

**UNIVERZITET SINGIDUNUM
FAKULTET INFORMATIKE I RAČUNARSTVA**

**Automatizacija procesa obrade podataka i objave oglasa
3D štampe**

- DIPLOMSKI RAD -

Mentor:
Prof. Dr. Marko Šarac

Kandidat:
Pavle Kuzmanović

Sastav

Uvod.....	4
1. Priprema	6
1.1 Arhitektura softvera.....	6
1.1.1 Use Case dijagram	6
1.1.2 Dijagram aktivnosti.....	7
1.2 Tehnologije.....	8
1.2.1 Python	9
1.2.1.1 Šta je Python?.....	9
1.2.1.2 Popularnost.....	9
1.2.1.3 Primene.....	10
1.2.1.4 Naša primena.....	10
1.2.1.5 Iskorišćeni moduli	10
1.2.2 HTML	12
1.2.2.1 Šta je HTML?	12
1.2.2.2 Primena.....	12
1.2.2.3 Elementi	12
1.2.2.4 Naša primena HTML elemenata	13
1.2.3 CSS	14
1.2.3.1 Šta je CSS?	14
1.2.3.2 Primena.....	14
1.2.3.3 Klase.....	15
1.2.3.4 ID Selektor	15
1.2.4 JavaScript.....	16
1.2.4.1 Šta je JavaScript	16
1.2.4.2 Primena.....	16
1.2.4.3 Dodaci	16
1.2.4.4 Frameworks	17
1.2.5 PHP	18
1.2.5.1 Šta je PHP?.....	18
1.2.5.2 Primena.....	18
1.2.5.3 Frameworks.....	19
1.2.6 MySQL	20

1.2.6.1 Šta je SQL	20
1.2.6.2 Šta je MySQL.....	20
1.2.6.3 Šta su relacione baze podataka.....	20
1.3 Alati	21
1.3.1 Visual Studio Code	21
1.3.2 MySQL Workbench.....	22
1.3.3 XAMPP.....	22
2. Izrada.....	23
2.1 Softver za automatizaciju	23
2.1.1 Izrada glavnog modula.....	24
2.1.2 Izrada modula za sajt koji sadrži 3D modele	27
2.1.3 Izrada modula za sajt prodaje	33
2.1.4 Kreiranje .exe fajla za naš softver.....	36
2.2 Baza podataka.....	37
2.3 Web Portal	38
2.3.1 Stranica za logovanje	38
2.3.2 Konzola	40
Zaključak	44

Uvod

Firme, korporacije i ostali tipovi uspešnih biznisa su uvek težili standardizaciji što više procesa radi ubrzanja izvršenja svakodnevnih poslova. Ona omogućava i dodatno olakšava pronalazak i obuku radnika, dodelu adekvatnih resursa, merenje učinka, kao i poveću uštedu svakom biznisu ili pojedincu koji je prisvoji. Prvobitna analiza procesa koji se odvijaju tokom rada i kasnija standardizacija njih je *kritična* za potpunu automatizaciju primenom mašina, programa i ostalih izvora funkcija za istu. Automatizacija je primena mašina na zadatke koju su nekad obavljala ljudska bića ili na zadatke koje bi inače bili nemogući. (Groover, n.d.)

U ovom radu razrađujemo potpunu standardizaciju i automatizaciju celokupnog procesa prikupljanja potrebnih datoteka i informacija vezano za modele 3D štampe pretežno sa sajta “*Thingiverse*”, njihovu izradu, kao i objavu oglasa pojedinačnih modela na internet sajtove za prodaju, specifično sajt “KupujemProdajem” koji *težinski* spada u *Consumer-To-Consumer* tip prodaje. Dodatno razrađujemo i pribavljanje podataka i informacija o samim proizvodima, kupcima i kupoprodajom radi moguće kasnije analize.

Cilj rešavanja pomenutog problema jeste ubrzanje i umanjivanje ili potpuno uklanjanje potrebe za ljudskim inputom tokom izvršavanja inače sporog ili repetitivnog procesa, kao i pribavka velike količine podataka radi efikasnog praćenja statistike i lakše analize radi kasnije optimizacije i nivelisanja rizika prodaje. Dalje gledano, tražimo modularno i skalabilno rešenje koje bi *lakom* kasnijom doradom moglo biti primenjeno na više različitih medija prodaje i omogućava pribavljanje i obradu 3D modela i njihovih informacija sa različitih sajtova koji nude tu uslugu.

Radi realizacije prethodno napomenutog, koristićemo tehnologije iz oblasti *frontend-a*, *backend-a*, i baza podataka radi izgradnje našeg ekosistema. Približnije, što se tiče samog softvera za automatizaciju, koristićemo programski jezik *Python* radi lake modularizacije, skriptinga i primena raznolikih biblioteka potrebne za uspešnu izradnju našeg zadatog posla. Za skladištenje pribavljenih i obrađenih podataka koristićemo *MySQL* pokrenut preko lokalnog servera. Za pristup, pregled i izmene nad našim upisanim podacima izgradićemo web portal koristeći *frontend* tehnologije *HTML5*, *CSS3* i *Javascript* sa integracijom *backend* dela koji pretežno služi i za samu komunikaciju sa našom bazom podataka, *PHP*. Naš portal će biti *host-ovan* na Apache lokalnom serveru uz obavezan algoritam logovanja korisničkim parametrima radi dodatne mere zaštite informacija i strukture podataka.

Kako bi dodatno olakšali celokupni proces realizacije, prvobitno je moguće izraditi plan naše aplikacije. Za izradu nama potrebnih dijagrama, koristićemo IBM-ov softverski paket “*Rational Software Architect*”. Primenom *Use Case* dijagrama iz prethodno napomenutog programa olakšavamo dizajn samog sistema koji izrađujemo, gde je moguće objasniti koje složene aktivnosti korisnik ili administrator može vršiti nad našom aplikacijom. Pritom i sama standardizacija procesa planiranih za automatizaciju može biti razrađena u potpunosti primenom dijagrama aktivnosti gde se svaka aktivnost prikazuje ponaosob kao sklop celokupnog. Sa obzirom da se naš planiran softverski deo za automatizaciju većinski sadrži iz skripti preko više modula, dijagram aktivnosti je i više nego dovoljan za opis celokupnog toka izrade funkcije.

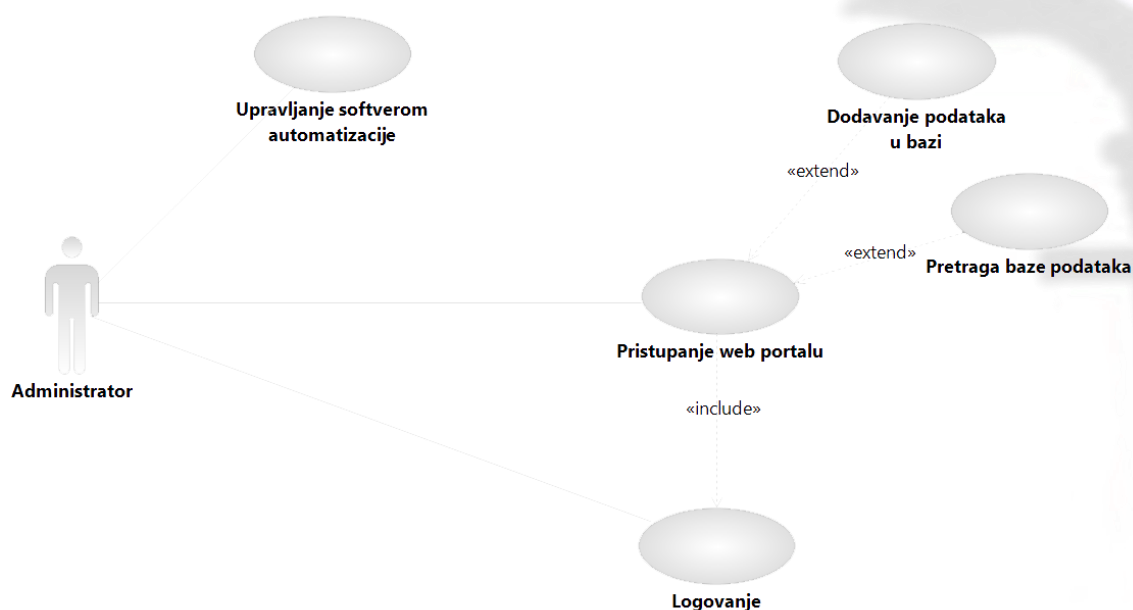
1. Priprema

1.1 Arhitektura softvera

Prvobitno želimo da predstavimo kakve interakcije i koje akcije administrator vrši nad softverom kao i na ostalim delovima našeg sistema u kome, pored ostalog, spada i naš web portal. Uz to, želimo da predstavimo i tok aktivnosti koji se odvija tokom procesa automatizacije.

1.1.1 Use Case dijagram

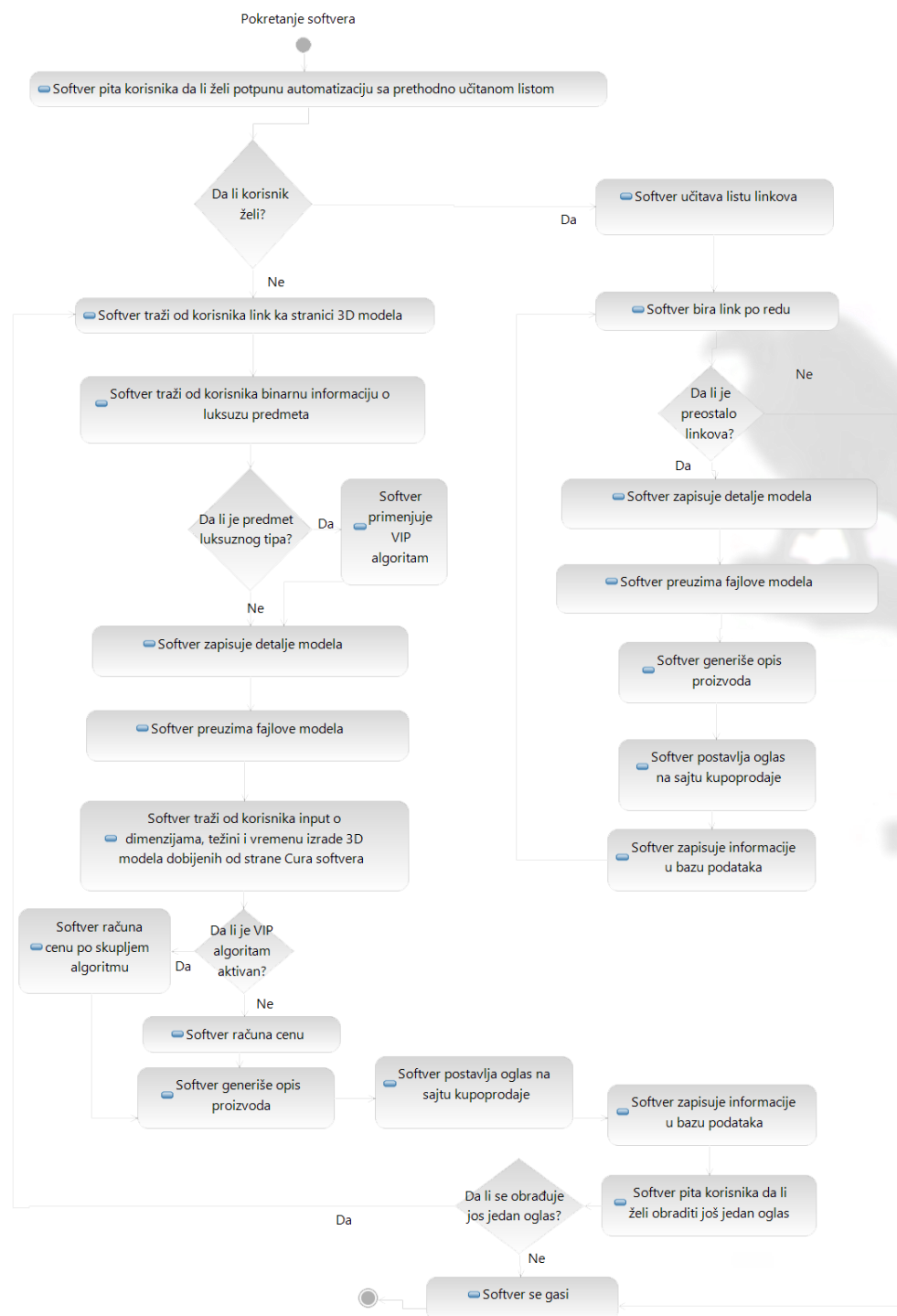
U našem Use Case dijagramu, korisnik ili administrator ima mogućnost upravljanja softverom automatizacije i pristupanje web portalu sa obaveznim prethodnim logovanjem adekvatnim parametrima. Preko našeg portala se dodatno mogu vršiti funkcije pretrage i dodavanje novih zapisa u bazu podataka.



