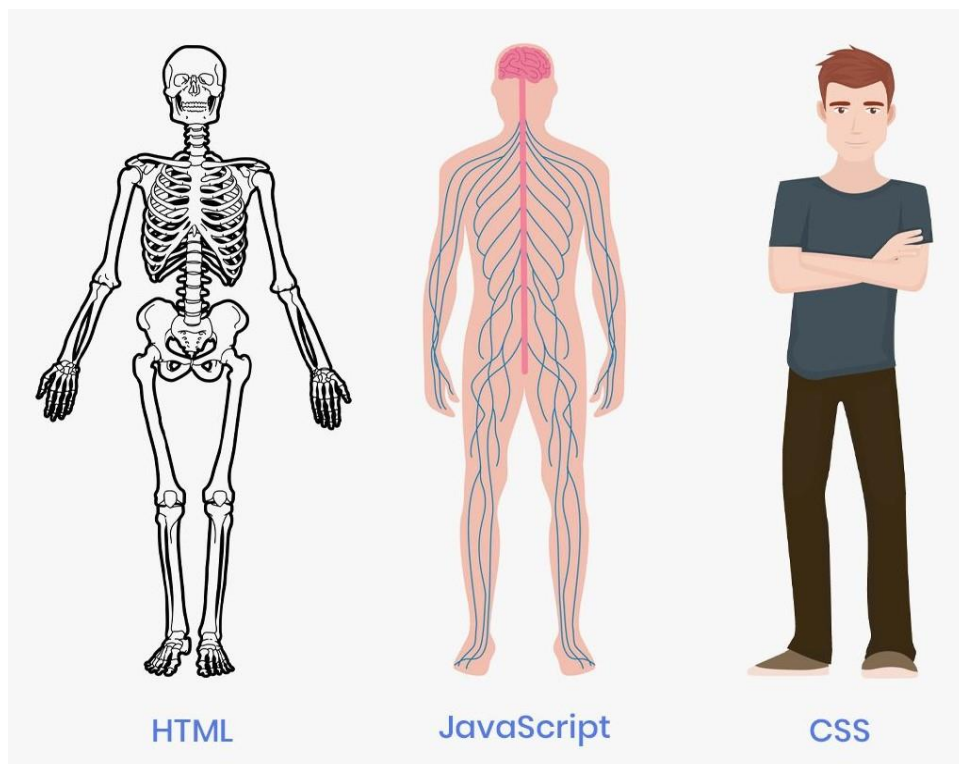


Slika 2.8 HTML i CSS

### 2.6.3 JavaScript

JavaScript je skriptni programski jezik, koji se izvršava u web pregledniku na strani korisnika. Uz HTML i CSS, JavaScript je jedna od temeljnih tehnologija World Wide Weba. Koristi se za razvoj dinamičkih elemenata web stranice. Napravljen je da bude sličan Javi zbog lakšega korištenja, ali nije objektno orijentiran kao Java i tu prestaje svaka njihova povezanost.

Izvorno ga je razvila tvrtka Netscape. 1994.g. tvrtka Netscape je izdala web preglednik Netscape Navigator koji je postao jedan od najkorištenijih preglednika u to doba. Tada su web stranice mogle biti samo statične, bez ikakvih dinamičkih elemenata nakon učitavanja stranice u preglednik. Postojala je želja da se ukloni to ograničenje pa je Netscape dodao skriptni jezik u Navigator. Uprava Netscapea dala je zadatak Brendanu Eichu da osmisli novi jezik sa sintaksom sličnom Javi. Iako su se novi jezik i njegova implementacija tumača zvali LiveScript kada su prvi put isporučeni kao dio beta verzije Navigatora u rujnu 1995., naziv je za službeno izdanje u prosincu promijenjen u JavaScript. Smatra se da je to bio marketinški trik budući da je Java tada bila novi popularni jezik.



Slika 2.9 HTML, JavaScript i CSS

### 3 RAZVOJNI ALATI

Za realizaciju projekta korišteni su različiti alati. Redom korišteni alati su:

- Docker Desktop
- JetBrains PHPStorm
- Postman
- TablePlus
- Cygwin Terminal
- Google Chrome web preglednik

Docker Desktop je aplikacija jednostavna za instaliranje za Mac ili Windows okruženja koja omogućuje izgradnju i dijeljenje kontejnerskih aplikacija i mikroservisa. Docker Desktop sadrži Docker Engine, Docker CLI client, Docker Compose, Docker Content Trust, Kubernetes i Credential Helper. Docker Desktop za Windows koristi Hyper-V virtualizaciju i umreživanje i najbrži je i najpouzdaniji način za razvoj Docker aplikacija na Windows operacijskom sustavu. Docker Desktop za Windows može pokretati i Linux i Windows Docker kontejnere.

PHPStorm je multiplatformsko IDE (integrirano razvojno okruženje) za PHP koje je razvila češka tvrtka JetBrains. PHPStorm omogućuje editor za PHP, HTML i JavaScript s analizom koda u hodu, sprečavanjem pogrešaka, automatiziranim prerađivanjem koda za PHP i JavaScript, kompletiranje koda, integraciju Git-a, integraciju Command Prompt-a, debugging i razne druge olakšice. Korisnici mogu proširiti PHPStorm instaliranjem dodataka stvorenih za PHPStorm (proširenje za Symfony) ili pisanjem vlastitih dodataka.

Postman je aplikacija koja se koristi za testiranje API-ja. Postman pojednostavljuje svaki korak u razvoju API-ja i pojednostavljuje suradnju tako da možemo brže razvijati API-je. To je HTTP klijent koji testira HTTP zahtjeve, koristeći grafičko korisničko sučelje, putem kojeg dobivamo različite vrste odgovora koje je potrebno naknadno potvrditi.

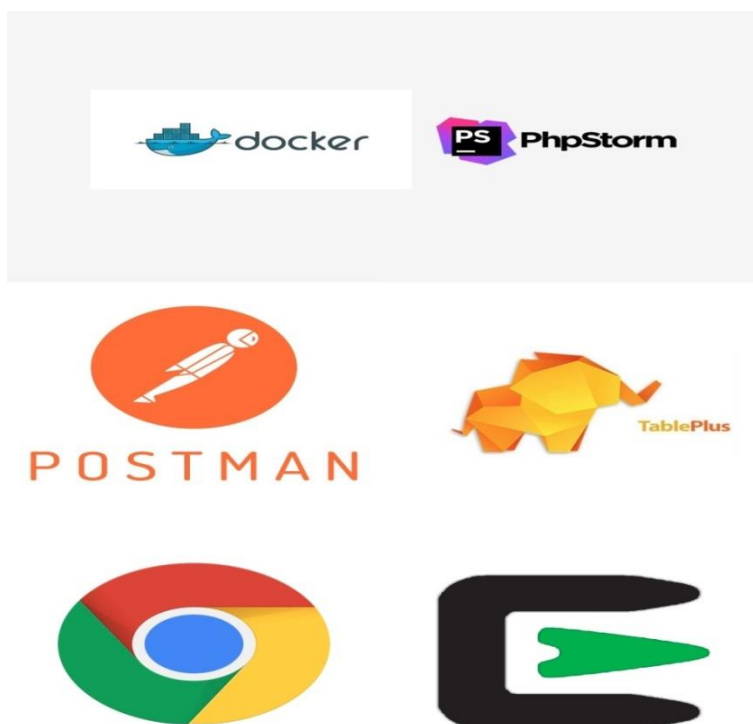
TablePlus je moderna aplikacija sa čistim korisničkim sučeljem koja programerima omogućuje istovremeno upravljanje bazama podataka na vrlo brz i siguran način.



TablePlus podržava većinu popularnih baza podataka kao što su MySQL, Postgres, SQL Server, SQLite, Microsoft SQL Server, Redis, Redshift, Oracle i mnoge druge.

Cygwin je zbirka alata otvorenog koda koja omogućuje sastavljanje i pokretanje Unix i Linux aplikacija na Windows operativnom sustavu unutar sučelja nalik Linuxu. Cygwin omogućuje korištenje ovih skripti i na računalima sa sustavom Windows. Cygwin instalacijski direktorij ima raspored direktorija koji je sličan korijenskom datotečnom sustavu sustava sličnih Unixu, s poznatim direktorijima kao što su /bin, /home, /etc, /usr, /var. Cygwin se instalira sa stotinama alata za naredbeni redak i drugim programima koji se obično nalaze na sustavu sličnom Unixu. Velika vrednost Cygwina je da je to besplatan softver otvorenog koda pa ga svi mogu koristiti. S njime je moguće pokrenuti Windows aplikacije iz Cygwin okruženja, kao i koristiti Cygwin alate i aplikacije u Windows operacijskom sustavu.

Google Chrome je web preglednik koji je razvila američka tvrtka Google. Preglednik koristi Appleov WebKit layout engine za prikazivanje web stranica. Neke od značajka Chrome preglednika su anonimni način rada, vrlo brzo izvršavanje JavaScript koda, jednostavno sučelje, te instaliranje dodataka i mogućnost sinkroniziranja zabilježaka s Google korisničkim računom.



Slika 3.1 Korišteni alati

## 4 ANALIZA ZAHTEJEVA

Definiranje zahtjeva je ključni dio cjelokupnog životnog ciklusa razvoja softvera. Definiranje zahtjeva nam govori što sustav u fazi razvoja treba raditi. Slabo definirani zahtjevi ili zahtjevi koji ne odražavaju stvarne potrebe korisnika mogu uzrokovati propast projekta, što znači da je značajan dio truda potrebno posvetiti da bi se razumjele potrebe korisnika. Funkcionalni zahtjevi predstavljaju svojstva i ponašanje sustava, dok nefunkcionalni zahtjevi određuju ograničenja i uvjete na sustav.

