



UNIVERZITET SINGIDUNUM
DEPARTMAN ZA INFORMATIKU I RAČUNARSTVO

Filip Arnerić

WEB APLIKACIJA ZA IZRADU
VIZUELNOG IDENTITETA KOMPANIJE
- Diplomski rad -



Beograd, Mart 2011.



UNIVERZITET SINGIDUNUM
DEPARTMAN ZA INFORMATIKU I RAČUNARSTVO

WEB APLIKACIJA ZA IZRADU VIZUELNOG IDENTITETA KOMPANIJE

- Diplomski rad -

Mentor:
Prof. dr Mladen Veinović

Student:
Filip Arnerić

Broj indeksa:
70/2006



Beograd, Mart 2011.



DEPARTMAN ZA INFORMATIKU I RAČUNARSTVO

UNIVERZITET SINGIDUNUM
DEPARTMAN ZA INFORMATIKU I RAČUNARSTVO
Beograd, Zorana Đinđića 44

Broj: _____/2005

Kandidat: Filip Arnerić

Broj indeksa: 70/2006

Smer: Programiranje i projektovanje

Tema: Web aplikacija za izradu vizuelnog identiteta kompanije

Zadatak: Isprojektovati, razviti i dokumentovati Web aplikaciju za izradu vizuelnog identiteta kompanije. Za realizaciju korisničkog interfejsa koristiti HTML, CSS i JavaScript sa bibliotekom jQuery, dok sam mehanizam (eng. *engine*) za izradu likovnih elemenata treba da bude izrađen korišćenjem PHP programskog jezika.

Datum odobrenja rada:
Beograd, 3.2.2011.

Mentor

Prof. Dr Mladen Veinović

Dekan

Prof. Dr Mladen Veinović

Web aplikacija za izradu vizuelnog identiteta kompanije

Sažetak: Kao osnovna ideja ovog rada uzeta je mogućnost dizajniranja likovnih elemenata vizuelnog identiteta korišćenjem Web tehnologija. U ovom radu biće detaljno opisani pojam vizuelnog identiteta i sve tehnologije koje su korišćene u izradi aplikacije (HTML, CSS, JavaScript, jQuery i AJAX i programski jezik PHP). Pored toga biće prikazan i model, zajedno sa principima rada aplikacije i primer upotrebe iste i to kroz proces kreiranja vizit karte. Uz sve ovo izneta su i moguća unapređenja, kao i plan razvoja aplikacije u budućnosti.

Ključne reči: Vizuelni identitet, Web tehnologije, Web aplikacije, Modelovanje, Vizit karte

Web application for creating visual identity of company

Abstract: The basic idea of this paper is the possibility of creating pictorial elements of visual identity by means of Web technologies. This paper will describe in detail the concept of visual identity as well as all the technologies that were utilized for the purpose of creating the application (HTML, CSS, JavaScript, jQuery, AJAX i PHP programming language).

Additionally, an application model will also be presented, together with the principles underlying the application as well as a practical example - the process of creating a business card.

Furthermore, possible improvements as well as a development plan for the future have been presented.

Key words: Visual identity, Web technologies, Web applications, Modeling, Business cards

Sadržaj

1. Uvod.....	8
1.1. Vizuelni identitet.....	8
1.1.1. Osnovni principi identifikacije (vizuelne konstante).....	8
1.2. Ideja projekta.....	9
1.3. Desktop i web aplikacije.....	9
1.3.1. Tradicionalni pristup izradi.....	9
1.3.2. Obeležja web aplikacija.....	9
1.3.2.1. Aplikacija je centralizovana (na jednom mestu)	10
1.3.2.2. Potreban je samo pretraživač (eng. Browser).....	10
1.3.2.3. Web aplikacije su sigurnije	10
1.3.2.4. Web aplikacije su sve popularnije.....	10
1.3.3. Prednosti i mane (ukratko)	11
2. Web tehnologije.....	12
2.1. HTML.....	12
2.2. PHP.....	13
2.2.1.1. PHP/FI.....	13
2.2.1.2. PHP 3.0.....	14
2.2.1.3. PHP 4.0.....	14
2.2.1.4. PHP 5.0.....	14
2.2.1.5. Uobičajen scenario po kom se izvršavaju PHP skripte je sledeći :.....	15
2.3. Javascript, jQuery, AJAX.....	16
2.3.1. Javascript.....	16
2.3.2. JQuery.....	17
2.3.3. AJAX.....	18
2.3.3.1. Prednosti AJAX-a.....	18
2.3.3.2. AJAX koristi HTTP zahteve.....	18
2.4. CSS.....	19
2.4.1.1. Sintaksa.....	20
2.4.1.2. Ciljni elementi.....	20
2.4.1.3. Organizacija CSS koda.....	21
3. Model Web aplikacije i UML.....	23
3.1. Model slučajeva korišćenja (eng. Use Case).....	24
3.2. Dijagram aktivnosti.....	25
3.3. Dijagram komunikacije.....	26
3.4. Princip rada aplikacije.....	27
3.4.1. Opšti princip rada.....	27
3.4.2. Prednosti AJAX-a u odnosu na POST/GET metode.....	27
3.4.3. Razlozi uvođenja base64 kodovanja slika.....	27
3.4.4. Dodavanje novih pozadinskih slika.....	27
3.4.5. Dodavanje novih rasporeda elemenata.....	28
4. Primer upotrebe aplikacije.....	29
4.1. Podešavanje osnovnih opcija.....	29
4.1.1. Veličina vizit karte.....	29
4.1.2. Čoskiranje.....	29
4.1.3. Jednostrana ili dvostrana vizit karta.....	30
4.2. Ubacivanje logotipa.....	31
4.2.1. Otvaranje panela za ubacivanje i podešavanje logotipa.....	31
4.2.2. Odabir slike logotipa sa korisnikovog računara.....	31
4.2.3. Učitavanje odabrane slike na server i njeno postavljanje na karticu.....	32
4.2.4. Podešavanje veličine i kompletno uklanjanje logotipa.....	32

