Zbornik radova Fakulteta tehničkih nauka, Novi Sad

**UDK: 004.9**
**DOI: https://doi.org/10.24867/01BE07Odalovic**

# APLIKACIJA ZA PRAVLJENJE MOCKUP-A I NJENA PRIMENA

# MOCKUP CREATION APPLICATION AND ITS USAGE

Mirko Odalović, *Fakultet tehničkih nauka, Novi Sad*

**Oblast – ELEKTROTEHNIKA I RAČUNARSTVO**

**Kratak sadržaj –** *Ovaj članak treba da demonstrira probleme koji se javljaju tokom implementacije aplikacije za pravljenje* ***mockup-a*** *kao i način na koji su spomenuti problemi rešeni. Takođe pojasniće se i tehnologije u kojima je implementirana aplikacija kao i pojam* ***mockup****.*

**Ključne reči:** ***Mockup-a****, WPF, MVVM, Dizajn sabloni*

**Abstract** – *This article should demonstrate problems that appear during implementation of application for creating mockups and a way in which the following problems are solved. Technologies used for application implementation and term mockup will also be explained.*

**Keywords:** ***Mockup****, WPF, MVVM,* ***Design patterns***

## 1. UVOD

U članku se opisuje način na koji je napravljena aplikacija za pravljenje ***mockup-a*** kao i primena aplikacije. U prvoj celini pojašnjeni su pojmovi čije razumevanje je neophodno kao bi se znalo za šta aplikacija služi. Pojam koji se objašnjava u prvoj celini je ***mockup*** jer je ključni pojam čije razumevanje je neophodno za uspešno praćenje i razumevanje ovog članka.

U drugoj celini opisan je način komunikacije pomoću ***mockup-a*** izmedju dizajnera i programera, to su dve uloge koje imaju najviše dodirnih tačaka sa ***mockup-om***.

U trećoj celini opisana je sama ***mockup*** aplikacija odnosno funkcionalnosti koje podržava, kako bi svojom bogatom ponudom funkcionalnosti privukla što više korisnika a isto tako ih i zadžala.

U četvrtoj celini opisani su dizajn šabloni, koji se koriste u implementaciji aplikacije, da bi se mogla pratiti demonstracija implementacije programskog rešenja.

U petoj celini demonstrirane su tehnologije u kojima je razvijena aplikacija za pravljenje ***mockup-a***. A to je *WPF* aplikacija gde se prilikom implementacije vodi računa da bude podržan *MVVM* šablon.

U poslednjoj šestoj celini opisani su problemi koji se javljaju tokom imaplementacije kao i način na koji su isti rešeni. Isto tako u ovoj celini je demonstrirana primena dizajn šablona na spomenute probleme.