Slika 1 – Use Case dijagram

1.1.2 Dijagram aktivnosti

Dijagram aktivnosti opisuje tok akcija koje se odvijaju tokom izrade poslova softvera.



Slika 2 – Dijagram aktivnosti

1.2 Tehnologije

Nakon izrade *UseCase* dijagrama i dijagrama aktivnosti, možemo lakše odlučiti koje bismo tehnologije koristili prilikom realizacije našeg sistema. U našem slučaju, glavnu rolu procesa automatizacije na standardizovane procese vršiće programski jezik Python verzije 3.x. Python je dobar izbor u našem slučaju speicifično iz razloga što nam je potreban modularan pristup rešenju radi lakše kasnije nadogradnje, kao i što je namena da se proces odvija kao skripta iliti “od gore na dole”. Dodatno, nisu nam potrebni nikakvi grafički interfejsi – ili *GUI*, već nam je dovoljna jednostavna kontrola preko interfejsa komandne linije – u daljem tekstu *CLI*.

Za skladištenje podataka nam je potrebna relaciona baza podataka. Za izradu iste, koristićemo *MySQL* tehnologiju preko softvera za *GUI* realizaciju baza podataka – *MySQL Workbench*. Preko tog programa možemo na lakši način izgraditi naše skladište bez dodatnih potreba kucanja koda.

Za naš web portal – sajt preko koga ćemo pristupati našoj bazi, bilo iz razloga pregleda informacija ili iz razloga vršenja radnji nad bazom, koristićemo više *frontend* tehnologija što se više tiče izgleda nego funkcionalnosti. Za sadržaj ili web dokument koristimo već davno ustanovljen standard *HyperText Markup Language* verziju 5, u daljem tekstu – *HTML5*. Radi stilizacije istog dokumenta koristimo verziju 3 poznatog *Cascading Style Sheets*, u daljem tekstu – *CSS3*. *HTML* i *CSS* su dve osnovne tehnologije za izgradnju bilo kojih Web stranica. Zadnji aspekt *frontend* tehnologije jeste *Javascript* – poznatiji kao *JS*. *Javascript* će nam omogućiti dodavanje dodatne funkcionalnosti nad našim elementima sajta.

Pored *frontend* dela, za naš web portal je potrebna tehnologija koja je sa serverske strane i može vršiti komunikaciju sa našom bazom podataka, kao i funkciju sigurnosne autentifikacije korisnika. Za taj deo biramo i dan danas najprostranjeniju *backend* tehnologiju koja se koristi prilikom izrade web sajtova – *Hypertext Preprocessor*, u daljem tekstu *PHP*.

Za razliku od čistog *HTML* dokumenata, za *PHP* je potreban pokrenut web server. Dodatno, i našoj bazi podataka je potreban *MySQL* server. Kako bi rešili taj problem, koristićemo *XAMPP* – softver za pokretanje servera i hosta na lokalnoj *Windows* mašini. Tačnije – u našem primeru koristimo ugrađen modul *Apache* web servera, kao i ugrađen modul *MySQL* servera.

Vredno je napomenuti da će sve tehnologije tipa programskog jezika biti rađene preko *Microsoft Visual Code* – softver sa širokom podrškom programskih i ostalih jezika i *runtime* modula.

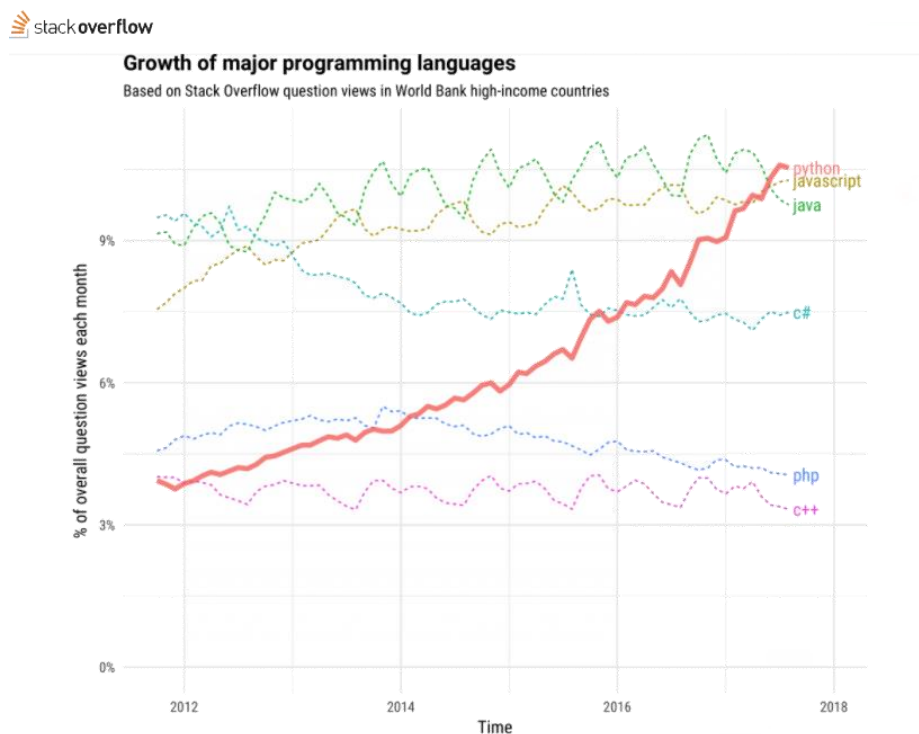
1.2.1 Python

1.2.1.1 Šta je Python?

Python je programski jezik visokog nivoa opšte namene koji podržava imperativni, objektno-orijentisan i funkcionalni stil programiranja. Prva verzija je bila puštena dvadesetog februara 1991. godine. Programski jezik je razvio “*Python Software Foundation*” od dizajna Guido Van Rossuma. Guido je krenuo sa radom na Python-u još osamdesetih godina dvadesetog veka – gde je Python zamišljen kao naslednik programskog jezika “ABC”.

1.2.1.2 Popularnost

Python beleži stabilan rast u popularnosti programskih jezika od svog prvog objavljivanja, gde je čak i uspeo od 2017te biti konkurencija najkorišćenijih programskih jezika kao što su *Java* i *Javascript* – pa čak i da ih prestigne kasnijih godina. Najveći razlog tolike popularnosti Python programskog jezika jeste ogroman spektar njegove primene, masivni broj biblioteka i lakoća kucanja koda što omogućava korisnicima koji nisu najaktivniji u polju programiranja da ga koriste i primenjuju u svojim svakodnevnim životima.



Slika 3 – Rast pitanja po programskom jeziku na StackOverflow platformi kroz više godina

1.2.1.3 Primene

Primene Python programskog jezika su brojne. I dok većina danas smatra Python najboljim jezikom za obradu podataka, njegove primene se ne završavaju tu, imaju tendenciju da se menjaju i da se, iz godine u godinu, pronađu nove koristi za ovim programskim jezikom. Neke od primena su AI i mašinsko učenje, analiza i vizuelizacija podataka, programiranje aplikacija, izgradnja web stranica putem *Python Frameworka* kao *Django* i *Flask*, izrada igrice, korišćenje u finansijske svrhe, automatizacija i skripting, pa čak i izgradnja totalno novih programskih jezika zahvaljujući svojoj elegantnoj sintaksi.

1.2.1.4 Naša primena

Naša primena Python-a, specifično Python verzije 3.9, biće u domenu automatizacije i obrade podataka. Koristićemo veliki broj dostupnih modula radi realizacije ovog cilja i dodatno ćemo se pozabaviti kreiranjem naših modula kako bi softver mogao biti unapređen drugim, kasnije ubačenim modulima radije nego biti menjan u srži svoga koda za nešto trivijalno kao što je, primera radi, dodavanje podrške za još jedan sajt prodaje. Gledamo i skalabilnost samog softvera – da pokretanje procesa za obradu jednog linka 3D modela bude isti i za deset – i za hiljadu.

1.2.1.5 Iskorišćeni moduli

Kao i prethodno napomenuto, Python ima eksponencijalan broj dostupnih biblioteka bilo da su integrisani u samoj instalaciji ili kreirani od strane korisnika. Na sledećoj strani je potpuna lista svih modula, kao i njihov kratak opis, koje ćemo implementirati u naš softver – ne računajući module koje ćemo mi napraviti tokom izrade. Dok su neke od biblioteka importovane za relativno manje funkcije u celoj priči, neki moduli svoje funkcije izvršavaju tokom celog procesa izrade posla. Svakako, svi moduli su kritični za uspešan rad softvera kako je prvobitno zamišljen.

Primenjeni moduli su:

- **Selenium** – uzgred i najveći deo našeg programa. Uglavnom korišćen u Java programskom jeziku kao softver za automatizaciju QA ima i svoju biblioteku za Python. Omogućava korišćenje web pretraživača integrisan u Python programu putem korišćenja njihovih webdrivera pušten od strane njihovih developera za korišćenje pretežno u dev svrhe. Preko ovog modula možemo da vršimo funkcije kao simulacija klika nad i unutar prozora pretraživača zadate u Python-u, kao i parsiranje i manipulisanje koda stranica.

- **Autoit** – koristimo ga kao pripomoć akcijama Selenium biblioteke. Specifično, kontrola pop-up prozora stranica koja traže input korisnika.
- **OS** – koristi se za funkcije nad operativnim sistemom kao što su kopiranje i preimenovanje fajlova i foldera. U našoj situaciji, koristimo ga za raspoređivanje preuzetih fajlova po folderima.
- **Shutil** – koristimo ga kao dodatak OS modulu – primarno radi brisanja foldera koji nisu prazni.
- **Pathlib** – isto kao pripomoć OS biblioteci, koristimo ga za lociranje fajlova i foldera.
- **Docx** – modul za *Microsoft Word* dokumente. U našoj situaciji, koristimo ga za kreiranje, iščitavanje i formatiranje teksta *Word* dokumenta primarno za generisanje opisa oglasa.
- **Time** – modul za import vremenskih funkcija u Pythonu. Koristimo ga za dodavanje tajmera tokom izvršavanja funkcija.
- **Zipfile** – modul za vršenje operacija nad kompresovanim fajlovima. U našem slučaju, koristimo ga za ekstrakciju datoteke koji sadrži slike i fajlove za 3D štampu.
- **Re** – jednostavna biblioteka za REGEX upite.
- **GoogleTrans** – biblioteka ili *API* za prevođenje jezika korišćenjem Google Translate funkcije unutar Pythona. U našem slučaju, prevodimo ime predmeta preuzetog sa stranice 3D modela sa engleskog na srpski, zatim prevedeno ime koristimo kao ime oglasa.
- **CyrTranslit** – koristimo ga kao dodatak GoogleTrans. Sa obzirom da prethodni modul ne podržava srpsku latinicu, ovaj jednostavni modul konvertuje srpsku ćirilicu u srpsku latinicu.
- **MySQL Connector** – biblioteka za vršenje operacija nad bazama podataka.
- **Colorama** – jedini modul koji nije kritičan za operaciju celog softvera, koristi se za dodavanje boja teksta i pozadine u interfejsu komandne linije.
- **Sys** – U našem slučaju, koristimo samo jednu funkciju iz ove biblioteke, a to je za gašenje celokupnog softvera.

1.2.2 HTML

1.2.2.1 Šta je HTML?

HTML - ili *Hypertext Markup Language* je standardizovan sistem za tagovanje tekstualnih dokumenata da bi se postigao efekat fonta, boje, grafike ili *hyperlink*-a na stranicama *World Wide Web*-a. (Oxford, n.d.) Prvi put pušten u industriju dvedeset i treće godine dvadesetog veka, HTML je zauzeo svoju poziciju kao standard za pisanje web dokumenata. Za HTML format je zaslužan “WHATWG”.