4.3. Pozadinska slika i boje.....	33
4.3.1. Otvaranje panela za odabir pozadine.....	33
4.3.2. Ubacivanje Vaše slike za pozadinu.....	33
4.3.3. Učitavanje pozadinske slike na server i njeno postavljanje na karticu.....	34
4.3.4. Odabir neke od ponuđenih pozadinskih slika.....	34
4.3.5. Jednobojna pozadina – uklanjanje slike i odabir boje.....	34
4.4. Rasporedi elemenata (layout-i).....	35
4.4.1. Otvaranje panela za odabir layout-a.....	35
4.4.2. Menjanje profila vizit karte.....	36
4.4.3. Izbor nekog od ponuđenih rasporeda.....	36
4.5. Stilizacija i formatiranje teksta.....	37
4.5.1. Otvaranje panela za stilizaciju i formatiranje teksta.....	37
4.5.2. Unos teksta.....	37
4.5.3. Izmena vrste fonta.....	38
4.5.4. Izmena veličine teksta.....	38
4.5.5. Izmena boje teksta.....	39
4.6. Čuvanje slike.....	39
4.6.1. Otvaranje panela za čuvanje slike.....	39
4.6.2. Slanje slike na E-Mail.....	40
5. Zaključak.....	41
6. Literatura.....	42

1 Uvod

1.1 Vizuelni identitet

Vizuelni identitet je sistem prezentacije kroz koju se postiže prepoznatljivost na tržištu. Svaka firma ima sebi svojstveno predstavljanje, neke tome pridaju više pažnje, a neke manje i to se da primetiti. Proces vizuelizacije počinje izradom osnovnih grafičkih elemenata i dalje se širi na sve oblike tržišne komunikacije. Spisak pravila opisuje knjiga grafičkih standarda i to je temelj na kojem se gradi imidž kompanije.

Ukoliko u početku, iz finansijskih ili drugih razloga nije moguće izraditi kompletnu knjigu grafičkih standarda, neophodno je da se kvalitetno definišu logotip/zaštitni znak, boje, vizitkarta, memorandum, račun i pečat.

Ostali elementi vizuelnog identiteta mogu se definisati postepeno, a preporuka je da se grafički standardi dizajniraju što ranije. U odnosu na druga ulaganja u firmu, ta stavka je zanemariva, a njeni rezultati su ono što firmu predstavlja kod kupaca.

1.1.1 Osnovni principi identifikacije (vizuelne konstante)

Sledeće osnovne likovne elemente možemo izdvojiti kao one koje nalazimo u najvećem broju modela stvaranja lika firme:

- **Naziv firme** – izbor naziva od kritičnog je značaja za firmu, jer naziv ima karakter signuma, kojim se čitavo preduzeće u svim vizuelnim i verbalnim komunikacionim situacijama objašnjava. Osim osnovnih parametra naziva firme, koje on mora zadovoljiti (jasnoća, čitljivost, informativnost, adekvatnost, asocijativnost itd.), naziv mora biti usklađen s elementima marketinga (ekonomska propaganda, razne promocije, odnosi sa javnošću i sl.).
- **Zaštitni znak firme** – takođe mora biti usklađen sa osnovnim intenzijama reprezentativnosti, jasnoće, atraktivnosti, jednostavnosti, mogućnostima aplikacije itd., uključujući njegove varijacije (pozitiv-negativ boje, veličine itd.).
- **Boje firme** moraju biti unapred programirane u svim svojim varijantama intenziteta, tonova i karaktera. Boje mogu imati i funkciju koda, bilo da globalno reprezentuju karakter preduzeća, bilo da postoji koloristički kod označavanja pojedinih pogona, odeli i preduzeća u sastavu.
- **Logotip** – predstavlja stilizovana slova naziva preduzeća ili njegove skraćenice; često zamenjuje zaštitni znak, ali to ne mora biti, nego se sa zaštitnim znakom može upotpunjavati.

- **Lettering** – predstavlja karakterističan tip slovnih znakova (tipografija) koji se u okviru kreiranja lika firme bira iz postojećih tipografskih modela. Pored karakterističnosti (koja od velikih firmi rezultira konstrukcijom i posebnom tipografijom), treba odabrati i takav tipografski repertoar koji se lagano nalazi u različitim reproduktivnim sistemima (mašinski slog, letraset sistem i slično),
- **Sistem preloma** osnovnih dokumenata, sredstava ekonomske propagande, sistem *layouta* – nasuprot prethodnim elementima koji predstavljaju vizuelne kostante lika firme, ovdje se radi o prostornoj organizaciji tih kostanti.