### 4.1 Funkcionalni zahtjevi

Funkcionalni zahtjevi definiraju usluge koje sustav mora pružiti, te kako se sustav mora ponašati u određenim situacijama. Funkcionalni zahtjevi aplikacije su :

- Posjetitelj mora imati pregled svih proizvoda na aukciji
- Aplikacija posjetitelju mora omogućiti registraciju
- Aplikacija mora imati formu za prijavu korisnika u aplikaciju
- Aplikacija mora omogućiti prijavljenom korisniku slanje ponuda za proizvode na aukciji
- Aplikacija mora omogućiti prijavljenom korisniku pregled svih njegovih ponuda
- Aplikacija mora omogućiti prijavljenom korisniku pregled svih ponuda za određeni proizvod
- Sustav mora ukloniti proizvode sa aukcije nakon isteka vremena aukcije
- Aplikacija mora omogućiti administratoru prijavu na sustav
- Aplikacija mora administratoru omogućiti pregled ponuda za sve proizvode
- Aplikacija mora administratoru dati mogućnost da prihvati ili odbije svaku ponudu
- Aplikacija mora omogućiti administratoru pregled svih proizvoda

- Aplikacija mora omogućiti administratoru dodavanje novih proizvoda
- Aplikacija mora omogućiti administratoru ažuriranje postojećih proizvoda
- Aplikacija mora omogućiti administratoru dodavanje novih administratora
- Aplikacija mora omogućiti administratoru oduzimanje privilegija postojećim administratorima
- Aplikacija mora omogućiti administratoru promjenu lozinke
- Aplikacija na kraju mora imati odjavu za sve korisnike

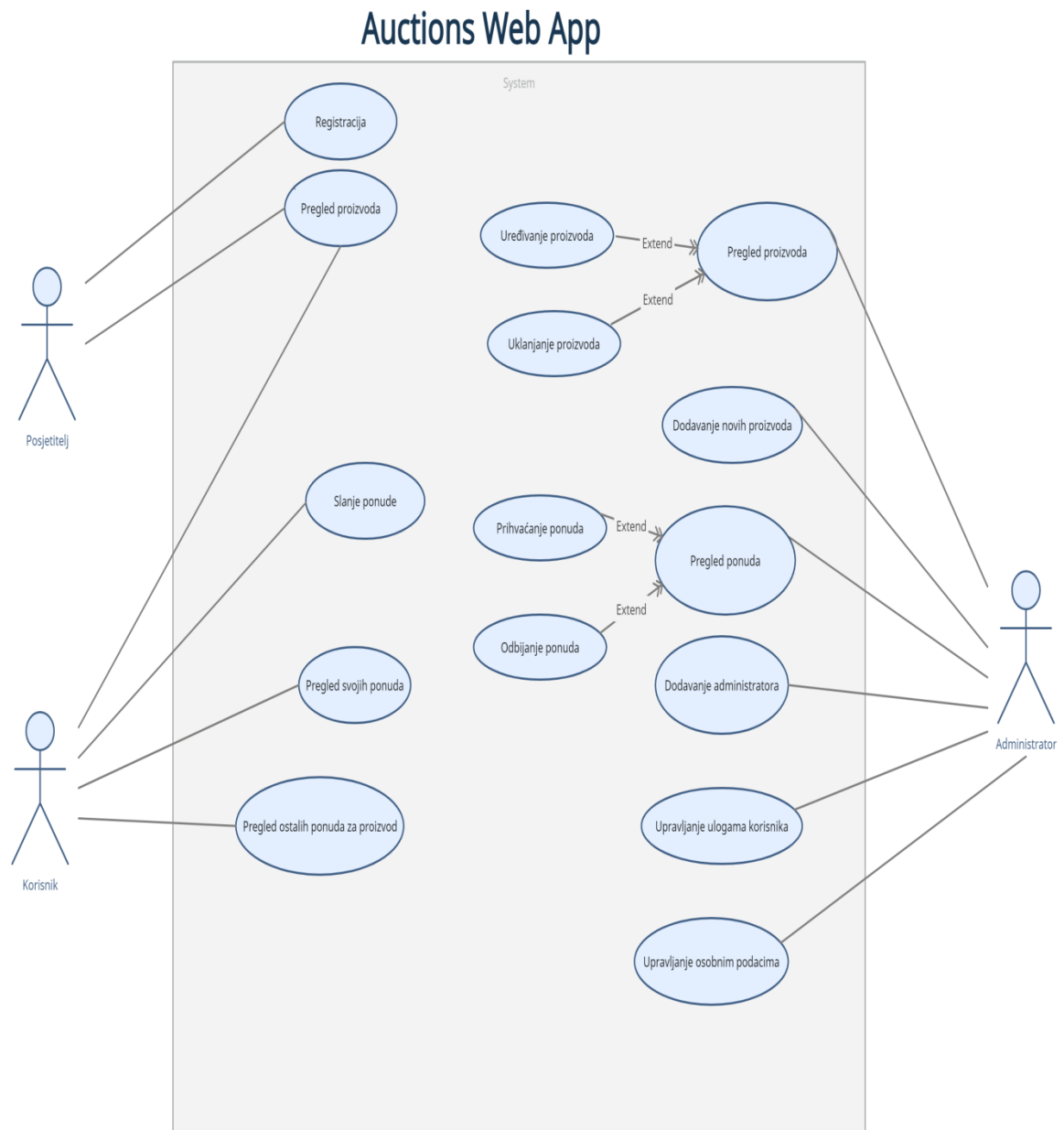
## **4.2 Nefunkcionalni zahtjevi**

Nefunkcionalni zahtjevi predstavljaju ograničenja u uslugama i funkcijama. Nefunkcionalni zahtjevi aplikacije su:

- Sučelje mora biti jednostavno
- Aplikacija treba biti dostupna 24 sata na dan
- Sve nepravilnosti u radu aplikacije treba otkriti na vrijeme
- Neovlašteni pristup aplikaciji treba biti onemogućen
- Aplikacija treba biti kompatibilna sa svim operativnim sustavima

## **4.3 Use case dijagram sustava**

Use case dijagram sustava prikazuje tipičnu interakciju korisnika i sustava. Sastoji se od izvođača, slučajeva korištenja (eng. use case) i veza. Izvođač predstavlja ulogu korisnika i use case opisuje cilj koji korisnik pokušava postići korištenjem sustava.



Slika 4.1 Use case dijagram

## 5 IMPLEMENTACIJA DOCKERA

### 5.1 Postavljanje Docker okruženja

Prilikom razvoja i pokretanja programa može doći do pogrešaka koje se pojavljuju samo na određenom operacijskom sustavu. Zato kada se pojave određene pogreške, postaje teško pronaći izvor problema. Zbog toga prilikom razvoja aplikacije koristimo Docker kontejnere. Kontejneri su mala i lagana okruženja za izvršavanje koja daju predvidljivost radu aplikacije tako što je stavljaju u kontejnere. Kontejneri zajednički koriste jezgru temeljnog operacijskog sustava, ali rade odvojeno jedan od drugog. Prilikom korištenja Dockera sa Symfony projektom koristimo Ngnix kao web server, PHP-FPM će obrađivati PHP zahtjeve, a MySQL će biti pozadinska baza podataka. Za komunikaciju između kontejnera će se koristiti Docker Compose. Unutar direktorija projekta nalazi se datoteka *docker-compose.yml*. Ova će datoteka sadržavati svu konfiguraciju za kontejnere koji će se koristiti u aplikaciji. Prvo definiramo php kontejner koji će obrađivati PHP zahtjeve. Unutar *.docker* direktorija kreiramo php direktorij unutar kojeg kreiramo Dockerfile koji sadrži:

```
ARG PHP_VERSION=7.4
FROM php:${PHP_VERSION}-fpm

RUN apt-get update && apt-get install -y \
    git \
    npm \
    nano \
    && docker-php-ext-install pdo_mysql
RUN apt-get update && apt-get install -y \
    git \
    npm \
    nano \
    && docker-php-ext-install pdo_mysql
COPY ../.../.
RUN composer install
RUN apt-get clean; rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
/usr/share/doc/*
```

Kôd 5.1: Dockerfile za PHP

Dockerfile definira verziju php-a 7.4 i instalira potrebne php ekstenzije i Composer. Zatim u docker-compose.yml dodajemo sljedeću konfiguraciju:

```
php:
  build:
    context: .
    dockerfile: .docker/php/Dockerfile
  container_name: ${PROJECT_NAME}-php
  working_dir: /application
  volumes:
    - ../application
    - ../.docker/php/php-ini-
overrides.ini:/etc/php/7.2/fpm/conf.d/99-overrides.ini
  ports:
    - "9000:9000"
```

Kôd 5.2: Docker-compose.yml konfiguracija PHP kontejnera

Sa ovom konfiguracijom određujemo način izrade php kontejnera prilikom izvođenja naredbe `docker compose`. Dockerfile definira lokaciju našeg php dockerfile-a, `container_name` naziv našeg kontejnera , a jednak je nazivu\_projekta-php, `working_dir` određuje radni direktorij, još deklariramo i volumen kako bismo zadržali podatke koje generira kontejner te port 9000 na računalu mapiramo u port 9000 na kontejneru.