1.2.2.2 Primena

HTML opisuje strukturu stranice svojim HTML elementima (ili *tagovima*). Web pretraživači učitavaju dokumenta HTML formata sa servera i pretvaraju ih u multimedijske web stranice. Identifikuju se preko svoje ekstenzije fajla *.html* ili manje poznatom *.htm*. Međutim, HTML je prisutan i u *.php* fajlovima, a može da se nađe i u mnogobrojnim drugim tipovima fajla, kao i da se koristi u više aplikacija kao definisanje struktura različitih ekrana.

1.2.2.3 Elementi

HTML dokument se uglavnom sadrži iz par obaveznih elemenata koji definišu strukturu samog dokumenta, kao i mnogobrojnih opcionih tagova koje dodatno specifičnije definišu i dodatno razlikuju tekst u sebi.

Obavezni elementi koji definišu samu osnovu HTML fajla su:

- `<!DOCTYPE>` - Definiše tip dokumenta
- `<html> </html>` - Definiše HTML dokumenat.
- `<head> </head>` - U sebi sadrži metadata – informacije za dokumenat
- `<body> </body>` - Definiše ‘telo’ dokumenta – prostor gde se ispisuje tekst kao i ostali elementi.

Ovi obavezni elementi dodatno spadaju u grupu osnovnih HTML tagova koji sadrže još par HTML elemenata. Pored grupe osnovnih, ima još par kategorija HTML elemenata, čija celokupna lista se nalazi na sledećoj stranici.

Podela HTML elemenata po kategorijama:

- Osnovni HTML elementi – U ovoj kategoriji, pored obaveznih, spadaju i tagovi za komentar, paragraf, headere i sl.
- Elementi za formatiranje – Ovde spadaju, tagovi za formatiranje teksta kao bold, citat, deo koda itd.
- Forme i inputi – Sadrži tagove za forme, input teksta, dugmad i slično.
- Frame-ovi – Elementi za embed videa ili aplikacija u HTML stranicu.
- Slike – Tagovi za dodavanje različitih tipova slika, kanvasa i vektorskih crteža.
- Audio i Video – Sadrži tagove koje dopuštaju puštanje i kontrolu multimedije.
- Linkovi – Elementi za *hyperlink* i navigaciju.
- Liste – Tagovi za definisanje i stilizovanje listi.
- Tabele – Grupa elemenata za definisanje i manipulisanja tabela.
- Stilovi i semantike – Sadrži elemente kao što su header, footer, sekcija i sl.
- Meta informacije – Elementi za definisanje metapodataka u HTML dokumentu.
- Programiranje – Za definisanje i izvršenje programskog koda sa klijentske strane.

Većina HTML dokumenata sadrži kombinaciju elemenata iz mnoštvo prethodno-navedenih grupa i naša specifična situacija se ne razlikuje mnogo od te tvrdnje.

1.2.2.4 Naša primena HTML elemenata

Našem sistemu su pretežno potrebni elementi iz grupacija formi i inputa, tabela, linkova, kao i podrazumevna grupacija osnovnih elemenata. Inputi su nam potrebni pri logovanju, kao i vršenju naredbi nad bazom podataka, dok su elementi tabele potrebni pri generisanju redova same baze podataka na našem web portalu.

1.2.3 CSS

1.2.3.1 Šta je CSS?

CSS, ili *Cascading Style Sheets* je specifičan *Style Sheet* jezik koji se koristi o opisu izgleda dokumenata koji je pisan u HTML formatu. CSS, zajedno za HTML-om i Javascript-om, spada u osnovne tehnologije za kreiranje i izradu web stranica. Nastao je devedeset i šeste godine dvadesetog veka od strane “*World Wide Web Consortium*”, cele tri godine nakon nastanka HTML formata.

1.2.3.2 Primena

Cascading Style Sheets se koristi kao dodatak HTML dokumentu. Kako bi se povezali, u HTML dokumentu se treba dodati meta tag u head delu koji navodi na putanju CSS dokumenta. CSS dokumenat se prepoznaje po svojom specifičnom ekstenzijom `.css`. Alternativa jeste korišćenje samog taga za stil unutar HTML dokumenta i unutar njega koristiti CSS format pisanja, ali se vrlo obeshrabruje takav način rada zbog svoje ne-efikasnosti. Potrebno je dodatno izmeniti HTML dokument sa CSS selektorima na pojedinom elementu, po potrebi. Moguće je i koristiti same HTML tagove kao selektore unutar CSS-a, ali zbog dodatnog izdvajanja, trebaju se implementirati klase, kao i identifikatori.

CSS se dodatno koristi kako bi se dodatno web stranice napravile “*responsive*”, ili da odgovaraju svakom ekranu bez gužvanja, gubljenja kvaliteta ili odstupanja od odobrenog dizajna. U fajlu je moguće definisati pri kojoj veličini ekrana koja linija koda da važi.

Preko CSS fajla je moguće dodavati i animacije za pojedine ili više elemenata, dodavati posebne izgled elementima pri posebnim fazama, primera radi, faza klika ili faza fokusa elementa. Moguće je i primenjivati posebne linije koda za posebne web pretraživače radi bolje pokrivenosti funkcija ili iskorišćavanja prednosti istih.

Cascading Style Sheets zaista sadrži prevelik broj interesantnih stvari koje bi mogle da se pokriju, ali radi simplističnosti i manjak potrebe za njima, držimo se osnova u ovom dokumentu.

1.2.3.3 Klase

Klase su atribut HTML elementa koje se koriste radi selekcije u CSS jeziku pri unikatnom stilizovanju i formatiranju. Vredi napomeniti da se klase uglavnom koriste kada treba većoj grupi elemenata primeniti iste opcije stilizovanja kako bi se izbeglo ponavljanje istog koda stilizovanja za više elemenata.

Klase se u HTML dokumentu definišu na sledeći način:

- `<tag class=" ime "> </tag>`

Klase se u CSS dokumentu pozivaju na sledeći način:

- `.ime {
 // kod za stilizovanje
}`

1.2.3.4 ID Selektor

Osim HTML tagova i atribut klase, moguće je dodatno obeležavati tagove ID selektorima u CSS dokumentu ili unutar style tag-a. ID selektor se uglavnom koristi na po jednom elementu u celom HTML dokumentu i više ima primenu za Javascript zbog unikatnosti, ali svakako doprinosi i obeležavanju elementa u CSS dokumentu radi unikatnog stilizovanja.

ID selektor se u HTML dokumentu definišu na sledeći način:

- `<tag id=" ime "> </tag>`

Klase se u CSS dokumentu pozivaju na sledeći način:

- `#ime {
 // kod za stilizovanje
}`

1.2.4 JavaScript

1.2.4.1 Šta je JavaScript

JavaScript, ili *JS* je objektno-orijentisan jezik za *skripting* koji je ujedno i deo osnovnih tehnologija *World Wide Web-a*, pored prethodno-navedenih tehnologija HTML i CSS. Dizajniran je od strane Brendan Eich iz firme “*Netscape*” i prvi put pušten devedeset i pete godine dvadesetog veka. Iako većina smatra *Javascript* programskim jezikom, on se razlikuje od njih iz razloga što, dok se programski jezici moraju prvo kompajlirati, kod JS-a je ceo u tekstu.

Vredno je napomenuti da JavaScript i Java – objektno-orijentisani *programski* jezik koji služi za izgradnju aplikacija nisu u srodstvu jedni sa drugim.

1.2.4.2 Primena

Javascript se koristi u izradi web stranica. Smatra se kao deo *frontend* tehnologija uprkos svojim očiglednim *backend* karakteristikama. Koristi se za kontrolu raznolikih multimedija koje stranice sadrže, animiranje slika i ostalih HTML elemenata, pridodavanje funkcija i *listener-a* elementima kao što su dugmad, forme i ostalih funkcionalnih elemenata HTML dokumenta.

1.2.4.3 Dodaci

Javascript ima svoje dodatke koji mogu da doprinesu korisniku na više načina. Par njih su olakšanje kucanja sintakse, bilo u smislu kraćeg ili jasnijeg koda, pojednostavljene ili predefinisane funkcije za određene radnje, čitanje različitih standarda fajlova, komuniciranja preko raznih aplikacionih programskih interfejsa – ili *API*. Najpopularniji među njima je *JQuery*.

JQuery je brza i relativno mala biblioteka *JavaScript-a* bogata raznih predefinisanih funkcija koje se većinski koriste za manipulisanje i “*prekopavanje*” HTML dokumenta, animacija, rukovanje *event-ima* i slično. Znatno je istaći da je *JQuery* promenio način na koji ogroman broj ljudi danas kuca *JavaScript*.

1.2.4.4 Frameworks

Framework-ovi u JavaScript-u je kolekcija biblioteka koda JavaScript skripting jezika koje služe kao pomoć korisnicima koji kreiraju web stranice ili sajtove tako što doprinose template kodove i predefinisane funkcije za problematike koje su čestih pojava tokom izrade.

Svaki od JavaScript framework-a dolazi sa svojim pravilima, kao i sa korisničkim uputstvom za korišćenje. Neki od njih omogućavaju uspešan rad sa malo, pa čak i bez znanja logike i sintakse ovakvih ,tipova objektno-orijentisanih jezika.

Neki od najpopularnijih framework-ova JavaScript jezika su:

- **AngularJS** – Jedan od najjačih i najefikasnijih framework-a JavaScript jezika. On je “*open source*” i koristi se za izrađivanje funkcionalnih aplikacija na jednoj strani (eng. *Single Page Application* – *SPA*).
- **React** – Koristi se za izgradnju dinamičkih korisničkih interfejsa na web stranicama namenjene za veliki broj posetioaca i integracija sa bilo kojom aplikacijom je relativno laka. Napravljen od strane *Facebook*-a, danas poznatija kao *Meta*.
- **Node.js** – Framework koji se izvršava sa serverske strane, isto je jedan “*open source*” framework-ova, ima visoke performanse, lak je za učenje i ima mogućnost asinhronne I/O komunikacije.

1.2.5 PHP

1.2.5.1 Šta je PHP?

PHP je *objektno-orijentisan* skripting jezik napravljen za svrhe izgradnje web stranica i sajtova. Kreiran je devedeset i pete godine dvadesetog veka, od strane Rasmus Lerdorf-a programera sa teritorije Danske. U početku, skraćenica *PHP* je značila “*Personal Home Page*”, ali kasnije je poprimila naziv, koji glasi “*PHP: Hypertext Preprocessor*” i taj naziv i dan danas stoji kao važeći.

1.2.5.2 Primena

PHP je skripting jezik integrisan u sam HTML dokument koji se izvršava sa strane servera, što se razlikuje od JavaScript skripting jezika koji se izvršava u web pretraživaču sa klijentske strane. Za uspešno pokretanje PHP koda unutar HTML stranice, potrebno je, prvobitno, promeniti samu ekstenziju fajla iz *.html* ili *.htm* u *.php*. Za razliku od HTML fajlova, PHP mora biti pokrenut preko samog web servera koji na sebi ima instaliran PHP modul.

Koristi se za *backend* deo izrade sajta, i uglavnom vrši rolu kao alat za kreiranje dinamičkog sadržaja na web stranicama, dodavanja funkcije logovanja na sajt putem sesija, kao i komuniciranje sa bazom podataka.

Mnoštvo danas popularnih sajtova i web aplikacija su izrađene u PHP-u. Par najpopularnijih koje vredi napomenuti su *Facebook*, *Wikipedia* i *Tumblr*. Pored toga, najpopularniji alat za jednostavno i olakšano kreiranje *frontend* dela web sajtova, *Wordpress*, koristi PHP. PHP je i dalje najpopularniji skripting jezik za izradu sajtova koji se izvršava sa serverske strane gde i danas, preko 77 odsto sajtova na svetu koji imaju *backend* jezik, koriste PHP. (W3Techs, 2022)

1.2.5.3 Frameworks

Kao i Javascript, PHP isto ima svoje framework-e dostupne za korišćenje radi olakšane izrade problematika. PHP framework-ovi dodeljuju korisniku biblioteke koda za najkorišćenije funkcije i time rešavaju problem viška koda koji bi morao da se otkuca bez njih radi sopstvenog kreiranja funkcije za isto.