1.2 Ideja projekta

Kao osnovna ideja ovog rada, na osnovu koje je i postavljen projektni zadatak, uzeta je mogućnost izrade vizualnog identiteta korišćenjem Web tehnologija. Web tehnologije ranije nisu nudile takav nivo funkcionalnosti i interoperabilnosti, neophodan za složen proces izrade vizualnog identiteta u formatu pogodnom za štampu. Međutim, ubrzan razvoj ovih tehnologija u poslednjih nekoliko godina ponovo otvara ovo pitanje. Iz tog razloga kao zadatak ovog rada postavljena je izrada Web aplikacije kojom bi se omogućila realizacija izrade vizit karte.

1.3 Desktop i web aplikacije

1.3.1 Tradicionalni pristup izradi

Likovni elementi vizuelnog identiteta se tradicionalno izrađuju uz pomoc desktop „*standalone*” aplikacija. Najpoznatiji programi za tu svrhu dolaze iz Adobe-a i to su Illustrator (za rad sa vektorskom grafikom) i Photoshop (za rad sa rasterskom grafikom). Pored Adobe-a , tu je i nezaobilazni Corel sa svojim paketom CorelDraw – Graphic Suite. Međutim visoka cena ovih programa (CorelDraw GS - oko 500EUR i Adobe Illustrator CS5 - oko 600EUR) često odbija korisnike od istih. Pored cene veliki minus predstavljaju i visoki hardverski zahtevi ovih programa , kao i neophodnost njihove instalacije, čime se uzimaju znatni resursi sa korisnikovog računara.

Ovi programi su najčešće veoma kompleksni i početnicima veoma komplikovani za rad tako da čak ni odlična ideja, moćan računar i skupoceni programi ne garantuju izradu kvalitetnog vizuelnog identiteta.

1.3.2 Obeležja web aplikacija

I desktop i web aplikacije imaju isti zadatak: rešavanje određenog ljudskog problema. Iako se pristup problemu i način na koji ga rešavaju ne razlikuju bitno, okruženje u kome se oba izvršavaju i način interakcije sa korisnikom prave veliku razliku među njima. Oba donose niz prednosti i mana

u odnosu na ovaj drugi, a među najupečatljivije odlike web aplikacija spadaju:

1.3.2.1 Aplikacija je centralizovana (na jednom mestu)

Desktop aplikacije moraju biti instalirane na klijent računaru da bi se uopšte mogle koristiti. Kod web aplikacija to nije slučaj jer se one nalaze na jednom mestu (serveru ili grupi servera). Ovim se uklanjaju mnogi problemi, pre svega problemi sa distribucijom, nadogradnjom aplikacija i generalno u slučajevima kada kod desktop aplikacija korisnik mora da preuzme i instalira/updatuje aplikaciju. Bitna stvar je da kod web aplikacija korisnik nikada ne dobija samu aplikaciju, već samo interfejs ka njoj što je u suštini sve što mu i treba. Ova činjenica znatno olakšava život obema stranama (i autorima i korisnicima web aplikacija).

1.3.2.2 Potreban je samo pretraživač (eng. Browser)

Ovo je možda najbitnije obeležje web aplikacija. Za korišćenje neke od ovih aplikacija potreban i dovoljan je samo Web pretraživač. Ovo povlači niz prednosti:

- Većina ljudi već zna da koristi Web pretraživač.
- Web aplikacija se ponaša kao sajt sa čim je većina već imala priliku da se sretne.
- Klijent računar nema nikakve hardverske ili softverske zahteve osim onih koje mu nameće sam browser.
- Aplikaciji se može pristupiti sa bilo kog mesta na planeti pod uslovom da nije izolovana (napravljena da radi samo na nivou firme – intranet).
- Klijent nema potrebe da instalira bilo kakav dodatni softver ako se sama aplikacija na bilo koji način izmeni (upgrade, ispravka grešaka i slično). Sve izmene postaju vidne i dostupne svim korisnicima onog trenutka kada su implementirane u aplikaciju.

1.3.2.3 Web aplikacije su sigurnije

U zadnjih par godina postali smo svedoci prave navale špijunskih programa, programa koji vrte reklame na vašem računaru ili čak nekih znatno ozbiljnijih štetočina. Jedan od čestih oblika distribuiranja ovih programa je kao dodatak nekim popularnim desktop aplikacijama (programi kao što su KaZaA Media Desktop ili popularni RadLight video player).

Kod web aplikacija ne morate brinuti o tome da li će vam one instalirati neki dodatni (neželjeni) softver ili na bilo koji način ugroziti stabilnost vašeg računara. Web aplikacije su sigurne koliko i browser na vašem računaru pošto on predstavlja sponu između njih i sistema.

1.3.2.4 Web aplikacije su sve popularnije

Možda se ova tvrdnja čini čudna ovako napisana, ali pogledajmo par primera. Ljudi vole da mogu da provere svoj email kada nisu za svojim računarom (ukoliko imaju pristup webmailu), ljudi vole

portale na kojima se informišu i međusobno komuniciraju, ljudi vole prodavnice na kojima mogu u par klikova da kupe ono što ih interesuje bez potrebe da izlaze iz kuće, ljudi vole da vesti dolaze ka njima, a ne da oni jure vesti i tako dalje. Sve ove zahteve uspešno izvršavaju web aplikacije i na ogromnu većinu njih smo se toliko odomaćili da nam je čudno ako negde nedostaju.