Nakon toga prelazimo na implementaciju Nginx kontejnera koji će se koristiti kao web server. Prvo trebamo napisati zadanu konfiguraciju za server. U korijenu projekta kreiramo direktorij pod nazivom `nginx` i u njemu kreiramo konfiguracijsku datoteku pod nazivom `nginx.conf` koja izgleda:

```

server {
    listen 80 default;
    server_name ${PROJECT_URL};

    client_max_body_size 108M;

    access_log /var/log/nginx/application.access.log;

    root /application/public;
    index index.php;

    if (!-e $request_filename) {
        rewrite ^.*$ /index.php last;
    }

    location ~ \.php$ {
        fastcgi_pass php:9000;
        fastcgi_index index.php;
        fastcgi_param SCRIPT_FILENAME
        $document_root$fastcgi_script_name;
        fastcgi_param PHP_VALUE
        "error_log=/var/log/nginx/application_php_errors.log";
        fastcgi_buffers 16 16k;
        fastcgi_buffer_size 32k;
        include fastcgi_params;
    }
}

```

Kôd 5.3: Nginx.conf file

Ovo je osnovna konfiguracija Nginx-a potrebna za pokretanje Symfony projekta. Zatim dodajemo konfiguraciju Nginx kontejnera u docker-compose.yml, nakon konfiguracije PHP kontejnera:

```

nginx:
  image: nginx:alpine
  container_name: ${PROJECT_NAME}-nginx
  working_dir: /application
  depends_on:
    - php
  volumes:
    - ../application
    - ../.docker/nginx/nginx.conf:/etc/nginx/conf.d/default.conf
    - ../mysql/data
  ports:
    - "80:80"

```

Kôd 5.4: Docker-compose.yml konfiguracija Nginx kontejnera

Deklariramo `image: nginx:alpine` koji govori Dockeru iz koje slike da izgradi kontejner, `container_name` definira naziv kontejnera kao `naziv_projekta-nginx`, radni direktorij je `/application`, ovisi o `php-u`, te postavljamo port 80. Još je potrebno definirati kontejner baze podataka.

```

mysql:
  image: mysql:8.0
  container_name: ${PROJECT_NAME}-mysql
  working_dir: /mysql/data
  volumes:
    - ../application
    - mysql-data:/var/lib/mysql
  environment:
    - MYSQL_ROOT_PASSWORD=${DATABASE_ROOT_PASSWORD}
    - MYSQL_DATABASE=${DATABASE_NAME}
    #in case of changing user from 'root' to something else
    #make sure to add - MYSQL_USER=${MYSQL_USER} env variable under
  ports:
    - "3306:3306"

```

Kôd 5.5: Docker-compose.yml konfiguracija MySQL baze podataka



U image deklariramo mysql: 8.0 jer želimo koristiti verziju 8 MySQL, naziv kontejnera je naziv\_projekta-mysql, radni direktorij je /mysql/data, environment je ključ s kojim možemo odrediti varijable okruženja kao što su lozinka glavnog korisnika i naziv baze podataka, te pomoću ključa portova određujemo port na našem lokalnom razvojnom stroju i preslikavamo ga u port u kontejneru koji će se koristiti za rukovanje vezama baze podataka. Na kraju još dodajemo i kontejner za mailcatcher koji će se koristiti za primanje mailova prilikom testiranja rada aplikacije.

```
mailcatcher:
  restart: on-failure:10
  image: dockage/mailcatcher:0.7.1
  ports:
    - "1080:1080"
    - "1025:1025"
```

Kôd 5.6: Docker-compose.yml konfiguracija za MailCatcher

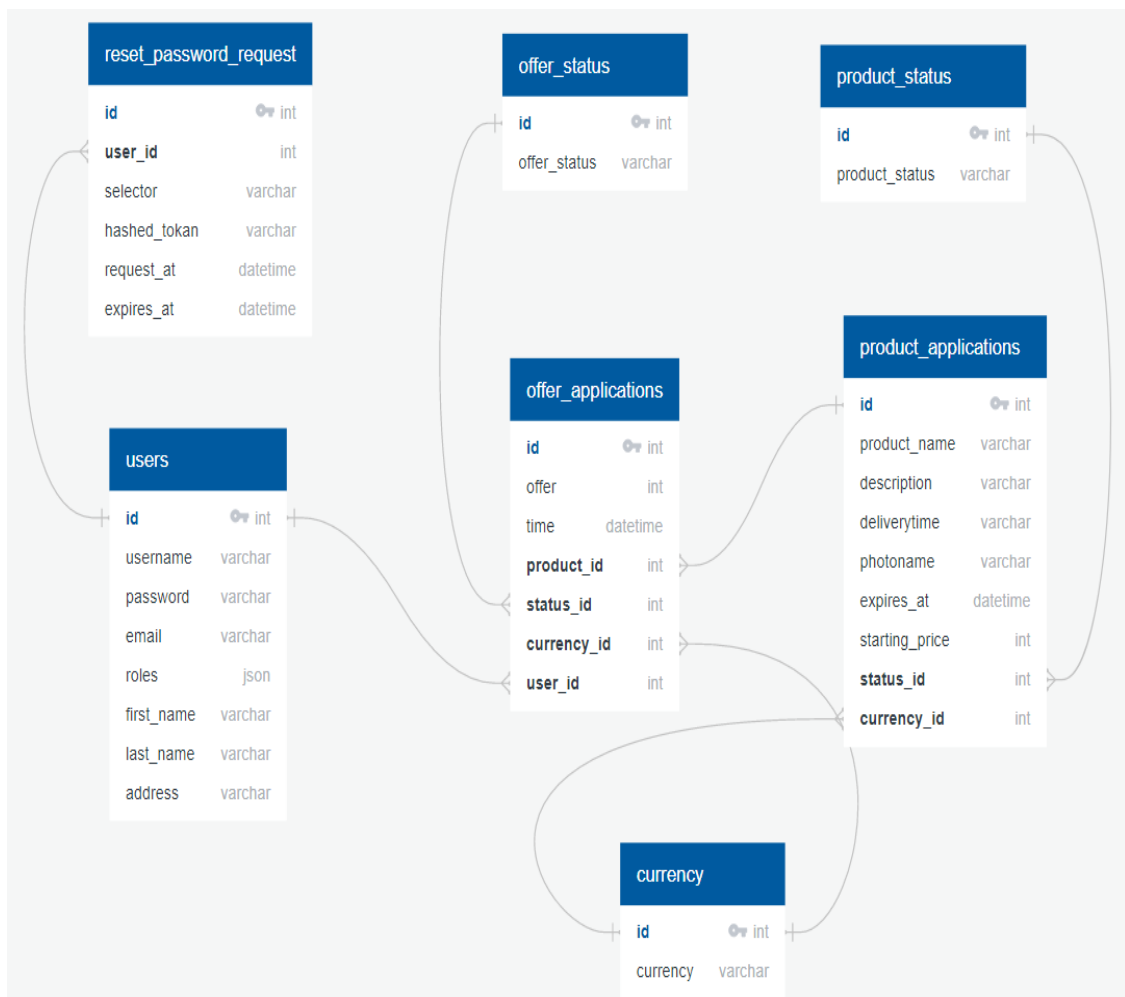
Konačno možemo izgraditi kontejnere s naredbom "docker-compose build" i pokrenuti ih s naredbom "docker-compose up". Da bismo stvorili Symfony aplikaciju, moramo pokrenuti terminal u našem PHP kontejneru. To možemo učiniti sa "docker-compose exec php /bin/bash". Nakon toga možemo kreirati Symfony project sa naredbom "symfony new ." .

## 6 IMPLEMENTACIJA BAZE PODATAKA

U ovom poglavlju bit će opisana logika baze podataka korištena za ovaj projekt. Za bazu podataka će se koristiti MySQL baza podataka.

### 6.1 Relacijski model baze podataka

Sljedeća slika prikazuje relacijski model baze podataka.



Slika 6.1 Relacijski model baze podataka

Popis tablica u relacijskom modelu:

- users
- reset\_password\_request
- offer\_applications
- offer\_status
- product\_applications
- product\_status
- currency

#### Tablica users

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>username</b>	varchar(25)	Not null	<i>Naziv korisničkog računa</i>
<b>password</b>	varchar(25)	Not null	<i>Lozinka</i>
<b>email</b>	varchar(25)	Not null	<i>E-mail adresa</i>
<b>roles</b>	json	Not null	<i>Uloga</i>
<b>first_name</b>	varchar(25)	Not null	<i>Ime korisnika</i>
<b>last_name</b>	varchar(25)	Not null	<i>Prezime korisnika</i>
<b>address</b>	varchar(45)	Not null	<i>Puna adresa korisnika</i>

Tablica 7.1 Struktura tablice users

Ova tablica služi za autorizaciju korisnika. Tablica se sastoji od korisničkog imena, lozinke, emaila, uloge, te imena, prezimena i adrese korisnika. Lozinka se prilikom spremanja hashira.

**Tablica reset\_password\_request**

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>user_id</b>	int	Not null	<i>Strani ključ (tablica users)</i>
<b>selector</b>	varchar(20)	Not null	<i>selektor</i>
<b>hashed_token</b>	varchar(100)	Not null	<i>Hashirani token za resetiranje lozinke</i>
<b>requested_at</b>	datetime	Not null	<i>Vrijeme slanja zahtjeva</i>
<b>expires_at</b>	datetime	Not null	<i>Vrijeme isteka zahtjeva (1 sat nakon slanja)</i>

Tablica 7.2 Struktura tablice reset\_password\_request

U ovoj tablici spremaju se zahtjevi za resetiranje lozinke koje šalju korisnici u slučaju zaboravljene lozinke. Korisnik na svoju mail adresu dobije hashirani token za promjenu lozinke koji ima vijek trajanja od sat vremena.

**Tablica currency**

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>currency</b>	varchar(25)	Not null	<i>Valuta</i>

Tablica 7.3 Struktura tablice currency

Tablica currency služi za pohranjivanje valuta i povezana je sa tablicama product\_applications i offer\_applications u kojima se pohranjuje samo iznos novca.

Tablica product\_applications

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>product_name</b>	varchar(64)	Not null	<i>Naziv proizvoda</i>
<b>description</b>	varchar(64)	Not null	<i>Opis proizvoda</i>
<b>deliverytime</b>	varchar(64)	Not null	<i>Vrijeme potrebno za dostavu</i>
<b>photoname</b>	varchar(64)	Not null	<i>Naziv slike proizvoda</i>
<b>expires_at</b>	datetime	Not null	<i>Vrijeme isteka aukcije za proizvod</i>
<b>currency_id</b>	int	Not null	<i>Strani ključ (tablica currency)</i>
<b>starting_price</b>	int(11)	Not null	<i>Početna cijena proizvoda</i>
<b>status_id</b>	int	Not null	<i>Strani ključ (tablica product_status)</i>

Tablica 7.4 Struktura tablice product\_applications

Tablica product\_applications služi za pohranjivanje proizvoda koji se šalju na aukciju. Tablica se sastoji od naziva proizvoda, opisa proizvoda, vremena dostave, naziva slike proizvoda (sama slika se pohranjuje u folder sa slikama), datuma i vremena isteka aukcije za proizvod, početne cijene proizvoda i stranih ključeva vezanih za valutu i status proizvoda.