**NAPOMENA:**
**Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.**

## 2. MOCKUP

U jednoj reči ***mockup*** se predstavlja kao skica, pomoću koje se opisuje izgled korisničkog interfejsa aplikacije. ***Mockup*** veoma često predstavlja nacrt dizajna ili čak stvarni dizajn aplikacije. Dobro kreiran ***mockup*** dobro reprezentuje strukturu informacija, vizualizaciju sadržaja i demonstraciju osnovnih funkcija na statičan način. Izgradnja ***mockup-a*** pomaže ostvarivanju idealne ravnoteže izmedju krajnjeg korisničkog interfejsa i jednostavnoj modifikaciji istog. Danas se ***mockup*** može brzo napraviti u poređenju sa vremenom potrebnim da se napravi aplikacija. Koristan je u fazi kada je potrebno osmisliti korisnički interfejs aplikacije, pomaže da osobe zadužene za kreiranje korisničkog interfejsa budu skoncentrisane samo na svoj deo posla, samim tim poboljšava kreativnost. ***Mockup*** je naročito koristan kada je potrebno na brz način privući potecijalne klijente, jer kada se klijentu ponudi nešto vizualno(konkretno), na njega će to ostaviti bolji utisak nego da se ponudi sama ideja. Na početku razvoja aplikacije potroši se više vremena kada se prave ***mockup-i***, ali se zato puno vremena uštedi u budućnosti jer nakon uspešno napravljenih ***mockup-a*** zna se tačno šta se radi i zna se da je to predstavljeno klijentu i da se klijent složi sa tim rešenjem. Uobičajena mera koja se koristi kako bi pokazala koliko su napravljeni ***mockup-i*** precizni ili tačni naziva se *vernost* (eng. ***fidelity***). Vernost je mera koliko je ***mockup*** sličan krajnjem proizvodu. Nisku vernost imaju ***mockup-i*** koji predstavljaju jednostavne skice. Visoku vernosti imaju ***mockup-i*** koji savršeno predstavljaju korisnički interfejs aplikacije nazivaju se još i piksel savršeni. Kada se prave ***mockup-i*** sa visokom vernosti treba imati na umu da je za takve ***mockup-e*** potrebno dosta više vremena kako bi se napravili, nego što je potrebno da se naprave ***mockup-i*** sa niskom vernosti. Zato se često prave ***mockup-i*** sa srednjom vernosti, jer su dovoljno verni da se na osnovu njih može napraviti korisnički interfejs aplkacije, a opet su dovoljno jednostavni da se mogu brzo napraviti.

## 3. MOCKUP KAO SREDSTVO KOMNIKACIJE

Da bi faze u životnom ciklusu razvoja softvera kroz koje je potrebno proći kako bi se stiglo do finalnog proizvoda odvijale što lakše i bez velikih zastoja potrebno je obezbediti dobru komunikaciju između ljudi koji učestvuju u tim fazama, a isto tako je važno da bude dobra komunikacija izmđu ljudi koji pripadaju raličitim fazama. U nastavku spominju se samo dve faze životnog ciklusa razvoja softvera jer u tim fazama se javalju uloge koje su od ineteresa odnosno uloge koje imaju najviše dodirnih tačaka sa ***mockup-om*** a to su uloga dizajner i uloga programer.

U fazi dizajna potrebno je osmisliti korisnički interfejs buduće aplikacije. Pravljenje korisničkog interfejsa aplikacije je posao dizajnera, znači da se uloga dizajnera javalja u fazi dizajna životnog ciklusa razvoja softvera. U fazi razvoja potrebno je razviti samu aplikaciju. Posao programera je da razvije aplikaciju, znači da se uloga programera pojavljuje u fazi razvoja životnog ciklusa razvoja softvera.

Grafiči dizajneri su obučeni da koriste računarske programe za stvaranje slike, kao što su logo ili sam dizajn aplikacije. Dizajneri za cilj imaju stvaranje efikasnog i atraktivnog korisničkog interfejsa, kako bi korisnici koji koriste njihov korisnički interfejs bili zadovoljni. Prvenstveno dizajneri dizajniraju raspored grafičkih komponenti na korisničkom interfejsu(tj. poziciju grafičkih komponenti) pored toga odlučuju o boji, fontu, stilu i veličini grafičkih komponenti, kako bi dobili željen korisnički interfejs. Posao dizajnera je: zna da koristi aplikacije za dizajniranje rasporeda elemenata i uređivanje slika, predstavlja svoje projekte klijentima ili poslodavcima, uvaži promene koje su mu tražili klijenti i poslodavci i bude u toku sa najnovijim tehnologijama i softverom.