Razloga zašto korisnici vole web aplikacije ima više, a jedan od najznačajnijih je svakako to što se lako koriste. Nema gomile padajućih menija, paleta, kontekst menija, specijalizovanih komponenti za unos ovoga i onoga. Tu su linkovi, tu su osnovne kontrole u vidu tekst polja, polja za odabir jedne ili više opcija, dugmad... I to je manje više to. Iako ovako ograničeno okruženje može predstavljati problem developerima ono nosi dobar deo zasluga za popularnost web aplikacija. Pokazalo se da bez obzira kakve sve mogućnosti neka desktop ili web aplikacija pružala većina korisnika će koristiti samo ograničen skup mogućnosti ili aplikaciju uopšte neće koristiti. Iz ovoga se izvlači prilično jasan zaključak: što je aplikacija kompleksnija to će ona više odbijati korisnike. Opšte je poznato da baš jednostavnost (odsustvo kompleksnih alata i mogućnosti) obično određuje u kojoj meri će neka aplikacija biti prihvaćena od strane korisnika.

1.3.3 Prednosti i mane (ukratko)

Desktop aplikacije:

<u>Prednosti</u>	<u>Mane</u>
Veliki broj opcija i mogućnosti	Visoki hardverski zahtevi
Siroka primena	Potrebna instalacija
Privatnost	Komplikovani za rad
	Visoka cena
	Ne kompatibilnost sa drugim platformama
	Problemi oko azuriranja

Web aplikacije:

<u>Prednosti</u>	<u>Mane</u>
Nema instalacije	Mali broj dodatnih opcija
Minimalni hardverski zahtevi	Neophodna brza internet konekcija
Dostupnost	
Veliki auditorijum	
Centralizovano azuriranje	
Jednostavni za korišćenje	
Minimalni troškovi	
Bolja zaštita od virusa	
Kompatibilnost sa skoro svim platformama	

2 Web tehnologije

Web tehnologije možemo podeliti u dve grupe i to na klijentske (eng. *Client side*) i serverske (eng. *Server side*) tehnologije. Za klijentske tehnologije je karakteristično to što se izvršavaju u korisnikovom Web pretraživaču i njihov kod je moguće videti, dok se serverske tehnologije izvršavaju na udaljenom, deljenom računaru (serveru) i za razliku od klijentskih tehnologija moguće je videti samo rezultat izvršenja skripte (programa), ali ne i njen kod.

U ovom poglavlju biće detaljno opisane sve tehnologije koje su korišćene pri izradi aplikacije i to:

- HTML – za definisanje sadržaja Web aplikacije
- PHP – za izradu slike uz pomoć GD biblioteke
- JavaScript i jQuery – za izradu korisničkog interfejsa
- AJAX – za asinhroni prenos podataka (za učitavanje samo dela strane)
- CSS – za definisanje izgleda Web aplikacije

2.1 HTML

HTML (HyperText Markup Language, jezik za označavanje hiperteksta) je jezik za označavanje sadržaja Web stranica. Pomoću njega se jednostavno mogu odvojiti elementi kao što su naslovi, paragrafi, citati i slično. Pored toga, u HTML standard su ugrađeni elementi koji detaljnije opisuju sam dokument - kratak opis dokumenta, ključne reči, podaci o autoru i slično. Ovi podaci su opšte poznati kao *meta* podaci i jasno su odvojeni od sadržaja dokumenta.

Aktuelna verzija standarda je HTML 5, a sam standard održava Konzorcijum za Web (W3C, *World Wide Web Consortium*). HTML je nastao uprošćavanjem SGML (*Standard Generalized Markup Language*, standardizovani uopšteni jezik za označavanje) standarda sa svrhom opisa dokumenta koji se objavljuju na Web-u. U početku su funkcije ovog jezika bile prilično ograničene i uglavnom su se svodile na jednostavno formatiranje teksta i referenciranja dokumenata. Porastom Web-a rasla je i potreba za naprednijim funkcionalnostima, tako da je u tom smeru razvijan i HTML standard. Tada su standardu dodati elementi za opis tabela, slika, slojeva, napredno formatiranje teksta itd.

Osnov HTML-a predstavljaju oznake (eng. *tag*) i njihovi atributi. Pomocu *tagova* se izdvaja određeni deo sadržaja i na njega se primenjuju podrazumevane i eksplicitno navedene karakteristike kao vrednosti atributa. Atributi se nalaze unutar *tagova* i omogućavaju da se pored samog imena *taga* i unapred definisanog ponašanja jos bliže odredi način prikaza i ponašanja označenog dela dokumenta.

Primer:

```
<p>Tekst paragrafa</p>
<p align="right">Tekst paragrafa poravnan sa desne strane</p>
```

HTML dokumenti se sastoje iz dva osnovna dela: dela koji opisuje dokument i dela koji predstavlja sadržaj dokumenta. Informacije koje opisuju sam dokument se smeštaju u zaglavlje dokumenta (prvi deo) koje je definisano putem *head tag*-a. Sadržaj dokumenta nalazi se u drugom delu uokvirenom *body tag*-om. Oba ova elementa se nalaze unutar *html tag*-a.