Ispod ovog teksta su neki od najpopularnijih framework-ova PHP skripting jezika:

- **CodeIgniter** – Jedan od najstarijih PHP framework-a, *CodeIgniter* je namenski za pripomoć tokom izrada web aplikacija brzo i sa minimalnom potrebnom konfiguracijom. Poznat je po svojoj brzini, fleksibilnosti i malog prostora koji zauzima - svega 1.2 megabajta. Izašao je februara 2006. godine.
- **Laravel** – Prvi put pušten u javnost 2011. godine, *Laravel* je danas možda i najkorišćeniji PHP framework. Kreirao ga je *Taylor Otwell* iz razloga što je želeo postojanje PHP framework-a sa elementima koje *CodeIgniter* nije posedovao. Danas je *Laravel* poznat kao “*PHP Framework za veb zanatlije*”.
- **Symfony** – Ujedno framework i kolekcija PHP komponenata za kreiranje web sajtova. Lansiran je oktobra 2005. godine. Jedna od stvari koje ističu ovaj framework od ostalih popularnih jeste broj baza podataka koje poričava.

1.2.6 MySQL

1.2.6.1 Šta je SQL

SQL ili *Structured Query Language* je jezik specifičnog domena koji je dizajniran za upravljanje podataka u sistemu za menadžment relacionih baze podataka – poznatiji kao *RDBMS*. *SQL* je dizajniran od strane *Donald Chamberlina*-a i *Razmond Boyce*-a i prvi put javno izašao sedamdeset i četvrte godine dvadesetog veka. *SQL* spada u struktuisani jezik upita.

SQL se koristi za kreiranje baza podataka, tabela, dodavanja i brisanja podataka u iste, kreiranje korisnika za pristup i kontrola istih, vršenje upita nad bazom i mnoštvo drugih funkcija.

1.2.6.2 Šta je MySQL

MySQL je *open source* sistem za upravljanje relacionih baza podataka – *RDBMS*. Nastao je 23. maja devedeset i pete godine dvadesetog veka čiji je autor softverska firma i Švedske – *MySQL AB*. Sistem je razvio korporacija *Oracle*.

1.2.6.3 Šta su relacione baze podataka

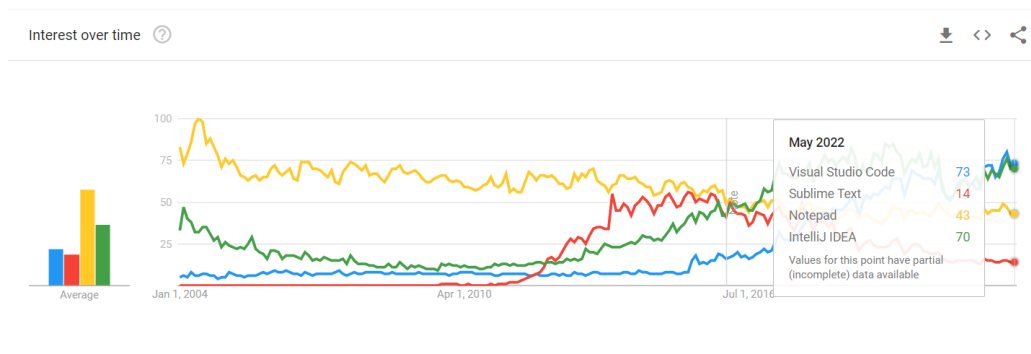
Relacione baze podataka su digitalne baze podataka bazirane na relacionom modelu podataka. Termin je prvo osmišljen od strane *E.F. Codd* iz *IBM-a* sedamdesetih godina dvadesetog veka.

Model relacione baze podataka organizuje podatke u jednu ili više tabela sa redovima i kolonama sa jedinstvenim ključem koji identifikuje svaki red posebno. Svaka tabela predstavlja jedan entitet dok svaki red predstavlja jednu instancu tog entiteta. Uz to, svaka kolona predstavlja atribut instance entiteta.

1.3 Alati

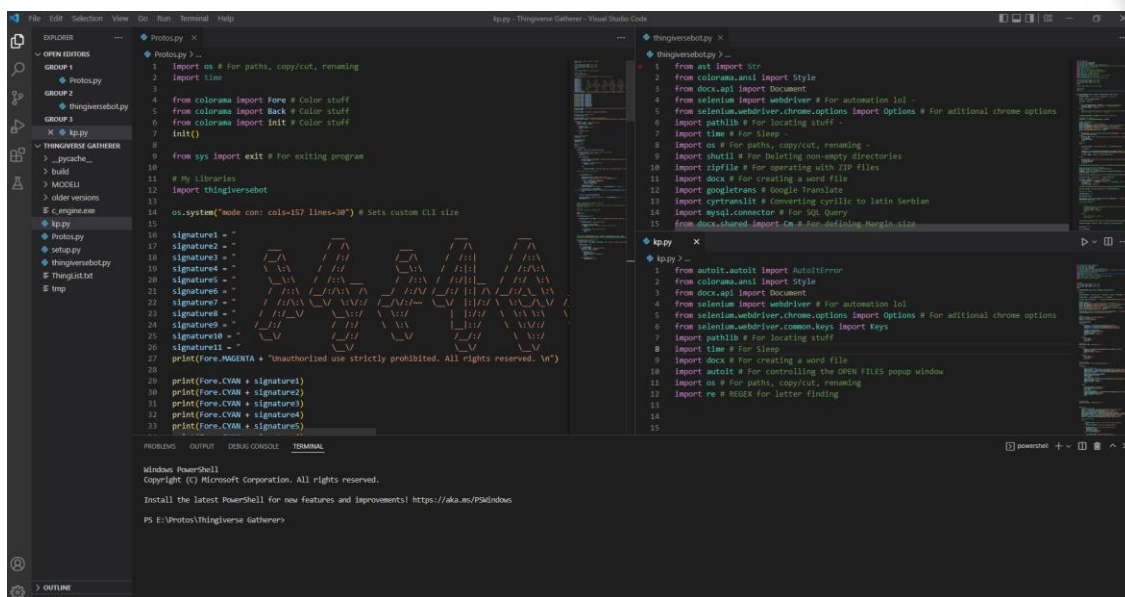
1.3.1 Visual Studio Code

Visual Studio Code – ili skraćeno VS Code je editor koda kreiran od strane Microsoft i dostupan je na platformama Windows, Linux i MacOS operativnih sistema. Nudi enormnu podršku različitih programskih jezika i pametno popunjavanje istih, kao i funkcije pokretanja koda, debugging mod, module – drugi naziv “ekstenzije“, za mnoštvo različitih situacija i dosta drugih opcija. Iz ovih razloga među ostalima, VS Code beleži stabilan rast popularnosti, gde, među svojom konkurencijom, danas zauzima prvo mesto po popularnosti.



Slika 4 – Broj pretraga po softveru za programiranje od 2004 god. do 2022 god. – Google Trends

Grafički korisnički interfejs je i više nego intuitivan, čiji je dizajn dosta modernizovan, nudi dosta podešavanja i različitih pregleda bilo za terminale ili otvaranje više prozora za različite fajlove koda odjednom.



Slika 5 – Primer izgleda interfejsa Visual Studio Code

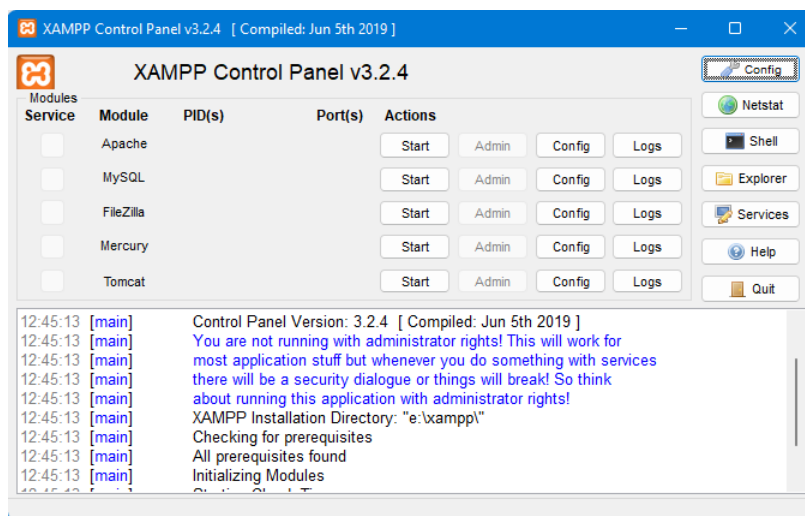
1.3.2 MySQL Workbench

MySQL Workbench služi kao softver vizuelnu kontrolu nad bazama. U sebi objedinjuje SQL naredbe, administraciju, dizajn baza podataka, njihovih održavanja kao i samo kreiranje istih. Izrađen je od strane korporacije *Oracle i Sun Microsystems*.

Prednost ovog softvera jeste što je besplatan za korišćenje i omogućuje dosta lakši rad nad baza podataka iz razloga što se sve odvija kroz grafički korisnički interfejs – naspram klasičnog koda za sve potrebe. Uspešno funkcionisanje ovog softvera se izvodi automatskim generisanjem SQL koda na osnovu prethodno izabranih opcija preko interfejsa.

1.3.3 XAMPP

XAMPP je softver koji je *open source* i besplatan za korišćenje čija instalacija je dostupna za *Windows, Linux i MacOS* operativne sisteme preko web sajta udruženja zaslužno za razvoj ovog programa "*Apache Friends*". Služi za pokretanje web servera sa lokalnog računara i po default-u sadrži module za *Apache* web server i *MySQL* server, dva modula koje ćemo koristiti u našem slučaju tokom izrade i pokretanja našeg sistema. Pored njih, sadrži i module za *FileZilla, Mercury i Tomcat*. Iz razloga što koristimo tehnologije kao što su *PHP* i baze podataka, radi uspešne upotrebe istih, potrebno ih je podići preko modula koji uključuje XAMPP.



Slika 6 – Primer izgleda interfejsa XAMPP panela za kontrolu

Vredno je napomenuti da *XAMPP* potiče od *LAMP* – *Linux, Apache, MySQL, PHP/Perl/Python* stek softvera koji se postavlja i koristi na *Linux* serverima radi postizanja iste funkcionalnosti.

2. Izrada

Kako smo prikupili sve potrebne alate za izradu zajedno sa izrađenom idejom preko iscrtanih dijagrama, vreme je da se započne sama realizacija celokupnog sistema. Prvi na redu jeste sam softver za automatizaciju prikupljanja podataka modela za 3D štampu, njihovih informacija i izrada oglasa na izabranim medijima prodaje.

2.1 Softver za automatizaciju

U okviru obima koji je zamišljen za izradu samog softvera, trenutna ideja jeste da se naš softver sastoji od 3 modula.

1. **Glavni modul** – Ujedno i modul od koga kreće celokupna procedura automatizacije, ovaj modul služi kao pokretač ostalih modula. Zamišljeno je da se kasnije, pri dodavanju dodatnih modula radi podrške više sajtova sa koga se preuzimaju modeli 3D štampe ili dodavanje modula koji podržavaju ostale medije prodaje, mogu birati ručno, pa čak i automatski implementacijom algoritma za prepoznavanje ubačenih linkova sa kojim modulom će trenutno softver da radi. Time rečeno, glavni modul treba da se sastoji od osnovnih elemenata biranja moda funkcionalnosti softvera pri pokretanju. Pod time smatramo opcije kao što su polu ili potpuna automatizacija – gde potpuna automatizacija isključuje računanje cene radi postizanja totalno samostalan rad softvera.
2. **Modul za sajt koji sadrži 3D modele** – Ovaj modul u sebi ima algoritam pretage stranica i pronalaženja adekvatnih elemenata koji su nam potrebni, zatim kasnije ih preuzima, zapisuje i preraspoređuje za kasnije upotrebu. U našem primeru, izrađujemo jedan modul za sajt 3D modela – www.thingiverse.com
3. **Modul za sajt prodaje** – Služi za objavu oglasa na adekvatnom sajtu za prodaju. U našem slučaju, izabrali smo domaći sajt – www.kupujemprodajem.com . Ovaj modul se sastoji iz algoritma za korišćenje nekih podataka i fajlova prethodno preuzetih pomoću prethodno-navedenog modula – *modul za sajt koji sadrži 3D modele* i prema njima samostalno generiše oglas na adekvatnom sajtu.