#### Tablica product\_status

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>product_status</b>	varchar(25)	Not null	<i>Status proizvoda (aktivno ili prodano)</i>

Tablica 7.5 Struktura tablice product\_status

Tablica product\_status služi za pohranjivanje statusa proizvoda. Product\_status određuje da li je proizvod još uvijek na aukciji ili je aukcija završena.

#### Tablica offer\_status

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>offer_status</b>	varchar(25)	Not null	<i>Status ponude (otvoreno, prihvaćeno, odbijeno)</i>

Tablica 7.6 Struktura tablice offer\_status

Tablica offer\_status služi za pohranjivanje statusa ponude. Offer\_status određuje da li je ponuda još uvijek otvorena, da li je prihvaćena ili je odbijena.

Tablica offer\_applications

Naziv atributa	Tip podatka	NULL/NOT NULL	Kratki opis atributa
<b>id</b>	int(11)	Not null	<i>Primarni ključ</i>
<b>offer</b>	int(11)	Not null	<i>Iznos ponude</i>
<b>time</b>	datetime	Not null	<i>Vrijeme podnošenja ponude</i>
<b>product_id</b>	int	Not null	<i>Strani ključ (tablica product_applications)</i>
<b>currency_id</b>	int	Not null	<i>Strani ključ (tablica currency)</i>
<b>status_id</b>	int	Not null	<i>Strani ključ (tablica offer_status)</i>
<b>user_id</b>	int	Not null	<i>Strani ključ (tablica users)</i>

Tablica 7.7 Struktura tablice offer\_applications

Tablica offer\_applications sadrži podatke o poslanoj ponudi. Sadrži iznos ponude i vrijeme slanja ponude, te strane ključeve na tablice sa proizvodima, korisnicima, valutama i statusom ponude.

## 7 IMPLEMENTACIJA SUSTAVA

Koristeći navedene tehnologije i alate razvijena je web aplikacija za aukcije. U ovom dijelu su raspisani koraci do konačne aplikacije.

### 7.1 Konfiguracija baze podataka

Symfony pruža sve alate koji su nam potrebni za rad sa bazama podataka zahvaljujući Doctrine-u. Doctrine je set PHP biblioteka za rad s bazama podataka. Podržava rad s relacijskim bazama poput MySQL i PostgreSQL, a također i NoSQL baze poput MongoDB. Jedna od ključnih značajki Doctrine-a je mogućnost pisanja upita u bazu podataka na Doctrine Query Language (DQL), objektno orijentiranom dijalektu SQL-a. Nakon instalacije Doctrine-a, podaci o povezivanju baze podataka pohranjeni su kao varijabla okruženja u .env datoteci. Potrebno je upisati naziv projekta, url projekta, root lozinku, naziv baze podataka, naziv MySQL korisnika i lozinku MySQL korisnika.

```
PROJECT_NAME=mvc
PROJECT_URL=localhost

DATABASE_ROOT_PASSWORD=root
DATABASE_NAME=mvc
MYSQL_USER=root
MYSQL_PASSWORD=root

DATABASE_URL=mysql://{$MYSQL_USER}:{$MYSQL_PASSWORD}@{$PROJECT_NAME}
-mysql:3306/{$DATABASE_NAME}
```

Kôd 7.2: Konfiguracija baze podataka u .env datoteci

Za rad aplikacije potrebno je napraviti određene tablice u bazi u koje ćemo pohranjivati podatke pri čemu će nam pomoći Doctrine. Pomoću naredbe `make:entity` možemo kreirati novu klasu i dodati koliko god stupaca hoćemo. Prva tablica koja je potrebna za rad aplikacije je tablica u koju ćemo pohranjivati proizvode koji idu na aukciju. Nakon izrade i konfiguriranja klase `ProductApplications` u kojoj će biti spremljeni svi proizvodi, klasa je spremna za spremanje u tablicu u bazu. Također je po potrebi moguće i ažurirati



postojeću klasu na isti način. Zato je potrebno napraviti migraciju sa `make:migration` i izvršiti je sa `doctrine:migrations:migrate`.

Izgled tablice `ProductApplications` prikazan je na slici 7.1.

Name	product_applications		Primary	id					
#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key	comment
1	id	int	NULL	NULL	NO	NULL	auto_increment	EMPTY	→ EMPTY
2	product_name	varchar(225)	utf8mb4	utf8mb4_unicode_ci	NO	NULL	EMPTY	EMPTY	→ EMPTY
3	description	varchar(225)	utf8mb4	utf8mb4_unicode_ci	NO	NULL	EMPTY	EMPTY	→ EMPTY
4	deliverytime	varchar(225)	utf8mb4	utf8mb4_unicode_ci	NO	NULL	EMPTY	EMPTY	→ EMPTY
5	photoname	varchar(255)	utf8mb4	utf8mb4_unicode_ci	NO	NULL	EMPTY	EMPTY	→ EMPTY
6	expires_at	datetime	NULL	NULL	NO	NULL	EMPTY	EMPTY	→ EMPTY
7	currency_id	int	NULL	NULL	YES	NULL	EMPTY	currency(id)	→ EMPTY
8	starting_price	int	NULL	NULL	NO	NULL	EMPTY	EMPTY	→ EMPTY
9	status_id	int	NULL	NULL	YES	NULL	EMPTY	product_status(id)	→ EMPTY
</									

## 7.2 Prikaz proizvoda

Kad korisnik posjeti web aplikaciju on može samo pregledati sve proizvode koji se nalaze na aukciji. Zbog toga je unutar repozitorija za proizvode prvo kreirana funkcija pod nazivom `findAllActive` koja iz tablice sa proizvodima dohvaća sve proizvode kojima nije prošao datum i vrijeme isteka aukcije.

```
public function findAllActive($expiresAt): array{
    $qb = $this->createQueryBuilder('p')
        ->where('p.expiresAt > :expiresAt')
        ->setParameter('expiresAt', $expiresAt)
        ->orderBy('p.expiresAt', 'ASC');

    $query = $qb->getQuery();

    return $query->execute();
}
```

Kôd 7.2: Metoda za dohvat svih aktivnih proizvoda

Radom aplikacije upravljamo iz controllera. Funkciju `findAllActive` pozivamo iz `FrontPageController` da bi na naslovnoj stranici mogli ispisati sve aktivne proizvode koji se nalaze na aukciji.

Najbolji način za organiziranje i generiranje HTML-a aplikacije je uz korištenje predložaka. Predlošci u Symfonyju se kreiraju uz pomoć Twig-a. Twig je brz, fleksibilan i siguran procesor predložaka za PHP. Korištenjem for petlje unutar Twig-a ispisat ćemo sve proizvode iz baze podataka koji se trenutno nalaze na aukciji.

```

class FrontPageController extends AbstractController
{
/**
 * @Route("/", name="frontpage")
 */
public function frontpage()
{
$now = new \DateTime();
$fps = $this->getDoctrine()
->getRepository(ProductApplications::class)
->findAllActive($now);

return $this->render('default/frontpage.html.twig', array('fps'
=> $fps));
}
}

```

Kôd 7.3 : FrontPageController za prikaz proizvoda

```


{% for fp in fps %}
<div class="col-md-4">
    <div class="product-item">
        
        <div class="down-content">
            <h4><strong>{{ fp.productname }} </strong></h4>
            <h7><b>Details:</b> {{ fp.description }}
</h7><br>
            <h7><b>Delivery time:</b> {{ fp.deliverytime }}
</h7><br><br>
            <h7><b>Expires at:</b> {{ fp.expiresAt|date('Y-m-
d H:i:s') }} </h7><br>
            <h7><b>Starting price:</b> {{
(fp.StartingPrice/100)|number_format(2, '.', ',') }}
{{fp.Currency.Currency}} </h7><br><br>

```

Kôd 7.4: Twig template za prikaz proizvoda

### 7.3 Login i registracija

Ruta `/login` je ruta kojoj mogu svi pristupiti. Korisnici mogu birati između prijave u aplikaciju ili registracije korisničkog računa. Login i registracija su određeni sa ugrađenim Symfony mehanizmima za login i registraciju. Kreiran je entitet User i odrađene su migracije na bazu. Ukoliko korisnik nema korisnički račun, potrebno je da prvo odradi proces registracije. Registracija zahtjeva unos korisničkog imena, emaila, imena, prezimena, adrese i lozinke. Izgled forme za registraciju prikazan je na slici 7.3.



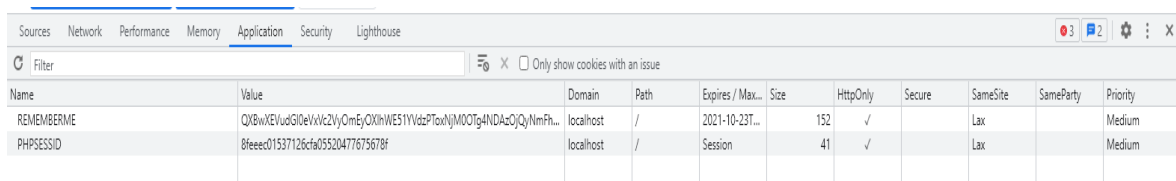
## Register

Username	<input type="text"/>
Email	<input type="text"/>
First name	<input type="text"/>
Last name	<input type="text"/>
Address	<input type="text"/>
Password	<input type="password"/>
Repeat Password	<input type="password"/>

Register!