Primer:

```
<html>
  <head>
    <title>Naslov dokumenta</title>
  </head>
  <body>
    <h1>Naslov 1</h1>
    <p>Ovo je tekst paragrafa</p>
  </body>
</html>
```

2.2 PHP

PHP (Hypertext Preprocessor) je interpretirani programski jezik čija je osnovna namena izrada dinamičkog sadržaja na Web-u.

PHP je stekao popularnost zbog svoje jednostavnosti i sintakse nasleđene iz programskog jezika C. Tokom vremena jezik se proširivao i sticao mogućnosti za objektno orijentisano programiranje, narocito od verzije 5.0. Nalikuje jeziku C++ u smislu da dozvoljava i čisto-proceduralno programiranje ali omogućava i korišćenje klasa i drugih koncepata objektno orijentisanog programiranja (nasledjivanje, apstraktne metode, interfejsi itd.).

2.2.1.1 PHP/FI

PHP predstavlja naslednika alata po nazivu PHP/FI, napisanog 1995. godine od strane Rasmusa Lerdorfa. PHP/FI je predstavljao skup alata napisanih u Perlu, i autor ga je koristio za sopstvene potrebe. Skup alata je dobio ime "alati za ličnu prezentaciju" (eng. *Personal Home Page Tools*), odakle i skraćenica PHP. Kako su rasle potrebe na sajtu, autor je prepisao kompletan projekat u C-u i omogućio da može da komunicira sa bazama podataka a korisnicima svog sajta da naprave sopstvene prezentacije pomocu njega. Rasmus je potom objavio svoje alate i učinio izvorni kod dostupan svima da bi se projekat brže razvijao i da bi se greške brže ispravljale.

PHP/FI, čija je puna definicija glasila "alati za ličnu prezentaciju/prevodilac obrazaca" (eng. *Personal Home Page Tools/Forms Interpreter*) je imao samo neke stvari zajedničke sa današnjim PHP-om - promenljive kao u Perlu, automatsko parsiranje promenljivih iz zahteva i ugrađeni HTML. PHP/FI 2.0 je konačno i zvanično objavljen 1997. godine, da bi ga ubrzo zamenio PHP 3.0.

2.2.1.2 PHP 3.0

PHP 3.0 je predstavljao prvu verziju koja dosta liči na današnje verzije PHP-a. Kreirali su ga izraelski programeri Zeev Suraski i Endi Gutmans sa fakulteta Tehnikon, kao potpunu preradu projekta PHP/FI. Otkrili su da je on imao ozbiljne nedostatke i da nije mogao zadovoljiti njihov univerzitetski projekat elektronske trgovine. U saradnji sa Ledorfom su objavili PHP 3.0 kao zvaničnog naslednika jezika PHP/FI, a njegov razvoj je zaustavljen.

Glavna osobina nove verzije jezika je bila njegova proširivost, koja je privukla brojne programere širom sveta da dodaju nove mogućnosti. Pored kvalitetne podrške za razne vrste baza podataka, verzija 3.0 je uvela i mogućnost objektno orijentisanog programiranja i veću konzistentnost u jezičkoj sintaksi.

Ime je promenjeno u PHP (bez dodatka "FI"), čije se značenje izmenilo da predstavlja rekurzivni akronim "PHP: preprocesor hiperteksta" (PHP: Hypertext Preprocessor).

PHP 3.0 je zvanično objavljen u junu 1998. nakon devet meseci korišćenja u probnoj fazi.

2.2.1.3 PHP 4.0

U zimu 1998. Zeev i Gutmans su započeli ponovno pisanje PHP-a ispočetka, pokušavajući da poprave jezgro tako da bi bolje radilo sa većim aplikacijama i da bi poboljšali modularnost. Novo jezgro su nazvali Zend mashina (eng. *Zend engine*), po delovima svojih imena Zeev i Endi, da bi kasnije formirali i *Zend Technologies* u Ramat Ganu, Izrael, koja nadgleda unapređenje PHP-a. Novo jezgro je uspešno dovršeno i pojavilo se u javnosti sredinom 1999, a nova verzija PHP-a, 4.0, opremljena novim jezgrom je zvanično objavljena u maju, 2000. Pored boljih performansi, nova verzija je uključivala i keširanje standardnog izlaza, sesije, bolju podršku za različite web-servere, sigurniji prenos korisničkih podataka i nekoliko novih kontrolnih struktura.

2.2.1.4 PHP 5.0

Verzija 5.0 je izašla u julu 2004. godine, nakon dugog razvoja i nekoliko probnih verzija. Koristi jezgro Zend mašina 2.0 sa novim objektnim modelom i mnogo novih mogućnosti. Verzija 5 je takođe donela mnoštvo izmena u Objektno Orijentisanom Programiranju.

Iako se PHP može koristiti za programiranje konzolnih aplikacija i grafičkih interfejsa (biblioteka PHP-GTK) njegova osnovna i glavna upotreba je u programiranju dinamičkih stranica na Internetu.