2.1.1 Izrada glavnog modula

Funkcija glavnog modula je relativno kratka i jasna – da služi kao pokretač ostalih modula sa specifičnom misijom. Prvobitno pišemo kod za prethodno navedenu glavnu funkciju. To je – da li želimo da program bude potpuno automatizovan tokom obrade liste linkova (*gde žrtvujemo algoritam za računanje cene iz razloga što je potreban ljudski input i korišćenje dodatnog programa*).

Da bi postigli to, jednostavno koristimo promenljivu koja će kasnije biti učitana. Dodatno pravimo i *failsafe* – da program traži upisan odgovor “Da” ili “Ne” dok god nije adekvatno unet. Uz to smo ubacili i deo koda za ignorisanje velikih ili malih slova radi dodatne optimizacije i lakoće korišćenja.

```
#Should price calculation be used
priceCalculation = True

print(Fore.WHITE + "Da li da racunam cenu proizvoda?")
print(print(Fore.YELLOW + "(Ova opcija onemogucava potpunu automatizaciju i eksponencijalno usporava proces.)"))

while True:
    print(Fore.WHITE + "Unesite DA ili NE")
    confirm = input("").upper()
    if confirm == "NE":
        priceCalculation = False
        break
    elif confirm == "DA":
        break
```

Vredi napomenuti da ćemo kasnije, tokom pisanja našeg softvera, dosta koristiti isti kod za binarnu potvrdu, i sa time na umu, nećemo i ponovno dodatno opisati niti obraćati pažnju.

Kada smo izradili inicijalni upit od strane korisnika, sledeći problem jeste – ponovno izvršenje celog programa dok korisnik ne prekiene proces ili dok lista koja će biti input ne ostane bez iščitanih linkova. Ovo rešavamo tako što ćemo odraditi par koraka. Prvi korak jeste postaviti ceo naredni kod pod *while* petljom – petlja koja se ponavlja dok se zadati uslov ne ispuni. Pod “*narednim kodom*” smatramo ceo preostali program, uključujući i pokretanje funkcija iz ostalih modula.

Kada je program stavljen pod *while* petljom (gde koristimo *boolean* vrednost “*programLoop*”) sledeći kod se odnosi na situaciju gde je korisnik odabrao nepotpunu automatizaciju – odnosno pokretanje sa računanjem cene. Ovaj *snippet* koda zatražuje od korisnika da ručno unese link koji vodi do specifičnog 3D modela na sajtu *Thingiverse*. U sebi sadrži proveru da li je link validan.

```
while programLoop:

    urlCheck = "https://www.thingiverse.com/thing:"
    if priceCalculation == True: # Ask for link if it is manual
        # Loop while link doesn't match #
        isNotLink = True
        while isNotLink:

            print(Back.WHITE + Fore.BLACK + "Postavite link za Thingiverse stranicu na kojoj je predmet: "
+ Back.RESET + Fore.RESET)
            urlLink = input("")
            if urlCheck in urlLink: #break loop
                isNotLink = False
            else:
                print(Fore.YELLOW + "Nije dobro unet link. Link treba da sadrzi: " + urlCheck)
```

Ispod ovog koda je i deo koji se izvršava u slučaju da je korisnik izabrao potupunu automatizaciju, gde se odvija iščitavanje fajla u direktorijumu gde je i sam glavni modul. Ovaj deo je alternativa za samostalno unošenje linka u slučaju nepotpune automatizacije.

```
else: # Load up the file with link
    if automatedCount == 0:
        print(Fore.YELLOW + "Ucitavam listu linkova...")
        thingListFile = open("ThingList.txt", "r", encoding="utf-8")

        for line in thingListFile.readlines():
            line = line.replace("\n","")
            thingList.append(line)

        thingListFile.close()
        print(Fore.GREEN + "Lista sadrzi " + str(len(thingList)) + " linkova.")

    elif len(thingList) == automatedCount:
        print(Fore.GREEN + "Uspesno sam obradio svih " + str(len(thingList) + 1) + "linkova.")
        break
    urlLink = thingList[automatedCount]
    print(Fore.YELLOW + "Radim na linku broj " + str(automatedCount + 1) + "...")
    automatedCount += 1
```

U ovom bloku koda se nalazi i provera pri svakom ponovnom izvršavanju – da li je lista ostala bez linkova. U slučaju da jeste, program izlazi iz *while* petlje. Razlika od situacije gde je izabrana polu-automatizacija je u tome što, u situaciji računanja cene, svako ponovno izvršavanje pita korisnika da li želi da obradi sledeći link.

Sledeći blok koda se izvršava samo u slučaju da je korisnik izabrao polu-automatizaciju. To je upit za korisnika – „*Da li je predmet luksuznog tipa?*”. Ovo se odnosi na koji algoritam će softver koristiti za računanje cene. Ova opcija jeste ubačena iz razloga što 3D modeli mogu da se odnose na stvari za svakodnevno korišćenje – kao kukice, čepovi, držači i ostale plastične stvari, ali i na stvari koje su unikat, retke i u trendu na kojima je moguće ostvariti dodatan profit.

```
price = 0
isVip = False

if priceCalculation == True:
    print(Back.WHITE + Fore.BLACK + "Da li je predmet luksuznog tipa? (Sezonalno, u trendu, unikat  
itd.)" + Back.RESET + Fore.RESET)

    while True:
        print(Fore.WHITE + "Unesite DA ili NE")
        isVipInput = input().upper()
        if isVipInput == "NE": # Standard calculate - 50% profits
            break
        elif isVipInput == "DA": # VIP calculate
            isVip = True
            break
```

Nakon ovog inputa (*ili bez ovog koraka u slučaju potpune automatizacije*) pokrećemo kod za iniciranje funkcije sa potrebnim parametrima iz našeg modula za pretragu thingiverse sajta - *thingiverseBot*.

```
thingiversebot.thingiverseBot(urlLink, urlCheck, priceCalculation, price, isVip)
```

Ovo je ujedno i poslednji blok koda koji se izvršava u našem glavnom modulu ako je u pitanju izabrana opcija potpune automatizacije. Ako ne – postoji još jedan blok koda koji korisnika dodatno pita da li želi obraditi još jedan model – u kojem se slučaju *while* petlja ponovno izvršava i proces kreće ispočetka.

2.1.2 Izrada modula za sajt koji sadrži 3D modele

Kada se završi inicijalni proces našeg glavnog modula, iz glavnog modula sa adekvatnim parametrima se pokreće funkcija iz modula za sajt *thingiverse* radi prikupljanja potrebnih informacija zadatog linka. Parametri koji se preuzimaju iz glavnog modula jesu link, promenjiva za proveru linka, *boolean* vrednost da li je uključena potpuna automatizacija ili ne, kao i cena i *boolean* vrednost za luksuz.

Pri pokretanju, modul radi *failsafe* proveru za već postojeće foldere koje bi hteo da napravi radi skladištenja potrebnih fajlova, slika i ostalih informacija. Tačnije, radi proveru tako što od linka oduzima sve sem identifikacionog broja 3D modela iz linka i ako postoji folder, briše ga. Do ove situacije dolazi ako je prethodno model već bio procesuiran. Nakon toga, kreira folder koji zauzima privremeno ime identifikacionog broja. Uz to, preko varijabla skladišti ID modela i putanju do foldera.

```
# Creating Folder #
print(Fore.GREEN + "===== ZAPOCINJEM =====")

thingID = urlLink.replace(urlCheck, "")
mainFolder = "MODELI\\"

try:
    shutil.rmtree(mainFolder + thingID) #Try to delete if already exists
    print(Fore.YELLOW + "Folder vec postoji...") #####ALERT
    print(Fore.GREEN + "Uklonjen!")
except:
    pass

os.makedirs(mainFolder + thingID) #Create folder per thing title
# Defining Folder Path #
thingFolder = os.path.abspath(mainFolder + thingID)
```

Nakon prvog koraka, pomoću *Selenium* modula, pali se *Google Chrome* web pretraživač sa određenim brojem dodeljenih opcija – jedna od kojih ga čini da radi u “*nevidljivom*” modu. Pretraživač za argument uzima *URL* unet od strane fajla ili korisnika, dodelje opcije i putanju do *webdriver-a*. Dodatno, kao *failsafe* se koristi provera verzije samog drajvera u slučaju da pukne pokretanje.

Blok opisanog koda se nalazi na sledećoj strani.

```
# Setting Chrome Options #
options = Options()
options.headless = True # Invisible window
options.add_argument('--window-size=1920,1080') # Set size due to bugging out while invisible
options.add_experimental_option("excludeSwitches", ["enable-logging"])
options.add_argument("--log-level=OFF") # Disables log of Chrome
prefs = {"download.default_directory" : thingFolder, "directory_upgrade" : "True",
"extensions_to_open" : ""} #WORKS!!!!!! - For setting different download path.
options.add_experimental_option("prefs",prefs)

try:
    browserChrome = webdriver.Chrome(str(pathlib.Path().absolute()) + '\c_engine.exe',
options=options)
    browserChrome.get(urlLink)
except:
    print(Fore.RED + "Molim vas preuzmite novu verziju Chrome Drivera, preimenujte ga u 'c_engine.exe'
i zamenite trenutni fajl u ovom direktorijumu sa novim. \n\n https://chromedriver.chromium.org/downloads")
    os.remove(thingID)
    exit()
```

Sledeći koraci funkcije za prikupljanje i preuzimanje podataka jesu navigiranje stranica preko *HTML* elemenata, klasa ili ID-a koristeći funkciju klik-a iz *Selenium* biblioteke. Uz to, određene delove teksta se skladište u promenjive, kao na primer originalan naziv 3D modela, koji se, pomoću *GoogleTrans* i *CyrTranslit* biblioteka prevodi na srpsku latinicu da bi se koristio kao ime kasnije na oglasima. Uz to se i određeni *failsafe* kodovi izvršavaju.

```
#Grab data
try:
    print(Fore.YELLOW + "Trazim detalje modela...") #####ALERT
    thingTitle = str(browserChrome.find_element_by_class_name("ThingPage__modelName--
3CMsV").text).replace(' ','').replace(' ','').replace('/','').replace('\\','').replace(':', '') #Find title
    thingDescription = browserChrome.find_element_by_class_name("ThingPage__description--14TtH").text
    print(Fore.GREEN + "Uspesno!") #####CONFIRM
except:
    print(Fore.RED + "Ovaj link ne vodi ka validnom modelu ili je doslo do greske u programu!
Proverite link i pokusajte opet.")
    browserChrome.close()
    os.remove(thingID)
    return -1

##### TRANSLATE #####
#Used for setting title on KupujemProdajem
print(Fore.YELLOW + "Prevodim...") #####ALERT
translator = googletrans.Translator()
thingName = cyrtranslit.to_latin(str(translator.translate(str(thingTitle), src="en", dest="sr").text))
```

Tokom izvršavanja prethodnog bloka koda se gasi prozor *Chrome* pretraživača i pali drugi koji direktno otvara link za preuzimanje *.zip* fajla 3D modela. Ovo je iz razloga što, iako je *Selenium* vrlo jaka alatka za korišćenje, ima svojih limitacija – jedna od kojih je manjak mogućnosti za otvaranja drugog *URL-a* u istom prozoru.