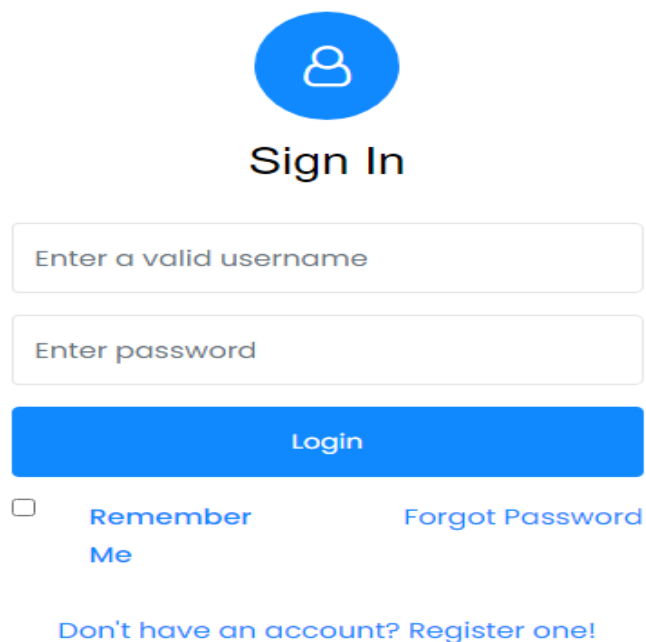
Slika 7.3 Forma za registraciju korisnika


Prilikom prijave u sustav korisnik može odabrati i opciju remember me. Nakon provjere autentičnosti korisnika, njihove vjerodajnice se obično pohranjuju u sesiju. To znači da će po završetku sesije biti odjavljeni te će sljedeći put kad žele pristupiti aplikaciji morati ponovno navesti svoje podatke za prijavu. Korisnicima se može dopustiti da odaberu ostati prijavljeni duže nego što traje sesija pomoću kolačića s opcijom Remember\_me. Kada korisnik prilikom prijave potvrdno označi kvadratić remember me, kolačić će se automatski spremiti. Trajanje kolačića je točno tjedan dana što znači da korisnik može biti prijavljen u sustavu tjedan dana bez ponovne prijave.



Name	Value	Domain	Path	Expires / Max...	Size	HttpOnly	Secure	SameSite	SameParty	Priority
REMEMBERME	QX8wXEvdG0eVkc2VjOmEjQXlhWE51YVdzPToxNjM0OTg4NDAsCjQyNmRh...	localhost	/	2021-10-23T...	152	✓		Lax		Medium
PHPSESSID	8feec01537126cfa05520477675678f	localhost	/	Session	41	✓		Lax		Medium

Slika 7.4 Remember me cookie





**Sign In**

☐ **Remember Me**
[Forgot Password](#)

[Don't have an account? Register one!](#)

Slika 7.5 Forma za login

Korisnici na raspolaganju imaju i opciju reset password u slučaju kada zaborave lozinku. Tada je potrebno da korisnik upiše svoju e-mail adresu i na mail će dobiti link za unos nove lozinke nakon čega će se moći ponovno prijaviti u aplikaciju. Lozinke korisnika se hashiraju pri spremanju u bazu. Za hashiranje se koristi bcrypt algoritam.

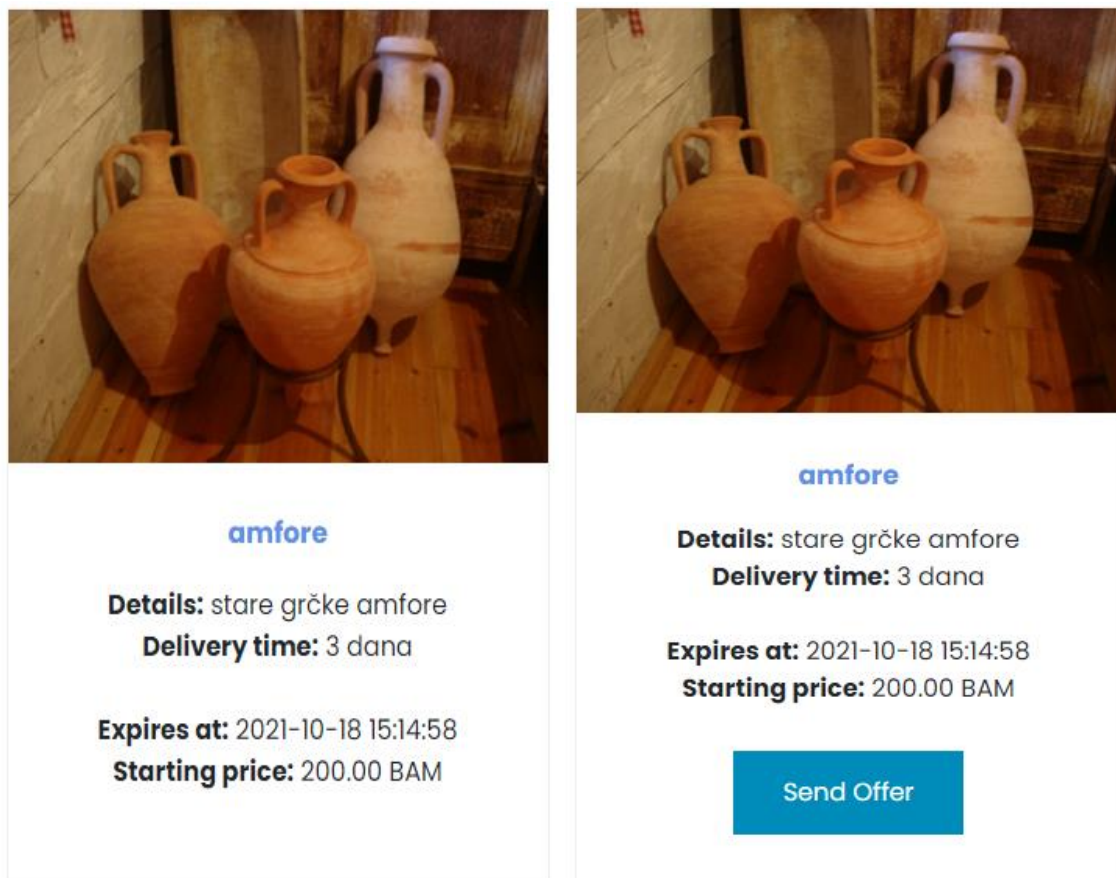
## 7.4 Slanje ponuda

Kada korisnik nije prijavljen on može samo vidjeti proizvode ali ne može poslati ponudu dok se ne prijavi. To se postiže postavljanjem ograničenja unutar Twig templatea za prikaz proizvoda. Gumb send offer se prikazuje samo prijavljenim korisnicima koji imaju ulogu `ROLE_USER`.

```
{% if is_granted('ROLE_USER') %}  
    <a href="{{ path('offer', { 'id': fp.getId() }) }}">  
    <button class="button">Send Offer</button></a>  
    {% endif %}
```

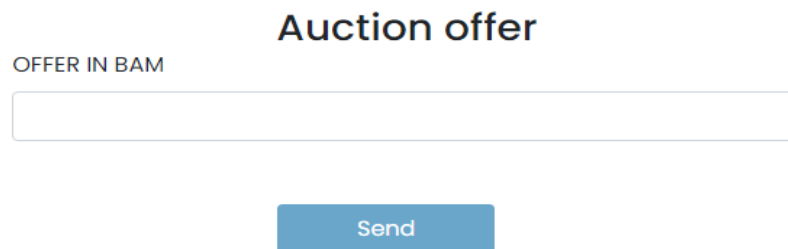
Kôd 7.5 Provjera uloge korisnika

Kako to izgleda u praksi možemo vidjeti na slici 7.6.



Slika 7.6 Prikaz proizvoda za posjetitelje i prijavljene korisnike


Kada korisnik odabere opciju Send Offer otvorit će mu se forma za slanje ponude koju vidimo na slici 7.7.



Slika 7.7 Forma za slanje ponude


Zbog mogućih problema sa zaokruživanjem kod korištenja decimalnih brojeva, iznos ponude se u bazi sprema kao cijeli broj tako što se pri spremanju množi sa 100, a pri ispisu dijeli sa 100. Tako se iznos u BAM sprema u feninzima, EUR u centima itd.

Osim iznosa ponude u tablici za ponude se spremaju i drugi podaci kao što je točan datum i vrijeme slanja ponude, vrsta valute, proizvod za koji se šalje ponuda i podaci o korisniku koji šalje ponudu. Vrsta valute je ista ona koja je postavljena za početnu cijenu proizvoda, a status ponude je postavljen na „Open“. Program također provjerava i da li je iznos ponude veći od početne cijene proizvoda. Ako nije, korisnik će dobiti obavijest da iznos ponude mora biti veći od početne cijene, a ako je onda će se ponuda upisati u bazu, a korisnik će o tome biti obaviješten mailom i flash porukom.



Offer must be greater than the starting price!

Slika 7.8 Warning flash poruka



Offer successfully sent!

Slika 7.9 Success flash poruka

```

/**
 * @Route("/offer/send/{id}", methods={"POST"})
 */
public function create($id, EntityManagerInterface $em, Request $request, MailerServiceInterface $mailerServiceInterface){

    if(!$request->get('offer')){

        $this->addFlash('offer', 'Offer field must be filled!');
    }else{

        $user_id = $this->getUser();
        $email = $user_id->getEmail();
        $offer = $request->get('offer')*100;
        $time = new \DateTime();
        $status = $this->getDoctrine()
            ->getRepository(OfferStatus::class)
            ->findOneBy(array('offer_status' => 'Open'));
        $product_id = $this->getDoctrine()
            ->getRepository(ProductApplications::class)
            ->find($id);
        $currency = $product_id->getCurrency();
        $starting_price = $product_id->getStartingPrice();
        $currency_name = $currency->getCurrency();

        if (!$offer) {
            $this->addFlash('offer', 'Offer field must be filled!');
        }
        else if($offer < $starting_price){
            $this->addFlash('offer', 'Offer must be greater than the starting price!');
        }
        else {

            $of = new OfferApplications();
            $of->setUser($user_id);
            $of->setOffer($offer);
            $of->setTime($time);
            $of->setCurrency($currency);
            $of->setStatus($status);
            $of->setProduct($product_id);
            $em->persist($of);
            $em->flush();

            $mailerServiceInterface->send(
                "auctions@mail.com",
                $email,
                'Offer submitted!',
                "email/offer_response.html.twig", ["offer" => $offer, "currency_name" => $currency_name]
            );
            $this->addFlash('offer-sent', 'Offer successfully sent!');
        }
    }
}