Do 1997. PHP je stekao nekoliko hiljada korisnika; do 1998. broj sajtova na kojima je instaliran PHP 3.0 je porastao na par stotina hiljada a broj korisnika istog na par desetina hiljada. Danas PHP koristi nekoliko stotina hiljada programera i nekoliko miliona sajtova.

Za razliku od većine programskih jezika koji poseduju početnu funkciju (main u C-u, prvi blok BEGIN u Paskalu, klasa koja poseduje main metodu u Javi itd.) , PHP je više nalik na većinu

skriptnih jezika, i PHP datoteka jednostavno sadrži skup instrukcija koje se izvršavaju jedna za drugom, od prve do poslednje gde sledi kraj programa.

U PHP datoteci, blok koji je okružen jezičkim strukturama “<?php” i “?>” se smatra PHP kodom i izvršava se, a ostatak - van tih znakova - se smatra tekstom koji jednostavno treba da se ispiše na standardni izlaz, bez interpretiranja.

PHP kod može biti organizovan u funkcije i klase, i može se organizovati u više datoteka. Kao početna datoteka, tj. datoteka čije instrukcije se izvršavaju prve, se uzima ona datoteka koja se da interpreteru na izvršavanje.

Program koji se napiše u PHP-u ne zahteva prevođenje (kompajliranje), nego se interpretira pri svakom izvršavanju. PHP interpreter može raditi po CGI principu, odnosno tako što će interpreter postojati kao eksterna aplikacija koja se poziva da izvrši datu skriptu svaki put kad bude zahtevana od nekog korisnika, a može biti instaliran i kao modul Web-servisa. Druga varijanta je danas u najvećoj upotrebi jer pruža znatno veću brzinu izvršavanja - interpreter je na taj način uvek učitao u memoriju te se ne mora pozivati spoljašnji program.

2.2.1.5 Uobičajen scenario po kom se izvršavaju PHP skripte je sledeći :

- Klijent (korisnik Interneta koji koristi neki pretraživač) zahteva PHP stranicu sa servera
- Server prosleđuje zahtev servisu za Web (program Web-server na serveru)
- Web-server prepoznaje da se traži PHP datoteka
- Ne šalje njegov sadržaj klijentu, nego ga izvršava kao program pomoću PHP modula
- Izlazni tekst programa (standardni izlaz) se šalje klijentu kao rezultat zahteva
- Klijent prepoznaje vrstu rezultata (HTML kod, slika, PDF sadržaj, arhiva itd.)
- Rezultat se prikazuje klijentu na odgovarajući način

PHP je podržan u većini popularnih operativnih sistema, uključujući UNIX, Linux, Microsoft Windows i Mac OS, može se besplatno skinuti širom Interneta i na zvaničnom sajtu PHP-a, a licenciran je PHP licencom.

Razvojni tim PHP-a se sastoji od nekoliko desetina programera, i još nekoliko desetina radnika koji rade na drugim projektima vezanim za PHP, kao što je PEAR i dokumentacija PHP-a. Pored ovoga, PHP-u su dobrovoljno doprinosili mnogi programeri širom sveta. Brz razvoj je prouzrokovao da PHP poseduje veliki broj biblioteka i funkcija, ali i problem nekonzistentnosti u imenovanju ugrađenih funkcija.

2.3 Javascript, jQuery, AJAX

2.3.1 Javascript

JavaScript je programski jezik koji se interpretira, sa objektno orijentisanim (OO) mogućnostima. Po sintaksi, jezgro jezika JavaScript slično je jezicima C, C++ i Java, s programskim konstrukcijama kao što su naredba `if`, petlja `while` i operator `&&`.

Međutim, ova sličnost se završava sa sintaksom. JavaScript je slabo tipiziran jezik, što znači da se za promenljive ne mora definisati tip. Objekti u JavaScriptu preslikavaju (mapiraju) imena svojstava u proizvoljne vrednosti svojstava. Zbog toga su sličniji heš tabelama ili asocijativnim nizovima (u Perlu) nego strukturama (u C-u) ili objektima (u jezicima C++ ili Java).

Mehanizam OO nasleđivanja u JavaScriptu zasnovan je na prototipovima, kao u malo poznatom jeziku Self. On je potpuno različit od nasleđivanja u jezicima C++ i Java. Kao i Perl, JavaScript je jezik koji se interpretira i nadahnut je Perlom u mnogim oblastima, kakve su regularni izrazi i mogućnosti rada sa nizovima.

Jezgro (engl. *core*) jezika JavaScript podržava brojeve, znakovne nizove (engl. *Strings*) i logičke vrednosti kao osnovne tipove podataka. Osim toga, ono ima ugrađenu podršku za objekte kao što su nizovi (engl. *arrays*), datumi i regularni izrazi.

JavaScript je programski jezik koji se izvršava na strani klijenta. On se najčešće koristi u pretraživačima (engl. *Web browsers*), pa se jezgro opšte namene proširuje objektima koji omogućavaju skriptovima interakciju s korisnikom, upravljanje pretraživačem i izmenu sadržaja dokumenta koji se pojavljuje unutar prozora pretraživača. Ova ugrađena (engl. *embedded*) verzija JavaScript-a izvršava skripte koji su pridruženi HTMLdokumentima.