Programeri koriste programske jezike pomoću kojih pišu kod za aplikacije kako bi obavili svoj deo posla. Obično poznaju jedan ili više programksih jezika kao što su *Java*, *C#*, *C++*, itd. Njihov posao nije samo da razvijaju programe(aplikacije) od nule, već treba da znaju na postojećim aplikacijama koje nisu oni razvijali da rešavaju probleme ili dodavaju nove funkcionalnosti ako klijent to zahteva, tj. da se lako i brzo snalze u drugim programskim rešenjima koje oni nisu razvijali. Programeri oživaljvaju projekat, od dokumentacije i ***mockup-a*** koji su napravljeni u ranijim fazama, programeri kodiranjem prave konkretan proizvod koji korisnicima omogućava da ga primene na svoje problem zbog kojih je i sam proizvod napravljen. Posao programera je: održavanje i ažuriranje programa koji već postoje, poznavanje velikog broja programskih biblioteka kako bi brže i lakše mogao da radi, testranje razvojnog koda i dizajniranje i potvrda funkcionalnosti koda.

Nakon objašnjenja uloga programera i dizajnera vraćamo se na samu temu ***mockup*** kao sredstvo komunikacije između dizajnera i programera. Tradicionalno se posao programera i dizajnra gleda skroz odvojeno ali da bi razvili dobru aplikaciju neophodna je saradnja. Na kraju svakog dana i programer i dizajner rade da bi ostvarili isti cilj, da su korisnici krajnje aplikacije zadovoljni, a ako su oni zadovoljni onda je i klijent koji je od njih tražio da napravi aplikaciju takođe zadovoljan. Da bi ostvarili svoj cilj aplikacija mora da radi, a to je u nadležnosti programera, isto tako mora da bude prilagođen korisnički interfejs kao i jednostavna interakcija korisnika sa aplikacijom što je u nadležnosti dizajnera tj. mora dobro da dizajnira aplikaciju. Komunikacija između programera i dizajnera upotrebom ***mockup-a*** se pokazala jako dobro, kao što je poznato slika govori više od hiljadu reči. Dizajner ima ideju ali koliko bi njemu trebalo vremena da tu ideju prenese programeru, može se desiti da programer u glavi stvori sasvim drugu sliku. Opšte je poznato programeri razmišljaju na različite načine, programer se već sretao sa nekim problemima i on će nastojati da ako je to moguće ponudi rešenja koja je već imao, da bi mu bilo lakše da uradi svoj deo posla. Dok dizajneri ima svoj pogled na svet i on nije svestan koliko je nešto teško za programera, dizajner nema ta ograničenja. Dizajner je zadužen za ***mockup***, i taj ***mockup*** se prosleđuje u vidu slike programeru, programer na osnovu slike zna koja je ideja dizajnera i može ideju da implementira, na ovaj način se smanjuje šansa za nesporazum u komunikaciji.

## 4. MOCKUP APLIKACIJA

Aplikacija za pravljenje ***mockup-a*** dizajneru treba da pomogne kao što samo ime kaže u pravljenju ***mockup-a***, na kraju tako napravljen ***mockup*** može da izveze u sliku, i da u formatu slike može da pošalje klijentu, programeru. Takođe aplikacija treba da omogući korisniku snimanje projekta koji je započeo u samoj aplikaciji kao bi posle mogao da nastavi sa radom, kako ne bi izgubio sve prethodno što je napravio. A isto tako kada šalje drugom dizajneru bolje da pošalje sam fajl od projekta kako bi dizajner kome je poslat fajl mogao da učita postojeći projekat i ukoliko želi da doda neke svoje ideje, može da doradi sam ***mockup*** bez da mora sve ispočetka već samo da nastavi na već postojećem projektu.

Korisnicima aplikacija treba da bude omogućeno dodavanje slika na nešto što bi trebalo da predstavlja ***mockup***. Pored slika korisnici mogu da ubacuju i tekstualni sadržaj na svoje ***mockup-e***, kako bi mogli da prave željene ***mockup-e***. Još jedna bitna stvar koja je potrebna korisnicima je rad sa slikama i tekstovaim, korisnik u svakom trenutku može da promeni sliku ili tekstualni sadržaj gde god želi upotrebom miša. Korisniku treba da budu prikazane predefinisane slike koje može da koristi za pravljenje svojih ***mockup-a***, pored predefinisanih slika korisnik trebalo da ima mogućnost dodavanja svoje slike koja se ne nalazi u predefinisanim slikama.