```

Slika 7.10 Metoda za slanje ponuda



## 7.5 Servis za slanje mailova

Za stvaranje servisa za slanje mailova prvo je potrebno instalirati Symfony Mailer. Mailer je moćan alat za stvaranje i slanje e-pošte s podrškom za integraciju Twig-a, privitke datoteka i još mnogo toga. Unutar MailerService.php datoteke kreiramo metodu send koja prima parametre: adresu pošiljatelja, adresu primatelja, naslov poruke, sadržaj poruke i HTML template.

```
public function send(string $from, string $to, string $subject,
    string $template, array $message): void
{
    try {
        $email = (new Email())
            ->from($from)
            ->to($to)
            ->subject($subject)
            ->html(
                $this->twig->render($template, $message),
                'text/html'
            );

        $this->mailer->send($email);
    } catch (TransportException $e) {
        print $e->getMessage()."\n";
        throw $e;
    }
}
```

Kôd 7.6 Servis za slanje mailova

Metoda se poziva preko interfecea MailerServiceInterface.

```
interface MailerServiceInterface
{
    public function send(string $from, string $to, string $subject,
        string $template, array $message): void;
}
```

Kôd 7.7 Interface za servis za slanje mailova

## 7.6 Admin panel

Administratori aplikacije se razlikuju od običnih korisnika po svojoj ulozi. Administratori imaju ulogu `ROLE_ADMIN` i mogu pristupiti admin panelu kojemu obični korisnici nemaju pristup. Postoji više načina na koje se može ograničiti pristup korisnicima određenim dijelovima aplikacije.

Unutar Twig templatea možemo postaviti uvjet da određene dijelove stranice može vidjeti samo korisnik s određenom ulogom.

```
{% if is_granted('ROLE_ADMIN') %}
    #HTML KOD
{% endif %}
```

Kôd 7.8 Ograničenje pristupa unutar twig templatea

Unutar `security.yaml` datoteke možemo postaviti ograničenje na rute kojima samo korisnici s određenom ulogom mogu pristupiti. S konfiguracijom u kodu 7.9 samo korisnici s ulogom `ROLE_ADMIN` mogu pristupiti putanji `/home`.

```
access_control:
    - { path: ^/home, roles: ROLE_ADMIN}
```

Kôd 7.9 Security.yaml ograničenje pristupa

Osim toga, ograničenja se mogu postaviti i unutar samog controllera tako da blokiraju pristup određenoj ruti ako nije zadovoljena određena uloga što možemo vidjeti u kodu 7.10.

```
$this->denyAccessUnlessGranted('ROLE_ADMIN');
```

Kôd 7.10 Ograničenje pristupa unutar controllera

### 7.6.1 Administracija ponuda

Jedna od najvažnijih zadaća administratora je upravljanje pristiglim ponudama. Tablica sa ponudama OfferApplications je povezana sa tablicom OfferStatus koja opisuje u kakvom je statusu svaka ponuda. Po defaultu svaka pristigla ponuda prvo ima status 'Open' što znači da je otvorena, tj. da nije niti prihvaćena niti odbijena. Osim toga, ponuda može imati status 'Accepted' ako je prihvaćena, i 'Rejected' ako je odbijena. Sve pristigle ponude se prvo nalaze u stupcu 'Open' nakon čega ih administrator može ili prihvatiti ili odbiti. Osim Symfonya, za implementaciju ovih funkcionalnosti koristio se i JavaScript.

```
if (e.target.className === 'btn btn-danger reject-article') {
    if (confirm('Are you sure?')) {
        const id = e.target.getAttribute('data-id');

        fetch(`/home/reject/${id}`, {
            method: 'POST'
        }).then(res => window.location.reload());
    }
}

if (e.target.className === 'btn btn-success accept-article')
{
    if (confirm('Are you sure?')) {
        const id = e.target.getAttribute('data-id');

        fetch(`/home/accept/${id}`, {
            method: 'POST'
        }).then(res => window.location.reload());
    }
}
```

Kôd 7.11 JavaScript kod za slanje zahtjeva na backend

Na backendu zahtjev stiže na POST rutu `/home/reject/{id}` ako želimo odbiti ponudu, ili na `/home/accept/{id}` ako želimo prihvatiti ponudu. Kod 7.12 predstavlja backend dio za odbijanje ponude.

```
/**
 * @Route("/home/reject/{id}", methods={"POST"})
 *
 */
public function reject($id, ResponseService $responseService) {
    $this->denyAccessUnlessGranted('ROLE_ADMIN');
    $entityManager = $this->getDoctrine()->getManager();
    $rejected = $entityManager->getRepository(OfferApplications::class)->find($id);

    if (!$rejected) {
        $responseService->responseCode(Response::HTTP_NOT_FOUND,
        'Application Not Found');
    }

    $status = $this->getDoctrine()
        ->getRepository(OfferStatus::class)
        ->findOneBy(array('offer_status' => 'Rejected'));
    $rejected->setStatus($status);
    $entityManager->persist($rejected);
    $entityManager->flush();
    return $responseService->responseCode(Response::HTTP_OK, 'Offer
    rejected!');
}
```

Kôd 7.12 Backend dio za odbijanje ponude

Kod 7.13 predstavlja backend dio za prihvaćanje ponude. Ako je ponuda prihvaćena korisnik će dobiti obavijest o tome na svoju mail adresu.

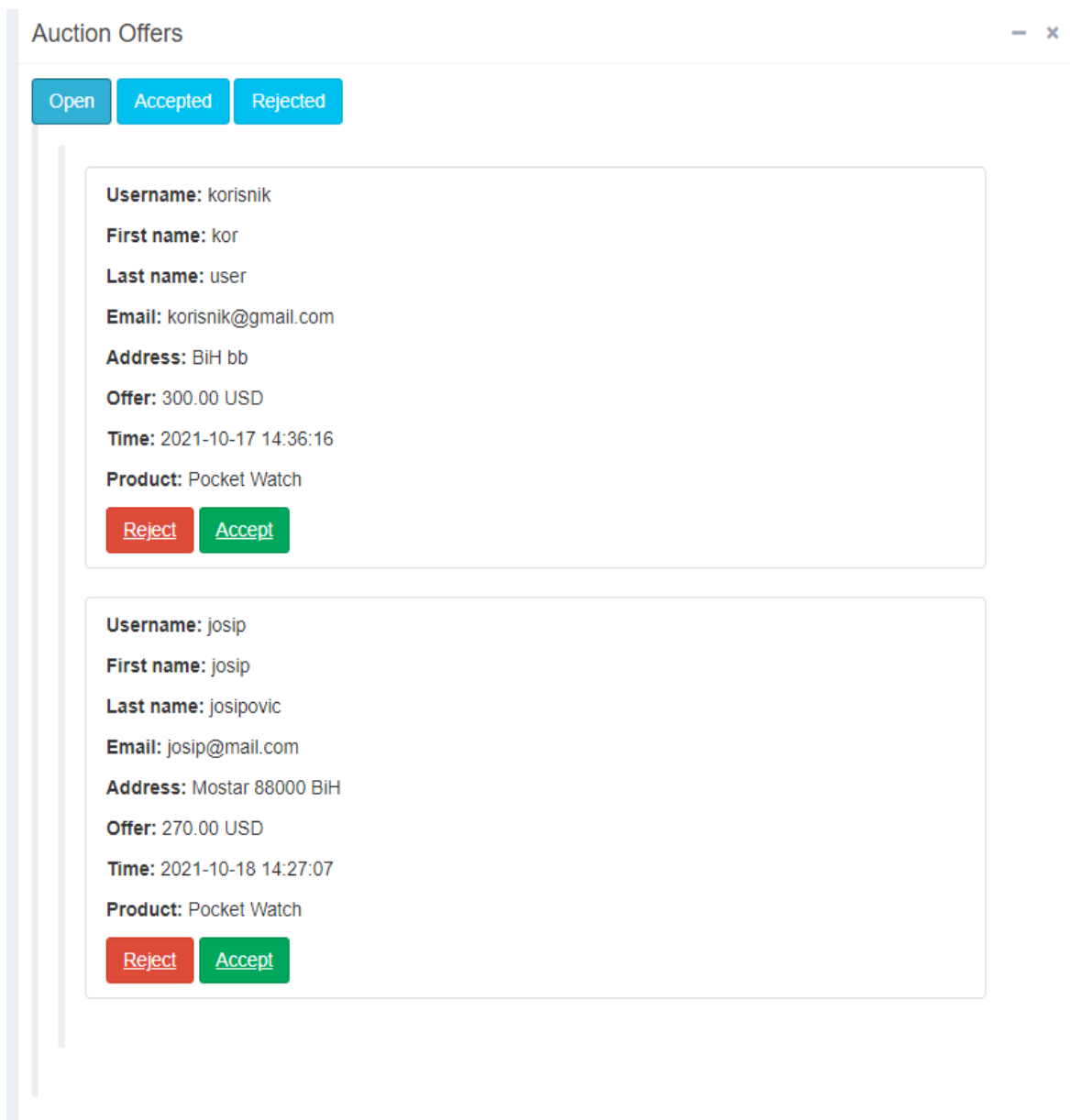
```
/**
 * @Route("/home/accept/{id}", methods={"POST"})
 */
public function archive($id, ResponseService
    $responseService, MailerServiceInterface $mailerServiceInterface)
    {
        $this->denyAccessUnlessGranted('ROLE_ADMIN');
        $entityManager = $this->getDoctrine()->getManager();
        $active = $entityManager-
            >getRepository(OfferApplications::class)->find($id);

        if (!$active) {
            $responseService->responseCode(Response::HTTP_NOT_FOUND,
                'Application Not Found');
        }
        $email = $active->getUser()->getEmail();
        $first_name = $active->getUser()->getFirstName();
        $offer = $active->getOffer();
        $currency_name = $active->getCurrency()->getCurrency();
        $status = $this->getDoctrine()
            ->getRepository(OfferStatus::class)
            ->findOneBy(array('offer_status' => 'Accepted'));
        $active->setStatus($status);
        $entityManager->flush();

        $mailerServiceInterface->send(
            "auctions@mail.com",
            $email,
            'Offer Accepted!',
            "email/offer_accepted.html.twig", ["first_name" =>
                $first_name, "offer" => $offer, "currency_name" =>
                $currency_name]
        );
    }
}
```

Kôd 7.13 Backend dio za prihvaćanje ponude

Slika 7.11 predstavlja izgled admin panela za administraciju ponuda. Ponude su podijeljene u tri stupca: otvorene, prihvaćene i odbijene.