Sledeći blok koda radi proveru uspešno preuzetog *.zip* fajla tako što na određenom intervalu proverava postojanje istog. Zatim ga otvara, broji količinu slika i fajlova za štampu i ekstrahuje ih u naš folder. Broj fajla za štampu se čuva u promenljivoj jer se kasnije koristi u slučaju da nije izabrana potpuna automatizacija pri pokretanju softvera.

```
##### SAVING DATA #####
print(Fore.YELLOW + "Preuzimam.", end="", flush=True) #####ALERT
noFileExists = True
while noFileExists:
    try:
        #Find ZIP file and extract it
        for file in os.listdir(thingFolder):
            if ".zip" in str(file) and ".crdownload" not in str(file):
                zippedFile = thingFolder + '\\' + file

                with zipfile.ZipFile(zippedFile, 'r') as zip_ref:
                    img_datatype = ['.png', '.jpg', '.jpeg', '.bmp', '.gif']
                    stl_count = 0
                    img_count = 0

                    for file in zip_ref.filelist:
                        if ".stl" in str(file).lower():
                            stl_count += 1
                            zip_ref.extract(file, thingFolder)
                        elif any(i in str(file).lower() for i in img_datatype):
                            img_count += 1
                            zip_ref.extract(file, thingFolder)
                        else:
                            continue

                    zip_ref.close()
                    noFileExists = False
                    break
            elif file == os.listdir(thingFolder)[-1]:
                raise Exception()

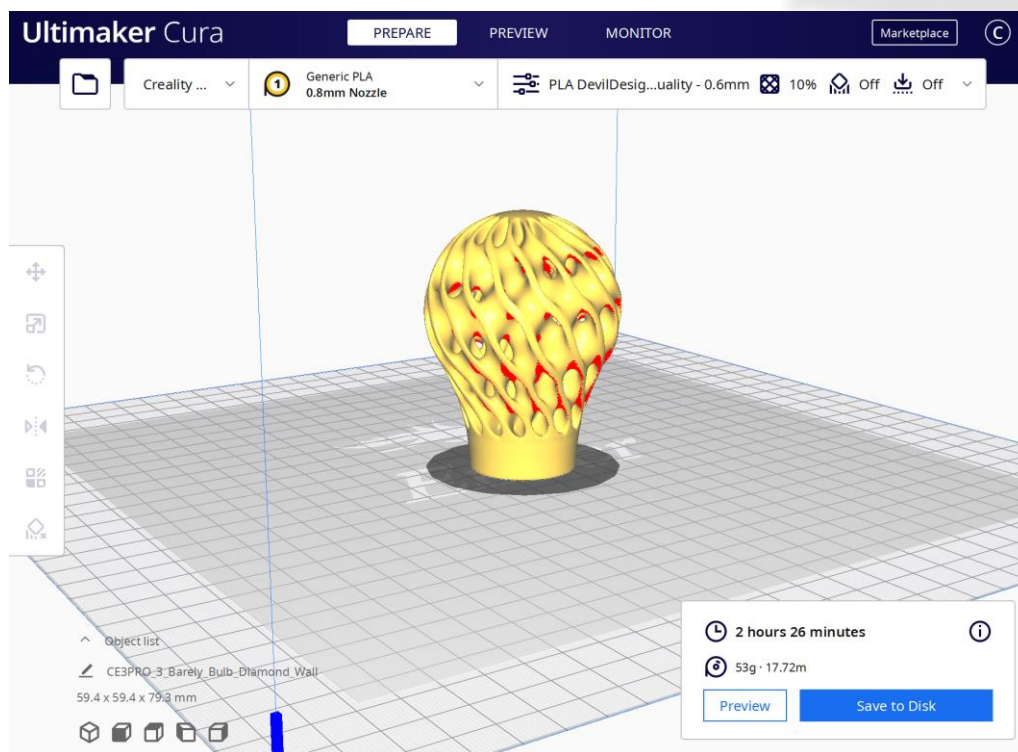
    except:
        print(Fore.YELLOW + ".", end="", flush=True)
        time.sleep(1)
print(Fore.GREEN + "\nUspesno!")
```

Nakon toga, izvršavaju se par bloka koda radi reorganizacije strukture direktorijuma i preimenovanje foldera u originalno ime 3D modela. Posle uspešnog izvršavanja, softver izbacuje poruku gde su sačuvani fajlovi i moguće upozorenje za prethodno pomenut broj fajla za štampu – upozorenje služi da, tokom polu automatizovanog procesa, korisnik primeti prisustvo više fajlova radi izbegavanja grešaka tokom generisanja modela.

```
print(Fore.GREEN + "=====ZAVRSENO=====")
print(" ")
print(Fore.WHITE + "Svi fajlovi sacuvani u direktorijumu: " + thingTitle)
print(" ")

if stl_count > 1:
    print(Back.YELLOW + Fore.BLACK + "!!!!!!!!!!!! OPREZ !!!!!!!!!!" + Back.RESET + Fore.RESET)
    print(Fore.YELLOW + "Imate vise fajlova za model, dobro pogledajte u softveru da li je iz vise delova!")
    print(Fore.YELLOW + "Broj fajlova za model: " + str(stl_count))
```

Sledeći blok koda je samo u situaciji da je odabrana polu automatizacija. Za ovaj deo je potrebno koristiti softver za 3D štampu “*Ultimaker Cura*” i ubaciti sve fajlove za štampu u njega, po mogućstvu složiti ih u obliku kakvom treba da ispadnu radi korektnog generisanja informacija i kliknuti na “*Slice*”.



Slika 7 – Interfejs Ultimaker Cura softvera sa učitanim modelom za štampu

Tokom ovog dela, naš softver traži input parametara koji se dobijaju u *Cura* softveru. Treba ih uneti u adekvatnom formatu. Tražene informacije su dimenzije, težina i vreme izrade koji, kada se unesu, softver generiše cenu uz pomoć parametara, *boolean* promenjive za luksuz i zajedno sa cenom pravi *Word* dokument u folderu gde generiše, formatira i upisuje opis predmeta.

```
print(Fore.YELLOW + "Generisem opis predmeta...")

document = docx.Document()
paragraph = document.add_paragraph()
templateFile = open("tmp", "r", encoding="utf-8")
lineCount = 1 #for distinguishing which lines should be edited
boldLines = [8,19,21,22,23,26,32,33,35,37]
italicLines = [3,4,6,24,30,34,36,38]
paragraph.add_run("Šifra artikla: ").bold = True
paragraph.add_run(str(thingID) + " ").italic = True
for line in templateFile.readlines():
    line.replace("\n", "")

    if "Dimenzije:" in str(line):
        paragraph.add_run(line).bold = True
        paragraph.add_run(str(mmx) + " mm x " + str(mmy) + " mm x " + str(mmz) +
"mm\n").italic = True

    elif "Težina:" in str(line):
        paragraph.add_run(line.replace("\n", "") + "\n").bold = True
        paragraph.add_run(str(g) + " g(+/-5%)\n").italic = True
    elif "Materijal:" in str(line):
        paragraph.add_run(line.replace("\n", "") + "\n").bold = True
        paragraph.add_run("100% PLA Plastika\n").italic = True
    elif "Vreme izrade:" in str(line):
        paragraph.add_run(line.replace("\n", "")).bold = True
        paragraph.add_run("\n" + str(h) + " sati ~\n").italic = True
    elif "Dostupne boje plastike:" in str(line):
        paragraph.add_run(line.replace("\n", "")).bold = True
        paragraph.add_run("\nCrna, Bela (Moguće i sve ostale boje uz
dogovor!)").italic = True

    elif lineCount in boldLines: # For BOLD lines
        paragraph.add_run(line).bold = True
    elif lineCount in italicLines: # For ITALIC lines
        paragraph.add_run(line).italic = True
    else:
        paragraph.add_run(line)
    lineCount += 1
```

Dodatno podešava margine kako bi se uklopio opis i čuva fajl u folderu predmeta.

```
#Setting document margins
sections = document.sections
for section in sections:
    section.top_margin = Cm(0.5)
    section.bottom_margin = Cm(0.5)
    section.left_margin = Cm(0.5)
    section.right_margin = Cm(0.5)
document.save("MODEL\\\" + thingTitle + "\\\" + "0 - OPIS OGLASA.docx")
templateFile.close()
print(Fore.GREEN + "Dokument uspesno generisan u folderu modela!")
kpLink = kp.kupujemProdajem(thingTitle, int(price), str(thingName),
priceCalculation)
```

Nakon pokretanja prethodnog bloka koda se i pokreće zadnji modul – *modul za sajt prodaje*. Vredno je napomenuti da se isti proces generisanja opisa izvršava i tokom potpuno automatizovanog procesa – izuzev generisanja cene i karakteristika predmeta u opisu.

Pored ovog poslednjeg bloka koda se nalazi i kod za upisivanje informacija u bazu podataka koji se izvršava nakon celokupne funkcije iz modula za sajt prodaje.

```
##### FOR DATABASE STUFF #####
print(Fore.YELLOW + "Ubacujem podatke u bazu podataka...")

mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="protos_in3d"
)
mycursor = mydb.cursor()

sql = "INSERT INTO product (product_id, product_kp_name, product_og_name, product_kp_link,
product_og_link, product_price) VALUES (%s, %s, %s, %s, %s, %s)"
val = (int(thingID), str(thingName), str(thingTitle), str(kpLink), str(urllink), int(price))

try:
    mycursor.execute(sql, val)
except:
    print(Fore.RED + "Neuspesna radnja dodavanja proizvoda u bazu podataka!")

mydb.commit()
print(Fore.GREEN + "Uspesno!")
```

2.1.3 Izrada modula za sajt prodaje

Nakon obrađene funkcije za preuzimanje svih potrebnih podataka se izvršava funkcija za postavljanje oglasa na sajtu KupujemProdajem iz modula sajta prodaje.

Izvršavanje koda počinje otvaranjem još jednog *Chrome* web pretraživača, ali ovaj put u vidljivom modu sa posebnim parametrima za dimenzije prozora – bitno iz razloga što neke potrebne funkcije za izvršavanje propadaju ako element na kojima se vrši radnja nije vidljiv – još jedna mana *Selenium* biblioteke.

Prvi korak koji softver radi jeste samo logovanje na sajt zadatim parametrima sa, opet, adekvatnim *failsafe*.

```
print(Fore.YELLOW + "Pokusavam logovanje...")
email = "pavlek@hotmail.rs"
password = "■■■■■■■■■■"
print(thingLocation)
while True:
    try:
        loginEmail = browserChrome.find_element_by_id("email")
        loginPassword = browserChrome.find_element_by_id("password")
        loginButton = browserChrome.find_element_by_id("submitButton")

        loginEmail.send_keys(email)
        loginPassword.send_keys(password)
        loginButton.click()
        print(Fore.GREEN + "Uspesno!")
        break
    except:
        print("Radim...")
        time.sleep(1)
        continue
```

Kada završi ovaj korak, softver započinje proces postavljanja oglasa. Većina procesa jeste kombinacija prethodno pomenutih funkcija – kombinacija pretraživanja HTML elemenata na različite načine i vršenje funkcija kao klik iz *Selenium* biblioteke, ali ima i par novih funkcija vrednih za napomenu u daljem tekstu.

Dodatno za napomenu jeste varijacija između polu i potpune automatizacije – gde kod polu automatizacije, softver dodeljuje prethodno generisanu cenu, dok u potpunoj automatizaciji softver bira opciju “*Pozvati*” umesto fiksne cene.

Tokom generisanja opisa na stranici za postavljanje oglasa, softver iŝitava prethodno generisani *Word* dokument i u skladu sa njegovim formatom, vrŝi funkciju boldovanja, kosih slova i novog reda na prozorčić *iframe* elementa HTML dokumenta.

Vredno je napomenuti kako *Selenium* komunicira sa *iframe* elementom u ovom sluĉaju. A to je da ga tretira kao *HTML* dokumenat unutar *HTML* dokumenta koji trenutno fokusira. Tako da je potrebno vrŝenje funkcije *switch_to* iz *Selenium* biblioteke kada ŝelimo da fokusiramo prozor. Iz ovog razloga, blok koda ispod na svakoj iteraciji mora da klikne na dugme za *bold* ili *italic* tekst van *iframe* elementa, pa da “uĉe” u sam elemenat gde postavlja liniju koda i opet da izaĊe kako bi iskljuĉio format teksta prethodno ukljuĉen za tu liniju opisa. Ovaj proces se ponavlja onoliko puta koliko linija teksta u opisu fajla postoji.

```
for line in file:
    iteration += 1
    if "====" in str(line): # This here is a godsend!
        adDescription.send_keys(Keys.RETURN + Keys.LEFT_CONTROL)
        adDescription.send_keys(Keys.ARROW_DOWN)

    if (len(line) < 6 or re.search('[a-zA-Z]', line) == False): #SKIP EMPTY LINES (has to be
length based)
        continue
    elif iteration in italicLines: # ----- Italic Lines
        browserChrome.switch_to.default_content()
        italicButton.click() # ----- click-in
        browserChrome.switch_to.frame(textFrame)
        adDescription.send_keys(str(line))
        browserChrome.switch_to.default_content()
        italicButton.click() # ----- click-out
        browserChrome.switch_to.frame(textFrame)
    elif iteration in boldLines: # ----- Bold Lines
        browserChrome.switch_to.default_content()
        boldButton.click() # ----- click-in
        browserChrome.switch_to.frame(textFrame)
        adDescription.send_keys(str(line))
        browserChrome.switch_to.default_content()
        boldButton.click() # ----- click-out
        browserChrome.switch_to.frame(textFrame)

    else:
        adDescription.send_keys(str(line))
file.close()
```

Nakon toga, vrŝi se proces dodavanja slika preuzetih iz *.zip* fajla modela – to jest ako postoje.