Što se tiče tekstualnog sadržaja korisnik može da podešava sam tekstualni sadržaj kako on želi, aplikacija treba da podržava promenu boje, veličine i fonta tekstualnog sadržaja kao i promenu samog sadržaja. Kako bi korisnik moga preciznije da pozicionira slike i tekstualni sadržaj na ***mockup-u***, korisnik ima mogućnost povećavanja i smanjivanja samog ***mockup-a***. Takođe korisnik u svakom trenutku može da poništi akcije koje je napravio, preko funkicije ***unod*** koja se nalazi u aplikaciji, a ima i funkciju ***redo*** kako bi korisnik mogao da opozove akciju.

Da bi korisnik mogao lakše da radi sa slikama i tekstualnim sadržajem obezbeđene su funkcije: ***cut***, ***copy***, ***paste***. Ukoliko je korisniku potrebna slika ili tekst koji je već koristio u projektu, ubačene su funkcije koje korisnik može da iskoristi za kopiranje istih slika ili tekstova, i ne mora da radi isti posao više puta. Korisnik ima mogućnost da postavi boju pozadine kao i mogućnost postavke željene slike za pozadinu ***mockup-a***.

Korisnik može da menja dimenzije samog ***mockup-a*** u zavisnosti od okruženja i zahteva samog klijenta. Ako korisnik želi da obriše neki tekstualni sadržaj ili sliku koju je prethodno dodao na ***mockupa***, aplikacija to obezbeđuje korisniku. Korisnik prilikom dodavanja nove slike u ***mockup*** ima mogućnost pregleda slike u prozoru koji je namenjen za pregled slike. U aplikaciji korisnik imam mogućnost da napravi novi projekat.

## 5. STATE I KOMAND DIZAJN ŠABLON

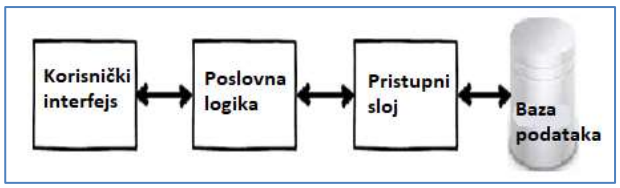
***State*** je dizajn šablon ponašanja koji imeplementira mašinu stanja na objektno orjentisan način. Sa ***state*** šablonom, mašina stanja se implementira primenom svakog pojedinačnog stanja koje implementira interfejs koji predstavlja stanja i imeplementira prelazak u sledeće stanje. ***State*** šablon može se tumačiti kao strategiski obrasac koji je u stanju da promeni trenutnu strategiju kroz pozive metoda koje su definisane u interfejsu stanja. ***State*** šablon se najčešće koristi u programiranju kako bi se enkapsuliralo različito ponašanje na osnovu njegovog unutrašnjeg stanja. Predstavlja čist način da objekta promeni svoje ponašanje bez pribegavanja uslovnim izrazima i na taj način se olakšava kasnije održavanje.

U objektno orjentisanom programiranju, komandni šablon predstavlja dizajn šablon ponašanja u kojem se objekat koristi za enkapsuliranje svih informacija potrebnih za izvođenje akcije ili pokretanje dogadja kasnije. Ove informacije uključuju imena metoda, objekata koji poseduju metode kao i vrednosti parametara koje treba proslediti metodama.