Slika 7.11 Ekran za administraciju ponuda

### 7.6.2 Administracija proizvoda

Administrator ima zadaću dodavanja novih proizvoda koji idu na aukciju, ali i administraciju nad proizvodima koji se već nalaze na aukciji ili su završili sa aukcijom. Prilikom dodavanja novog proizvoda potrebno je popuniti formu koja se sastoji od naziva proizvoda, opisa proizvoda, vremena dostave, točnog datuma i vremena isteka aukcije, početne cijene proizvoda, valute u kojoj se primaju ponude i unosa slike proizvoda. Slika 7.12 prikazuje formu za unos novog proizvoda.

Insert Product

**PRODUCT NAME**

**DESCRIPTION**

**DELIVERY TIME**

**EXPIRES AT**

dd.mm.gggg. --:--:--

**STARTING PRICE**

**CURRENCY**

EUR

**UPLOAD PHOTO**

Odaberi datoteku Nije odabrana niti jedna datoteka.

Add product

Slika 7.12 Forma za unos novog proizvoda

Prilikom unosa slike potrebno je uraditi validaciju. Prvo sa javascriptom na frontendu provjeravamo ekstenziju slike da vidimo da li se radi o image datoteci, a nakon toga provjeravamo i da li veličina slike prekoračuje dopuštenu maksimalnu veličinu koja iznosi 5 Mb po slici. Sadržaj javascript koda za validaciju prikazan je u kodu 7.14. Ako je validacija uspješna, slika i sadržaj cijele forme za unos proizvoda ide na backend.

```

document.getElementById("photoName").addEventListener("change",
    validateFile)

function validateFile(){
    const allowedExtensions = ['jpg','png','jpeg'],
        sizeLimit = 5000000;

    const { name:file, size:fileSize } = this.files[0];

    const fileExtension = file.split(".").pop();

    if(!allowedExtensions.includes(fileExtension)){
        alert("Please upload only photo files!");
        this.value = null;
    }else if(fileSize > sizeLimit){
        alert("File size too large!")
        this.value = null;
    }
}

```

Kôd 7.14 JavaScript validacija slike

Na backend dijelu unosa novog proizvoda provjeravamo da li su sva polja u formi popunjena, te ako jesu dohvaćamo njihov sadržaj. Početnu cijenu proizvoda množimo sa 100 jer ćemo novac spremati kao cijeli broj , a status proizvoda postavljamo kao 'Active' sve dok mu vrijeme isteka aukcije ne prođe. Zatim ponovno provjeravamo ekstenziju slike i ako je ispravna njezin naziv ćemo spremiti u bazu podataka, a samu sliku ćemo spremiti u folder images. Ukoliko je proizvod uspješno spremljen u bazu, administrator će dobiti obavijest u obliku flash poruke. Backend dio unosa slike prikazan je u kodu 7.15.



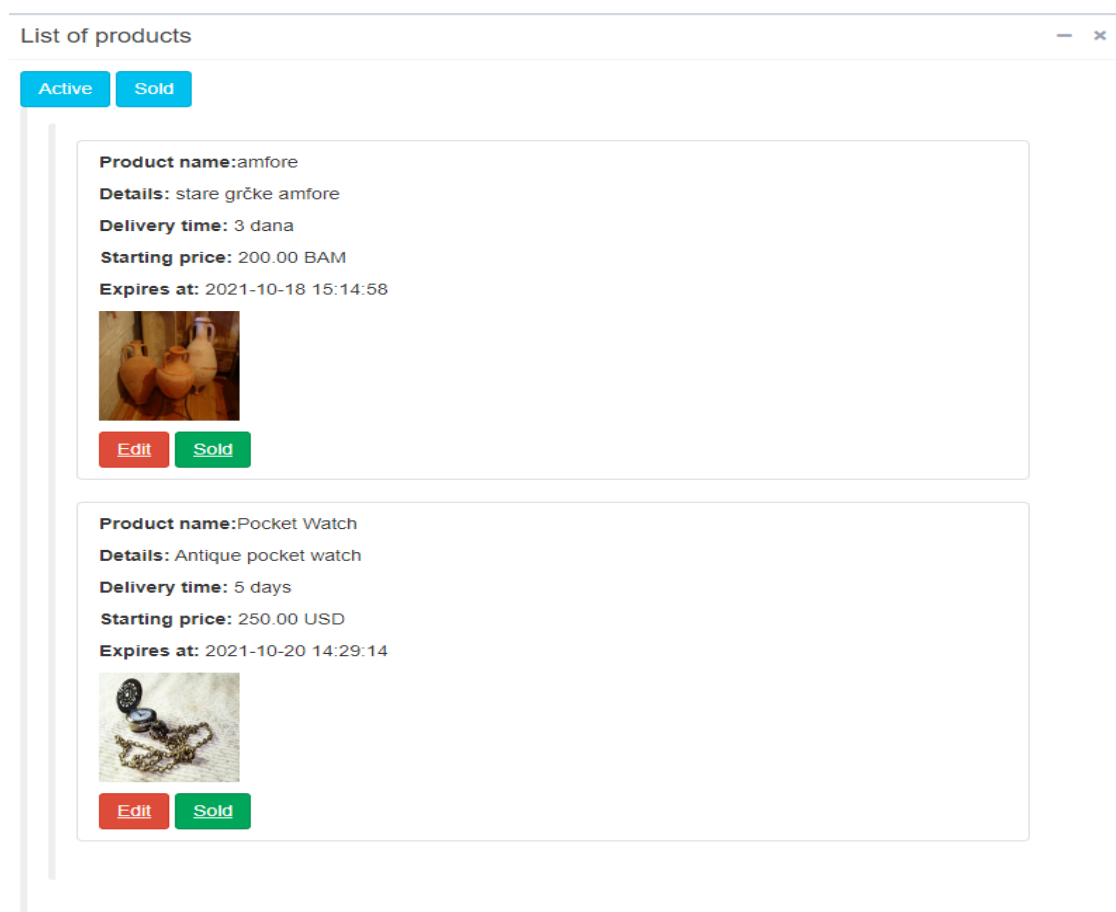
```

if($uploadedPhoto->getClientOriginalExtension() === "png" ||
    $uploadedPhoto->getClientOriginalExtension() === "jpg" ||
    $uploadedPhoto->getClientOriginalExtension() === "jpeg"){
    if (!$product_name || !$description || !$deliverytime ||
        !$expires_at || !$starting_price || !$currency ||
        !$uploadedPhoto) {
        $this->addFlash('warning','All fields must be
filled!');
    } else {
        $destination = $this->getParameter('kernel.project_dir') .
'/public/images';
        $originalPhotoname = pathinfo($uploadedPhoto-
>getClientOriginalName(), PATHINFO_FILENAME);
        $newPhotoname = $originalPhotoname . $uploadedPhoto-
>getClientOriginalExtension();
        try {
            $uploadedPhoto->move(
                $destination,
                $newPhotoname
            );
        } catch (Exception $e) {
            $this->addFlash('error','Internal server
error!');
        }
    }
}

```

Kôd 7.15 Backend dio za unos slike

Osim unosa novih proizvoda administrator mora voditi računa i o postojećim proizvodima koji se nalaze na aukciji. Svi proizvodi koji se nalaze na aukciji imaju status 'Active', a oni proizvodi kojima je isteklo vrijeme aukcije imaju status 'Sold' i nalaze se u drugom stupcu što je prikazano na slici 7.13. Proizvodi automatski mijenjaju svoj status nakon što prođe datum i vrijeme završetka aukcije, no administrator ima i mogućnost da sam promijeni status proizvoda odabirom gumba Sold. Osim toga administrator ima i mogućnost izmjene informacija o proizvodu odabirom gumba Edit što je prikazano na slici 7.14.



Slika 7.13 Pregled proizvoda

### Edit Product

Product name	amfore
Description	stare grčke amfore
Deliverytime	3 dana
Expires at	<div>Oct</div> <div>18</div> <div>2021</div> <div>15</div> <div>14</div>
Photo	<div>Odaberi datoteku</div> <div>amfore.jpg</div>
Starting price	200
Currency	BAM

Update

Slika 7.14 Ažuriranje proizvoda

### 7.6.3 Administracija korisnika

Posao administratora je i administracija nad korisnicima koji pristupaju aplikaciji. Administrator ima mogućnost kreiranja novih administratora koji mu mogu pomoći u obavljanju posla. Potrebno je samo da ispuni formu s informacijama o korisniku, a program će mu postaviti ulogu admina i lozinku koja se sastoji od slučajno odabranih brojeva. Novi administrator će podatke o lozinki i korisničkom računu dobiti na svoju mail adresu.