Jezgro jezika JavaScript i njegovi ugrađeni tipovi, predmet su međunarodnih standarda, i veoma je dobra kompatibilnost različitih implementacija. Neki delovi klijentskog JavaScript-a zvanično su standardizovani, neki su *de facto* standardi, a ostali predstavljaju proširenja koja zavise od pretraživača. Kompatibilnost između različitih pretraživača značajna je za programere koji koriste klijentski JavaScript.

Jedna od najčešćih zabluda o JavaScriptu jeste da je to pojednostavljena verzija Jave, programskog jezika kompanije Sun Microsystems. Osim delimične sličnosti u sintaksi i činjenice da i Java i JavaScript daju sadržaj koji može da se izvršava u pretraživačima, između ova dva jezika ne postoji nikakva veza. Sličnost naziva je čisto marketinški potez kompanija Netscape i Sun (ovaj jezik je prvobitno nazvan LiveScript i ime mu je u poslednjem trenutku promenjeno u JavaScript). Međutim, JavaScript može da skriptuje Javu.

Posto se JavaScript interpretira umesto da se prevodi, često se smatra da je to jezik za skriptovanje,

a ne pravi programski jezik. Time se nagoveštava da su jezici za skriptovanje jednostavniji i da su to programski jezici za početnike. Činjenica da je JavaScript manje strog po pitanju tipova podataka čini ga nešto pristupačnijim neiskusnim programerima. Osim toga, mnogi Web dizajneri mogu da koriste JavaScript za brzo rešavanje manjih programerskih zadataka.

Međutim, ispod svoje tanke glazure jednostavnosti, JavaScript je programski jezik sa potpunim skupom mogućnosti, složen kao i bilo koji drugi, a od nekih i složeniji. Programeri koji pokušaju da iskoriste JavaScript za rešavanje netrivialnih zadataka često otkrivaju da je taj postupak težak, ukoliko dobro ne razumeju ovaj jezik.

2.3.2 *Jquery*

Sa ubrzanim razvojem web-a porasla su i prosečna očekivanja stepena funkcionalnosti web sajtova. JavaScript kao jedan od veoma laganih, ali ujedno i glavnih jezika web-a pruža pregršt mogućnosti za developere. Međutim potpuna moć JavaScript-a se otkriva korišćenjem njegovih biblioteka. Jednu od najboljih JavaScript biblioteka predstavlja jQuery.

Sa jQuery-jem možete uz pomoć samo jedne linije koda da postignete isto što biste postigli sa 10-20 linija regularnog JavaScript koda. jQuery je i sam pisan u JavaScript-u i nalazi se u obliku jednog .js fajla koji linkujete sa svoje web stranice. Nakon toga Vaš JavaScript kod pristupa biblioteci pozivajući razne jQuery funkcije.

jQuery biblioteka dolazi u 2 oblika:

- nekompresovani .js fajl- prost za čitanje i modifikaciju, ali ima oko 160kb veličine.
- smanjeni .js fajl - svi komentari, bele površine i drugi nepotrebni karakteri su uklonjeni iz fajla, čime je celokupna biblioteka smanjena na 20-ak kb. Iako je teška za čitanje, ovo je verzija koja treba da bude postavljena na sajt radi što bržeg učitavanja strane.

jQuery je besplatan za skidanje i upotrebu. Osim što Vam je dostupna jQuery biblioteka, dostupno je i na stotine besplatnih jQuery dodataka (plugin-ova). **Za šta sve možete da koristite jQuery?** Pomoću njega možete da pišete moćne aplikacije u JavaScripti i kreirate animirane efekte koji prijaju oku, a koji pariraju Flash animacijama. Između ostalog, jQuery je dobar i za:

- Dodavanje efekata tekstu (efekti bleđenja teksta, nestajanja, ulaska/izlaska teksta)
- Pravljanje xml (Ajax) upita (za traženje dodatnih informacija od web servera bez potrebe za osvežavanjem stranice)
- Manipulacija DOM-om (jednostavno dodavanje, uklanjanje i premeštanje sadržaja web stranice pomoću par redova koda)
- Pravljenje slajd šoua

- Pravljenje menija sa podmenijem
- Kreiranje drag & drop interfejsa (tako da na stranici elementi mogu da se premeštaju jednostavnim prevlačenjem)
- Pravljanje moćnijih formulara

jQuery je kompatibilan sa brojnim pretraživačima (IE, Firefox, Safari) i predstavlja trenutno najpopularniju JavaScript biblioteku, koju koristi oko 30% najpopularnijih svetskih sajtova.

2.3.3 AJAX

AJAX predstavlja skraćenicu od *Asynchronous JavaScript And XML*. Kratko rečeno AJAX je zasnovan na JavaScript i HTTP zahtevima i njegova glavna osobina je da omogućiti da se na nekom sajtu šalje/prima sadržaj bez ponovnog učitavanja stranice.

AJAX nije novi jezik programiranja, već tehnika za kreiranje boljih, bržih i interaktivnijih Web aplikacija. Sa AJAX-om, vaš JavaScript može da komunicira direktno sa serverom, koristeći JavaScript XMLHttpRequest objekat. Pomoću ovog objekta, vaš JavaScript može da razmenjuje podatke sa Web serverom, bez ponovnog učitavanja stranice.