## 6. WPF i MVVM

***Windows Presentation Foundation*** (WPF) je sistem nove generacije za prezentaciju i izgled ***Windows*** klijentskih aplikacija sa vizuelnim boljim korisničkim iskustvom. Pomoću *WPF-a* može se kreirati širok opseg samostalnih aplikacija i aplikacija koje se izvršavaju u internet pretraživaču. Samo jezgro *WPF-a* je nezavisno od rezolucije i vektorkog načina crtanja, napravljeno da iskoristi prednosti modernih grafičkih procesora. *WPF* proširuje jezgro sa sveobuhvatnim skupom funkcija razvoja aplikacije koje uključuju ***Extensible Application Markup Language*** (XAML), kontrole, povezivanje podataka, *2D* i *3D* grafici, animacije, stilove, šablone, dokumente, medije i teks. *WPF* je uključen u ***Microsoft*** *.NET* platformu, tako da se mogu napraviti aplikacije koje sadrže u sebi elemente iz *.NET* platforme.

Život je razvoj koji počinje kada se čovek rodi, on vremenom uči i razvija se kako bi od deteta postao odrasla osoba. Isto važi i za arhitekturu softvera, u početku se radi sa osnovnim strukturama a zatim se razvija i napreduje prema traženim zahtevima i potrebama. U ***Model View View Model*** (*MVVM*) arhitekturi projekat je podeljen u tri logičke celijne: korisnički interfejs (*UI*), poslovni sloj (eng. ***Bisiness Logic***) i pristupni sloj (eng. ***data access layer***). Svaki od ovih slojeva odgovoran je za svoj deo posla. Na slici 1 prikazana je *MVVM* arhitektura.

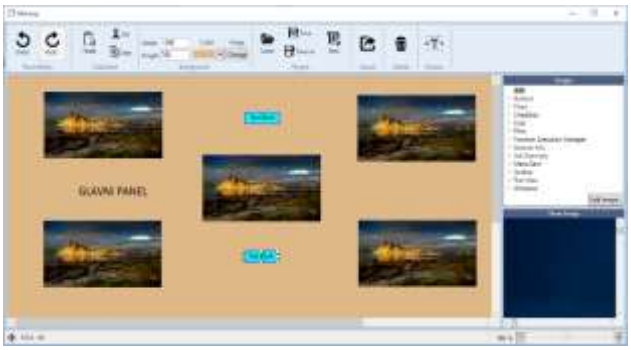



Slika 1. *MVVM arhitektura*

Korisnički interfejs zadužen je za prezentaciju podataka, poslovni sloj vodi računa o validaciji podataka i sloj za pristup podacima zadužen je za rukovanje sa bazom podataka. Prednosti troslojne arhitekture: zadržava promene (promene u jednom sloju ne utiču na druge slojeve), ponovna iskoristivost (povećana je ponovna iskoristivost jer su svi slojevi odvojeni, samostalni i pojedinačni delovi).

## 7. IMPLEMENTACIJA APLIKACIJE

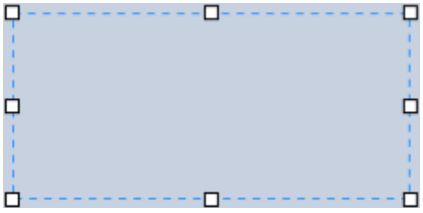
Aplikacja je pisana u programsko jeziku *C#*, pomoću *WPF-a* i primenom *MVVM* šablona. Na slici 2 prikazan je korisnički interfejs ***mockup*** aplikacije.




Slika 2. *Korisnički izgle mockup aplikacije*

Glavni panel, samo ime kaže, predstavlja glavni deo ove aplikacije i on prekriva veći deo korisničkog interfejsa. Pomoću glavnog panela se prikazujue korisniku ***mockup***, isto tako se omogućuje da pravi ili menja već postojeće ***mockup-e***. Jedna od glavnih funkcionalnosti koje korisnik može da radi je ubacivanje druge komponente u glavni panel. Glavni panel je trebalo da podrži rad sa više komponenti različitog tipa npr. može da primi liniju, pravougaonik, neku tekstualnu komponentu i neku grafičku komponentu unutar sebe. Da bi se podržao rad sa više različitih komponenti napravljena je klasa koja predstavlja baznu komponentu koja sadrži sve osobine, metode i komande koje se neophodne glavnom panelu da bi mogao da podrži bazne funkcije za komponente kao što je, na određenim pozicijama da iscrta komponentu, da zna koliko su komponente visoke i široke. I sada kada postoji ova bazna komponenta lako mogu da se prave nove komponente. Da bi se napravila nova komponenta potrebno je samo naslediti baznu komponentu i dodati osobine i metode koje su karakteristične za komponentu koja se pravi. I naravno za svaku komponentu potrebno je definisati ***View***, da bi glavni panel znao da nacrta komponentu. Ovakvom implementacijom obezbeđeno je lako dodavanje novih komponenti.