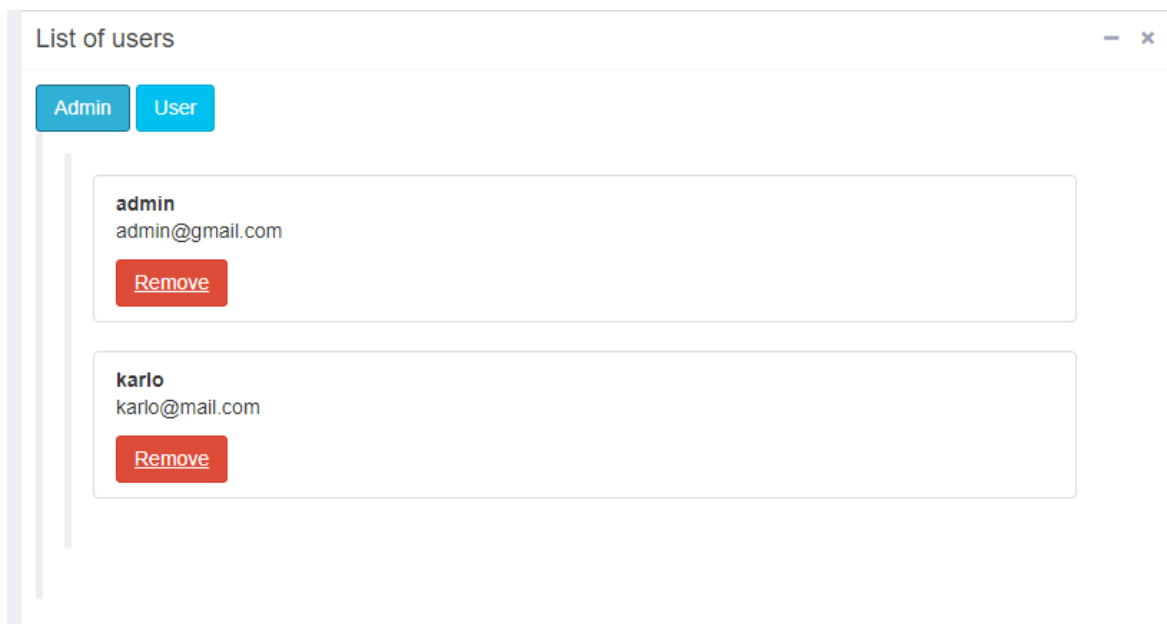
The image shows a web form titled "Add User". It contains five text input fields labeled "Username", "Email", "First name", "Last name", and "Address". Below these fields is a blue button with the text "Add User".

Slika 7.15 Kreiranje novog administratora

```
$user = new User();  
$form = $this->createForm(UserType::class, $user);  
$form->handleRequest($request);  
if ($form->isSubmitted() && $form->isValid()) {  
    $pass = random_int(10000000, 1000000000);  
    $password = $passwordEncoder->encodePassword($user,  
$pass);  
    $user->setPassword($password);  
    $user->setRoles(["ROLE_ADMIN"]);  
  
    $entityManager = $this->getDoctrine()->getManager();  
    $entityManager->persist($user);  
    $entityManager->flush();  
}
```

Kôd 7.16 Kreiranje novog administratora

Administrator ima uvid u popis svih administratora i običnih korisnika. Ima i mogućnost oduzimanja privilegija određenom administratoru čime bi on postao običan korisnik koji bi mogao slati ponude za proizvode ali ne bi imao pristup admin panelu. Odabirom opcije Remove program pronalazi id odabranog korisnika i mijenja njegovu rolu iz `ROLE_ADMIN` u `ROLE_USER` što nam prikazuje slika 7.16.



Slika 7.16 Popis korisnika

Korisnici još imaju i mogućnost promjene lozinke gdje je potrebno da prvo unesu postojeću lozinku, a zatim i dva puta novu lozinku. Program provjerava da li je stara lozinka ispravna i da li je nova lozinka dva puta ispravno upisana.

The image shows a web application window titled "Change Password". It contains three input fields: "Password", "New password", and "Repeat Password". Below the fields is a blue button labeled "Change password". The form is simple and functional, with a light gray background and rounded corners.

Slika 7.17 Forma za promjenu lozinke

## 8 ZAKLJUČAK

Projekt razvijen i opisan u ovom radu je izrađen za potrebe diplomskog rada, ali nadogradnja projekta se može i nastaviti. Ovaj projekt se ubraja u područje elektroničkog poslovanja koje je zadnjih godina zabilježilo ogroman rast zahvaljujući razvoju interneta. Svakog dana razvijaju se nove web aplikacije za određene poslovne projekte. Glavni razlog popularnosti web aplikacija leži u tome da su dostupne s bilo kojeg uređaja koji je povezan na internet.

Pri izradi su korištene popularne web tehnologije današnjice koje su nam omogućile izradu funkcionalne aplikacije i jasnog i razumljivog korisničkog sučelja. Symfony okvir nam je omogućio jednostavnu implementaciju serverskog dijela aplikacije, MySQL implementaciju baze podataka, a Docker kontejneri laganu prenosivost na druga računala i operacijske sustave. Za dizajn aplikacije su korišteni HTML, CSS, JavaScript i njezine biblioteke kao što su jQuery i Bootstrap. Izradi aplikacije je prethodila izrada dokumentacije i jasno definiranje svih zahtjeva koje aplikacija mora ispunjavati.

Uspješnoj implementaciji projekta prethodilo je iskustvo stečeno pri izradi projekata tijekom studiranja i na studentskoj praksi koje su mi omogućile upoznavanje sa navedenim programskim jezicima, okvirima i alatima za izradu aplikacija.

# LITERATURA

- [1] <https://www.indeed.com/career-advice/career-development/what-is-web-application>
- [2] <https://aplextor.medium.com/a-brief-history-of-web-app-50d188f30d>
- [3] <https://www.devsaran.com/blog/history-web-application-development>
- [4] <https://msatechnosoft.in/categories-of-web-applications-characteristics-of-web-applications/>
- [5] <https://medium.com/laravel-power-devs/web-programming-with-php-b6c96d187070>
- [6] <https://www.ideamotive.co/blog/php-vs-java-the-best-choice-for-web-development>
- [7] <https://symfony.com/at-a-glance>
- [8] <https://symfony.com/six-good-technical-reasons>
- [9] <https://www.tutorialspoint.com/mysql/mysql-introduction.htm>
- [10] <https://frontendmasters.com/books/front-end-handbook/2018/what-is-a-FD.html>

## POPIS SLIKA:

Slika 2.1 Povijest razvoja web aplikacija.....	4
Slika 2.2 PHP kao skriptni jezik na strani poslužitelja .....	6
Slika 2.3 PHP logo .....	7
Slika 2.4 Symfony framework logo .....	8
Slika 2.5 Shema Docker kontejnera .....	11
Slika 2.6 Docker pokretač.....	12
Slika 2.7 Frontend tehnologije: GTML, CSS, JavaScript .....	13
Slika 2.8 HTML i CSS.....	15
Slika 2.9 HTML, CSS i JavaScript .....	16
Slika 3.1 Korišteni alati.....	18
Slika 4.1 Use case dijagram .....	21
Slika 6.1 Relacijski model baze podataka .....	27
Slika 7.1 Tablica ProductApplications .....	34
Slika 7.2 Primjer podataka u tablici .....	34
Slika 7.3 Forma za registraciju korisnika.....	37
Slika 7.4 Remember me cookie.....	38
Slika 7.5 Forma za login.....	38
Slika 7.6 Prikaz proizvoda za posjetitelje i prijavljene korisnike .....	39
Slika 7.7 Forma za slanje ponuda .....	40
Slika 7.8 Warning flash poruka .....	40
Slika 7.9 Success flash poruka.....	40
Slika 7.10 Metoda za slanje ponuda.....	41
Slika 7.11 Ekran za administraciju ponuda .....	47
Slika 7.12 Forma za unos novog proizvoda .....	48
Slika 7.13 Pregled proizvoda .....	51
Slika 7.14 Ažuriranje proizvoda .....	51
Slika 7.15 Kreiranje novog administratora.....	52
Slika 7.16 Popis korisnika .....	53
Slika 7.17 Forma za promjenu lozinke .....	53

## POPIS TABLICA:

Tablica 7.1 Struktura tablice users .....	28
Tablica 7.2 Struktura tablice reset_password_request .....	29
Tablica 7.3 Struktura tablice currency .....	29
Tablica 7.4 Struktura tablice product_applications.....	30
Tablica 7.5 Struktura tablice product_status .....	31
Tablica 7.6 Struktura tablice offer_status .....	31
Tablica 7.7 Struktura tablice offer_applications .....	32



## POPIS KODOVA:

Kod 5.1 Dockerfile za PHP .....	22
Kod 5.2 Docker-compose.yml konfiguracija PHP kontejnera .....	23
Kod 5.3 Nginx.conf file .....	24
Kod 5.4 Docker-compose.yml konfiguracija Nginx kontejnera .....	25
Kod 5.5 Docker-compose.yml konfiguracija MySQL kontejnera.....	25
Kod 5.6 Docker-compose.yml konfiguracija za MailCatcher .....	26
Kod 7.1 Konfiguracija baze podataka u .env datoteci .....	33
Kod 7.2 Metoda za dohvat svih aktivnih proizvoda .....	35
Kod 7.3 FrontPageController za prikaz proizvoda .....	36
Kod 7.4 Twig template za prikaz proizvoda .....	36
Kod 7.5 Provjera uloge korisnika .....	39
Kod 7.6 Servis za slanje mailova.....	42
Kod 7.7 Interface za servis za slanje mailova .....	42
Kod 7.8 Ograničenje pristupa unutar twig templatea .....	43
Kod 7.9 Security.yaml ograničenje pristupa .....	43
Kod 7.10 Ograničenje pristupa unutar controllera .....	43
Kod 7.11 JavaScript kod za slanje zahtjeva na backend .....	44
Kod 7.12 Backend dio za odbijanje ponuda .....	45
Kod 7.13 Backend dio za prihvatanje ponuda .....	46
Kod 7.14 JavaScript validacija slike .....	49
Kod 7.15 Backend dio za unos slike .....	50
Kod 7.16 Kreiranje novog administratora .....	52

## SKRAČENICE:

- **WWW** – World Wide Web
- **W3C** – World Wide Web Consortium
- **JSON** – Javascript Object Notation
- **API** – Application User Interface
- **IDE** – Integrated Development Environment
- **CLI** – Command Line Interface
- **PDO** – PHP Data Objects
- **b2b** – business to business
- **HTTP** – Hypertext Transfer Protocol
- **FPM** – FastCGI Process Manager
- **RDBMS** – Relational Database Management System
- **HTML** – HyperText Markup Language
- **CSS** – Cascading Style Sheets
- **CRUD** – Create, Read, Update, Delete
- **JVM** – Java Virtual Machine
- **LAMP**– Linux, Apache, MySQL, PHP/Perl/Python