Dodavanje slika se opet vrši kombinacijom prethodno pomenutih funkcija *Selenium* biblioteke – ali sa dodatnim funkcijama *AutoIT* modula. Prvobitno softver klikće na dugme za dodavanje pojedine slike. Zatim se otvara prozor za *upload* fajla. Ovde *AutoIT* stupa na snagu tako što uzima putanju direktorijuma i ukucava je u otvoren prozor. Vredno je napomenuti da će ovaj proces da se ponovi onoliko puta koliko slika postoje – ili ako se ne dostigne limit postavljen od strane sajta – 15 slika.

```
#Adding images
print(Fore.YELLOW + "Dodajem slike...")
acceptedFormat = ['.png', '.jpg', '.jpeg', '.bmp', '.gif']
limit = 0

for file in os.listdir(thingLocation):
    if limit == 15:
        print(Fore.YELLOW + "Naisao sam na limit slika. Obustajem.")
        break
    elif any(i in str(file).lower() for i in acceptedFormat):
        limit += 1
        print(Fore.YELLOW + "Dodajem sliku " + str(limit) + "...")
        adPhoto.click()
        autoit.win_active("Open")
        time.sleep(2)
        autoit.send(str(thingLocation) + "\\\" + str(file))
        autoit.send("{ENTER}")
        time.sleep(1)
    try:
        for i in range(4):
            autoit.win_active("Open")
            autoit.win_close("Open")
    except:
        pass
```

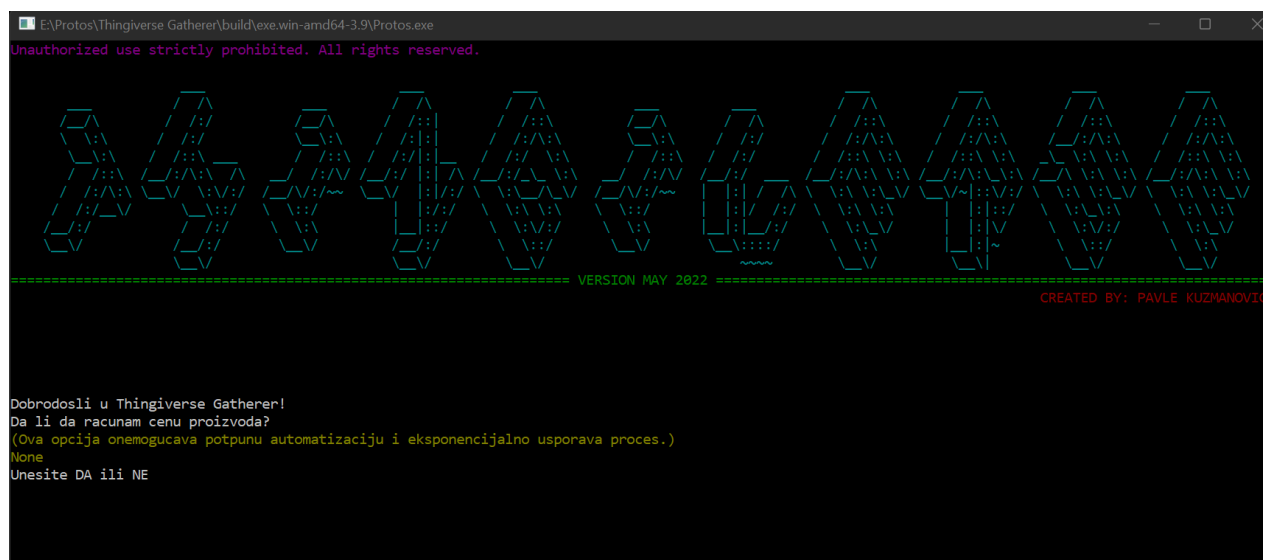
Ako je aktivna polu-automatizovana opcija, softver dodatno, sa otvorenim prozorom, daje korisniku par sekundi prostora za raspored slika po želji nakon uspešno postavljene poslednje slike. Ova opcija nije prisutna u potpunoj automatizaciji radi ubrzanja procesa.

Posle ovog bloka se izvršavaju poslednje par kombinacije koda iz ovog modula za klik na dugme, gde na kraju naš softver skladišti *URL* postavljenog oglasa u promenjivu koju šalje u prethodni modul radi ubacivanja podatka u našu bazu podataka primenom prethodno pomenutog koda.

2.1.4 Kreiranje .exe fajla za naš softver

Kada smo završili sa kucanjem našeg softvera, možemo ga pretvoriti u pravu *Windows* aplikaciju koristeći *cx_freeze*. To radimo tako preuzmemo jednostavan *setup.py* fajl sa stranice za dokumentaciju tog modula, promenimo putanju *.py* fajla koji želimo da pretvorimo u *.exe* fajl – u našem slučaju je to naš glavni modul. Na kraju samo pokrenemo preko *Python*-a fajl *setup.py* koji izgeneriše *.exe* fajl i u njega integriše sve naše module koje smo koristili, zajedno sa onima koje smo mi napravili.

Naša aplikacija je gotova i sprema za korišćenje.



Slika 8 – CLI interfejs gotove aplikacije za automatizaciju

2.2 Baza podataka

Za izradu baze podataka, koristimo softver *MySQL Workbench*. Da bi uopšte pristupili našoj bazi, moramo i upaliti MySQL server preko *XAMPP* softvera za lokalni hosting. Na početku vršimo inicijalna podešavanja tokom kreiranja naše baze – ili *schema* u MySQL Workbench-u. To su ime i Charset – utf8. Kada idemo na “*Apply*”, softver generiše kod za kreiranje naše baze podataka.

Što se tiče modela naše baze podataka, potrebne su nam četiri tabele.

1. **Tabela za proizvode** – Mesto za skladištenje podataka kao što su ime i cena predmeta, zajedno sa ID preuzet iz dela linka stranice originalnog sajta za 3D model, linkom ka stranici originalnog predmeta, kao i sa linkom stranice našeg oglasa na sajtu za kupoprodaju.
2. **Tabela za kupce** – Relacija za upisivanje osnovnih informacija o kupcu – u našem slučaju korisničko ime i broj telefona.
3. **Tabela za narudžbine** – Ujedno i međutabela proizvoda i kupaca, služi za skladištenje informacija o narudžbinama – ko je kupio koj proizvod, datum narudžbine, da li je narudžbina vraćena, materijal koji je korišćen i istrošena gramaža istog.
4. **Tabela za admina** – Tabela u kojoj se skladište informacije za logovanje na web portal za pristup bazi podataka.



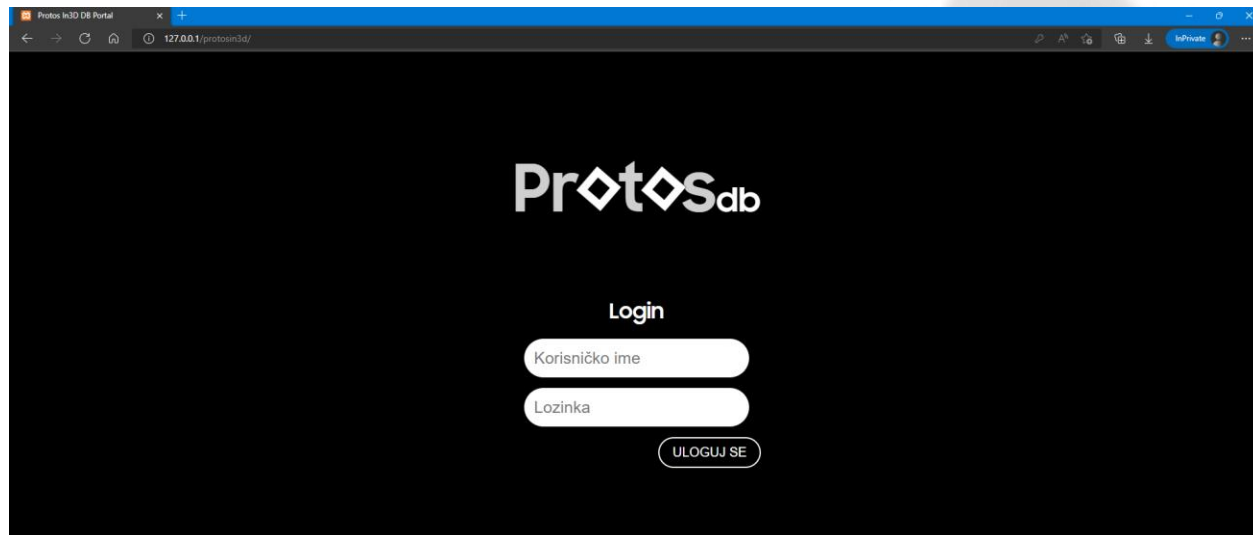
Slika 9 – Model baze podataka

2.3 Web Portal

Naš Web Portal treba da služi za izvršavanje par funkcija. Uz to, treba da bude obezbeđen odgovarajućim merama zaštite iz razloga što će služiti funkciju pristupanja osetljivim podacima – kao i vršenja funkcije izmena na istim. Sa time na umu, prvo i osnovno jeste zabranjivanje pristupa neovlašćenim licima korišćenjem koda za logovanje. Vredno je napomenuti da smo upalili *Apache* server preko *XAMPP*-a i kreirali folder za naš sajt u adekvatan pod-direktorijum kako bi omogućili kasnije izvršavanje *PHP* koda.

2.3.1 Stranica za logovanje

Prvo kreiramo opštu *HTML* formu sa 2 inputa – jedan sa korisničko ime i jedan za šifru. Dodajemo i dugme koje će vršiti funkciju predaje unetih podataka. Malo ćemo urediti to preko *CSS* koda i dobijamo sledeće – index stranicu našeg portala.



Slika 10 – Primer stranice za logovanje na našem web portalu

Kako bi omogućili uspešnu funkciju logovanja, želimo kreirati naš prvi upit u našu bazu podataka, a to je provera unetih inicijala. Logika iza ove funkcije jeste kada bude bilo kliknuto dugme, sajt učitava posebnu stranicu gde se izvršava *PHP* kod – u ovom slučaju kod upita za našu bazu podataka. Kod će preuzeti ime i šifru iz forme i sačuvati ih kao promenjive. Zatim, vrši upit na našu relaciju *admin* gde traži šifru za uneto korisničko ime – ova akcija ima 3 ishoda.

Prvi i drugi ishod jesu da ili korisničko ime ne postoji, ili je netačna kombinacija imena i šifre. U tom slučaju, naš sajt javlja da je operacija neuspešna i vraća nas na početnu stranu logovanja. Treći ishod jeste da je logovanje uspešno obavljeno i sajt nas preusmerava na stranicu konzole – stranica koja dozvoljava korisniku da vidi podatke baze podataka, da vrši pretragu nad njome, kao i da ubacuje nove informacije.