Selekcija komponenti, korisnik može da selektuje željenu komponentu koja se nalazi na glavnom panelu. Da bi selekcija bila podržana napravljena je grafička komponenta koja se iscrtava oko komponente koja treba da bude selektovana i predstavlja selekciju. Komponenta koja predstavlja selekciju se sastoji od više drugih grafičkih komponenti, tako da je napravljena grafička komponenta koja predstavlja četvorougao. Komponenta za selekcijiju sadrži devet komponenti četvorougla, kao što se vidi na slici 3.



Slika 3. *Grafička komponenta za selekciju*

Grafička komponenta se sastoji od jednog glavnog pravougaonika i osam malih kvadrata koji se nalaze na uglovima i polovinama stranica glavnog pravougaonika.

Pomeranje i promena dimenzije selektovane komponente, pored selekcija korisnik može i da pomera i menja dimenziju selektovane komponente. Pošto ove dve funkcionalnosti trebaju da se pozivaju na isti događaj, tj. kada korisnik pritisne levi klik i krene da pomera miš a i dalje nije pustio klik, onda bi trebalo u zavisnosti gde je korisnik pritisnuo klik na nekoj selektovanoj komponenti da se dogodi pomeranje komponente ili promena dimenzije. Ova funkcionalnost je implementirana upotrebom ***state*** šablona. Za svaki četvorougao koji se nalazi u komponenti za selekciju postoji jedno stanje, jer kada se klikne na bilo koji od tih četvorouglova različite operacije treba se izvrše nad komponentom koja je selektovana. Ako se klikne na glavni pravougaonik onda treba da se pomera selektovana komponenta kada se pomera miš ako klik nije pušten. A ako korisnik pritisne levi klikne na neko od kvadrata onda u zavisnosti na koji je pritisnu menja se dimenzija ili pozicija selektovane komponente. Npr. ako korisnik klikne na gornji levi kvadrat otiće se u ***LeftTopResizeState*** stanje, u tom stanje je definisano šta treba da se desi sa komponentom kada korisnik pomer miš. U ovom slučaju komponente treba da menja dimenzije a isto tako treba i da joj se menja pozicija jer se koridinate računaju u četvrtom kvandrantu.

Slikovna komponenta, korisnik na glavni panel može da dodaje slkovnu komponentu. Ova komponenta ko i sve do sad komponente da bi se napravila mora da nasledi baznu komponentu, dodatna osobina koju poseduje ova komponenta je putanja do slike koja treba da se iscrta unutar ove komponente. I kao što je već rečeno treba napraviti i izgled same komponente i sada na brz i jednostavan način je napravljena nova komponenta.

Tekstualna komponenta, korinik na glavni panel može da dodaje i tekstualnu komponentu. Isti je postupak pravljenja bio kao kod slikovne kompnente stim da ova komponenta ima dve dodatne osobine a to su sam tekst i još jednu osobinu u kojoj se nalaze sve neophodne informacije za sam izgled teksta kao što su font, boja, veličina, da li treba boldvati tekst.

***Undo***, ***Redo*** funkcije, omogućuju korisniku da poništava sve akcije koje je stvarao dok je pravio ***mockup*** na glavnom panelu. Isto tako ako je već poništio neke akcije može da ih opozove. Prilikom implementacije ovih funkcionalnosti korišten je komandni šablon. Sve akcije koje korisnik može da uradi na glavnom panelu su implementirane preko komandi. Komande koje postoje su: komanda za promenu pozicije i dimenzije komponente, komanda za dodavanje, komanda za brisanje, komanda za isecanje, komanda za nalepljivanje, komanda za promenu fonta tekstualnoj komponenti.

Mehanizam zumiranja, korisnik ima mogućnost da uveća glavni panel ako mu je sadržaj koji se nalazi u glavnom panelu sitan, isto tako ima mogućnost da uveća glavni panel ako to želi. Ovaj mehanizam je implementiran upotrebom ***state*** šablona. Svako stanje u sebi definiše koje je sledeće a koje prethodno stanje i određuje parametre koji definišu trenutno stanje. Korisnik ima mogućnost da vidi glavni panel u sledećim razmerama: 25, 33, 50, 67, 75, 80, 90, 100, 125, 150, 175, 200, 250, 300, 400 i 500. I za svaku ovu razmeru postoji posebno stanje, stanje 100 predstavlja redovan prikaz glavnog panela.

Čuvanje i otvaranje projekta, korisnik ima mogućnost da sačuva projekat koji trenutno radi, kasnije kada želi može da ga otvori i da nastavi gde je stao. Kako bi se ovo implementiralo bilo je potrebno da sve klase koje se snimaju budu serijabilne odnostno anotirane anoticajom „***Serializable***“, kao i sve osobine koje se u njima nalaze. Problem koji se ovde javalje jeste to što klase koje su u sebi imale osobine koje nisu serijabilne nisu mogle da se snime u fajl, a ne postoji mogućnost da te osobine postanu serijabilne. Način na koji je ovo rešen je da su dodate nove osobine koje su serijabilne i u koje se prilikom snimanja u fajl kopiraju informacije koje sa nalaze u osobinama koje nisu serijabilne, da bi se posle kada se čita iz fajla postojala informacije da bi se kreirali ti objekti koji nisu bili serijabilni.

## 8. ZAKLJUČAK

U ovom članku demonstrirana je implementacija aplikacije za pravljenje ***mockup-a***. Pored same aplikacije prikazana je i primena aplikacije, kojim ulogama u životnom razvoju softvera ova aplikacija pomaže. I na koji način te uloge mogu da iskoriste aplikaciju kako bi rešili svoje probleme. Isto tako opisani su glavni pojmovi zbog kojih je i sama aplikacija napravljena kako bi se razumeo rad aplikacije kao i sam primena. Odluka da se pravi *WPF* aplikacija prateći *MVVM* šablon u programskom jeziku *C#* se pokazala kao dobra jer su uspešno i kvalitetno rešeni svi problem na koje se nailazilo u toku imlementacije. Kao i većina današnjih aplikacija i ova aplikacija se može poboljšati, jedan od predloga za poboljšanje je da se korisniku omogući da pretražuje slike pomoću meta podataka, kako bi brže pronašao željenu sliku koju želi da stavi na glavni panel, a ne kako je sada slučaj da korisnik mora da prođe kroz sve slike dok ne nađe sliku koju želi.

## 9. LITERATURA


[1] https://www.vikingcodeschool.com/web-design-basics/what-are-mockups
[2] https://study.com/articles/graphic_designer_vs_-programmer.html
[3] https://sourcemaking.com/design_patterns
[4] https://docs.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/aa970268(v=vs.100)
[5] https://www.codeproject.com/Articles/819294/WPF-MVVM-step-by-step-Basics-to-Advance-Level, 2014


**Kratka biografija:**


**Mirko Odalović** rođen je 20.01.1994. godine u Bačkoj Topoli. Završio je srednju tehničku školu Ivan Sarić u Subotici 2013. godine. Fakultet tehničkih nauka u Novom Sadu je upisao 2013. godine. Ispunio je sve obaveze i položio je sve ispite predviđene studiskim programom.