Do duše, potrebno je dodatno ubaciti još jedan deo koda kako bi sajt mogao prepoznati da li je korisnik ulogovan. Ovo je obavezan proces iz razloga dodatne zaštite koja odgovara na pitanje – *Šta ako napadač jednostavno zna URL koji direktno vodi na konzolu?*

Ovo rešavamo upotrebom sesija u *PHP-u*. Ispod ovog teksta se nalazi celokupan kod za logovanje – to jest kod koji uzima podatke iz forme, kod za upit na bazu podataka, specifičnije na tabelu *admin*, kod za paljenje sesije za logovanje, kao i kod za *failsafe* ako neki deo koda ne uspe da se izvrši.

```
if(isset($_POST['button_login'])) {  
    if(!empty($_POST['username']) && ($_POST['password'])) {  
        $admin_name=$_POST['username'];  
        $admin_pass=$_POST['password'];  
        $connection = mysqli_connect('localhost', 'root', '', "protos_in3d"); //The Blank string is the  
password  
        mysqli_select_db($connection, 'protos_in3d');  
  
        $result = mysqli_query($connection,"SELECT admin_password FROM admin WHERE  
admin_email='$admin_name'");  
        $row = mysqli_fetch_array($result);  
  
        if(($row[0]) == $admin_pass)  
        {  
            $_SESSION['loggedin'] = true;  
            header( "Location: console.php" );  
        } else {  
            header( "Location: fail.html" );  
        }  
        mysqli_close($connection);  
    }  
}
```

Dodatni korak jeste da svaku našu stranicu na sajtu obezbedimo blokom koda koji kaže da mora biti uspostavljena *loggedin* sesija kako bi se pristupilo istoj, ili , u suprotnom, da ih baci na stranicu za logovanje.

```
<?php  
    session_start();  
    if ($_SESSION['loggedin'] == false) {  
        header( "Location: index.php" );  
    }  
?>
```

Dodatno, radi bolje funkcionalnosti, dodajemo jednostavan kod na našu početnu stranicu za logovanje koji, ako je sesija već uspostavljena, odmah baca na stranicu naše konzole bez prikaza forme za logovanje.

```
<?php
    session_start();
    if ($_SESSION['loggedin'] == true) {
        header("Location: console.php");
    }
?>
```

2.3.2 Konzola

Stranica konzole jeste mesto gde ćemo moći vršiti pretragu i nadgledanje podataka naše baze. Sa obzirom da smo uspeali da je zaštitimo prvobitnim logovanjem – možemo rukovati informacijama. Konzola će se sastojati od 3 dela – dela za pretragu baze po kupcu i proizvodu, dela za dodavanje zapisa u tabele proizvoda, narudžbine i kupaca, i na samom kraju listing kompletnih tabela, kao i sve njihove redove i kolone.

Time rečeno, koristićemo 3 varijacije upita. Za prvi deo ćemo koristiti upite nad bazom gde ćemo uzimati podatke iz više tabela i prikazivati ih pomoću SQL funkcija *INNER JOIN*.

```
try {
    $search = $_POST['customer_search'];
} catch (\Throwable $th) {
    $search = "askjdaoidjasuoidhaskjdhud";
}

$query = "SELECT customer.customer_name, COUNT(product.product_kp_name),
SUM(order_is_returned) FROM protos_in3d.order INNER JOIN protos_in3d.customer ON order_customer_id =
customer.customer_id INNER JOIN protos_in3d.product ON order_product_id = product.product_id WHERE
customer_name LIKE '%$search%' OR customer_id LIKE '%$search%' OR customer_phone LIKE '%$search%' GROUP BY
customer_name;";

$result = mysqli_query($connection, $query);
echo "<table>";
echo "<tr><td>" . " Kupac " . "</td><td>" . " Kupio proizvoda " . "</td><td>" . "Vratio
proizvoda" . "</td></td></tr>";
while($row = mysqli_fetch_array($result)){

    if ($row[2] != 0) {
        echo "<tr style='background-color: #f55142;'><td>" . $row[0] . "</td><td>" . $row[1] .
"</td><td>" . $row[2] . "</td></tr>";
    } else {
        echo "<tr><td>" . $row[0] . "</td><td>" . $row[1] . "</td><td>" . $row[2] . "</td></tr>";
    }
}
```

Ovaj blok koda, specifičan za pretragu po kupcu, se izvršava klikom na dugme *Pretraga* i vrši upit na bazu SQL kodom koji se nalazi pod promenjivom *\$query*. Ispisuje 3 kolone u formi *HTML* tabele – ime kupca, koliko je proizvoda kupio i koliko je proizvoda vratio. U slučaju da je kupac vratio bilo koj proizvod, red će biti crven kako bi korisniku obratio dodatnu pažnju.

Pretraži po kupcu:

Rezultat pretrage

Kupac	Kupio proizvoda	Vratio proizvoda
Petar Hranisavljevic	3	2
Petar Tronkar	1	0
Toma Smakic	1	0

Slika 11 – Primer pretrage po kupcu

Drugi tip upita jeste ubacivanje podataka u bazu. Ima 3 varijacije, za proizvod, kupca i narudžbinu. Koristimo *SQL* funkciju *INSERT INTO*.

```
if(isset($_POST['button_buyer'])) {
    if(!empty($_POST['customer_id']) && ($_POST['customer_name']) && ($_POST['customer_phone'])) {
        $customer_id=$_POST['customer_id'];
        $customer_name=$_POST['customer_name'];
        $customer_phone=$_POST['customer_phone'];
        $connection = mysqli_connect('localhost', 'root', '', "protos_in3d");
        mysqli_select_db($connection, 'protos_in3d');
        if(mysqli_query($connection,"INSERT INTO customer VALUES ('$customer_id', '$customer_name',
'$customer_phone')"))
        {
            header( "Location: success.html" );
        } else {
            echo("Query Error");
        }
        mysqli_close($connection);
    }
}
```

Primer koda je uzet za ubacivanje zapisa u tabelu kupca preko forme zadate na našoj stranici.

Dodavanje zapisa

Novi Proizvod

Nova Narudžbina

Novi Kupac

Slika 12 – Forme za ubacivanje zapisa u bazu podataka

Treća varijacija upita jeste i najosnovniji. *SQL* funkcijom *SELECT* biramo sve podatke iz svake baze posebno i ispisujemo ih pomoću *HTML* elementa tablele.

Baza podataka

Kupci		
KupujemProdajem ID	Ime Kupca	Telefon Kupca
230431	Zoran Jovanovic	065245187
312251	Petar Hranisavljevic	064583491
381151	Petar Hranisavljevic	064594923
381152	Ceca	0645819048
381153	Marko Manic	0645817440
381157	Toma Smakic	063432092
768598	Uros	0603335974
840493	Buzgac Husein	0615606220
1657872	Beata Mihalic	0692806977

Narudžbina						
Narudžbina ID	Proizvod ID	Kupac ID	Datum Narudžbe	Vraćeno	Boja Materijala	Gramaza Materijala
12	2637487	381151	2022-04-25 14:15:03	1	FLUORESCENT	125
13	2637486	381151	2022-04-25 14:15:03	1	YELLOW	125
14	2637486	381151	2022-04-25 14:02:48	0	WHITE	125
15	2637486	312251	2022-04-25 15:06:41	0	BLACK	152
16	5354686	381157	2022-04-29 15:46:41	0	BLACK	152

Proizvodi					
Proizvod ID	Ime KP Oglasa	Cena KP Oglasa	Link KP Oglasa	Ime Thingiverse Proizvoda	Link Thingiverse Proizvoda
2637486	Držac za USB, SD i MicroSD kartice	235 RSD	https://www.kupujemprodajem.com/foto/memoriske-kartice/drzac-za-usb-sd-i-microsd-kartice/oglas/1268787077?filter_id=	https://www.thingiverse.com/thing/2637487	https://www.thingiverse.com/thing/2637486
2637487	Držac za USB, SD i MicroSD kartice	296 RSD	https://www.kupujemprodajem.com/foto/memoriske-kartice/drzac-za-usb-sd-i-microsd-kartice/oglas/1268787077?filter_id=	USB SD and MicroSD holder for wide USB sticks	https://www.thingiverse.com/thing/2637487
5340635	Upravljanje kablom i kablom	0 RSD	https://www.kupujemprodajem.com/muzika-i-instrumenti/gitarska-oprema-ostalo/upravljanje-zicama-drzaca-kabla/oglas/136741811	Cable holder wire management	https://www.thingiverse.com/thing/5340635
5354686	Konj u zatvoru - Mesečev vitez 2022	0 RSD	https://www.kupujemprodajem.com/antikviteti-i-umetnička-dela/stane-figure/konj-u-zatvoru-mesecev-vitez-2022/oglas/136548146	Khonshu Imprisoned - Moon Knight 2022	https://www.thingiverse.com/thing/5354686

Slika 13 – Ispis kompletnih relacija baze podataka

Dodajemo još jedan detalj na našem ispisu kompletne baze – filter. Služi funkciju filtriranja baze po unešenim slovima. Funkcija je pisana u *Javascript* jeziku i opaljuje se na svakoj promeni u input polju.

```
let a = document.getElementById("filter");
let b = document.getElementsByClassName("searchable");
a.addEventListener("input", function () {
    var c = a.value;
    for (let d = 0; d < b.length; d++) {
        if (!(b[d].innerHTML.includes(c))) {
            b[d].style.display = "none";
        } else {
            b[d].style.display = "table-row";
        }
    }
})
```

Ovim obuhvatamo sve delove našeg portala za pristup bazi podataka. Završetkom ovog dela, naš sistem je potpun i može komunicirati između sebe. Dodatno je moguće našu bazu podataka, kao i naš web portal objaviti na javnom serveru sa mogućstvom pristupanja preko globalne internet mreže, gde je samo potrebno promeniti parametre kao što su *IP* adresa baze podataka, parametri za pristup istoj i par ostalih podataka. Ovo bi omogućilo uspešno izvršavanje našeg softvera za automatizaciju sa različitih lokacija, pa i više pokretanja u isto vreme.

Zaključak

Naš sistem pruža rešenje za izvršavanje jako repetitivnog posla i obrade podataka i svodi ga pa i na potpunu automatizaciju gde je jedini obavezan ljudski input prvobitno pokretanje softvera. Dodatno pruža mogućnost lakog proširenja na druge platforme koristeći modularni pristup izgradnje softvera. Uz to, izgradnjom baze podataka, kao i web portala za pregled i manipulisanje istom, moguće je dodatna analiza podataka, izgradnja informacija od istih što omogućava dodatnu mogućnost analize tržišta i optimizaciju prodaje, cena i sličnih parametara radi poboljšanja načina poslovanja što je potreba pri kreiranju uspešnog biznisa ili optimizaciju postojećeg. Ovaj sistem može da se implementira za bilo koju firmu koja se bavi sličnim tržištem, mada je i moguće adaptiranje samog sistema i na biznise sa drugim tipom ponude ili prodaje.

Implementacijom opcionih *quality-of-life* funkcija kao što je, primera radi, filter baze podataka ili čuvanje podataka u posebnom fajlu u svrhe log-ovanja informacija tokom obrade podataka kroz softver automatizacije, ali i optimizacijom obaveznih, značajno povećavamo kvalitet našeg sistema i, samim tim, cenu istog. Time se odvaja od moguće konkurencije.

Što se tiče automatizacije na globalnom spektru, sa sve većim porastom potražnje radnika za poslove koje traže više ljudskih veština i intelekcie, automatizacija omogućava rešavanje više problema u isto vreme, bilo za firme ili generalnu populaciju stanovništva. Firmama predstavlja eksponencijalno umanjivanje troškova u smislu redukcije zaposlenih za poslove gde je standardizacija procesa potpuna – gde je potrebno svesti broj zaposlenih na samo one koji vrše *monitoring* na odvijanje automatizovanih procesa, bilo softvera ili hardvera i dodatno, firma samo treba otplatiti jednokratnu sumu novca za izradu sistema automatizacije. Dodatno, automatizacija omogućava pojedincu, bilo svesno ili nesvesno, viši nivo edukacije – u smislu da, iako automatizacija može dovesti do smanjenja pozicija nižih nivoa, tako otvara vrata za više pozicije, što, ujedno može i povećati inspiraciju i volju za većim uspehom pojedinca. Automatizacija otklanja potrebu ljudske snage za veliki broj pozicija i tako eliminiše tražnju za iste što dovodi da radnici ne stiču veštine za takve poslove i teže ka drugim, postojećim mestima za rad koji ne moraju, ali mogu biti na višem nivou. Sa sve većim brojem automatizovanih procesa na svetu, verovatnoća je sve veća da to bude realnost.

Literatura

Groover, M. P. (n.d.). *Automation*.

Oxford. (n.d.).

W3Techs.com (2022, April 1)

Wikipedia. (n.d.).

M. Šarac, D. Radovanović, A. Jevremović and M. Mravik, Internet marketing, Univerzitet Singidunum, Beograd, 2020.