AJAX koristi asinhroni transfer podataka (HTTP zahtevi) između pretraživača i servera, omogućujući web aplikaciji da obrađuje samo one podatke koji su zaista potrebni, umesto čitavih stranica.

AJAX je zasnovan na sledećim web standardima:

- JavaScript
- XML
- HTML
- CSS

Web standardi koji se koriste u AJAX-u su dobro definisani i podržani od svih velikih pretraživača.

2.3.3.1 Prednosti AJAX-a

Web aplikacije imaju mnogo prednosti nad desktop aplikacijama, osnovna je da one lako mogu da dopru do više korisnika, lakše su za instaliranje i obezbeđivanje podrške, i lakše za razvijanje. Sa AJAX-om, internet aplikacije mogu da budu bogatije i brže za korišćenje nego što su bile do sada.

2.3.3.2 AJAX koristi HTTP zahteve

U "klasičnom" programiranju, ako da bi dobili bilo kakvu informaciju iz baze podataka ili nekog fajla koji se nalazi na serveru, treba napraviti HTML formu i preko GET ili POST metode dobiti ili

poslati podatke serveru. Što bi značilo da nakon što se pošalje/dobije informacija od servera, učita opet cela strana kako bi rezultati bili vidljivi i samim tim, pošto server vraća novu stranicu svaki put kada korisnik pošalje neke podatke, tradicionalne web aplikacije znaju da budu spore.

Pomocu AJAX-a, JavaScript komunicira direktno sa serverom, kroz JavaScript XMLHttpRequest objekat. Pomoću HTTP zahteva, Web stranica šalje zahtev serveru, i dobije odgovor od njega - bez ponovnog učitavanja web stranice, i korisnik gotovo da i neće primetiti da je skript poslao zahtev za stranicom, ili da je u pozadini poslao podatke na server.

2.4 CSS

CSS (engl. *Cascading Style Sheets*) je jezik formatiranja pomoću kog se definiše izgled elemenata Web-stranice.

Prvobitno, HTML je služio da definiše kompletan izgled, strukturu i sadržaj Web-stranice, ali je od verzije 4.0 HTML-a uveden CSS koji bi definisao konkretan izgled, dok je HTML ostao u funkciji definisanja strukture i sadržaja.

Kako je HTML postajao komplikovaniji, davao je sve više mogućnosti za definiciju izgleda elemenata, ali je istovremeno postajao nečitljiviji i teži za održavanje. Različiti pretraživači su prikazivali dokumente na različite načine, i postojala je potreba za doslednom tehnikom definisanja prikaza elemenata na stranici.

Da bi se ovo postiglo, devet različitih metoda je predloženo na zvaničnom forumu W3C-a. Od devet, dve metode su izabrane kao temelj onoga što je kasnije postalo CSS: CHSS (engl. *Cascading HTML Style Sheets*) i SSP (engl. *Stream-based Style Sheet Proposal*). Prvo je Hakon Vium Li (koji je sada šef tehničke službe kompanije Opera) predložio CHSS u oktobru 1994, jezik koji je imao dosta sličnosti sa današnjim CSS-om. Bert Bos je radio na pretraživaču po nazivu Argo, koji je imao sopstveni način definisanja stilova, SSP. Li i Bos su radili zajedno da bi osnovali CSS standard (slovo H je izbačeno iz skracenice CHSS jer se CSS mogao odnositi i na druge jezike pored HTML-a).

Za razliku od postojećih jezika poput DSSSL-a i FOSI-a, CSS je dozvoljavao da više opisa utiče na dokument, tj. jedna definicija stilova je mogla naslediti osobine od druge.

Lijev predlog je postavljen na konferenciji "Web mozaik" u Čikagu 1994. godine, i ponovo sa Bosovim predlogom 1995. Otprilike u ovo vreme je osnovan W3C, koji je preuzeo funkciju razvoja CSS-a. Do kraja 1996, CSS je bio spreman da se objavi kao standard, i CSS1 je objavljen u decembru.

Razvoj HTML-a, CSS-a i DOM-a se odvijao u jednoj istoj grupi, *HTML Editorial Review Board* (ERB). Početkom 1997. grupa ERB se podelila na tri radne grupe: radna grupa za HTML, kojom je

upravljao Den Konoli iz W3C-a, radna grupa za DOM, kojom je upravljao Loren Vud iz kompanije Softkvod, i radna grupa za CSS, kojom je upravljao Kris Lili iz W3C-a.

Radna grupa za CSS je počela da radi na problemima koji nisu bili obuhvaćeni CSS-om verzije 1, koji se tako razvio u CSS 2, 4. novembra 1997. Objavljen je kao zvanična verzija 12. maja 1998. CSS3, čiji je razvoj započet 1998. se još uvek razvija.

2.4.1.1 Sintaksa

CSS sintaksa se sastoji od opisa izgleda elemenata u dokumentu. Opis može da definiše izgled više elemenata, i više opisa može da definiše jedan element. Na taj način se opisi slažu jedan preko drugog da bi definisali konačni izgled određenog elementa (otuda naziv *Cascading* (engl. *cascade* - crep) da bi se dočaralo slaganje jednog stila preko drugog u definisanju konačnog izgleda elementa).

Svaki opis se sastoji od tri elementa:

- definicija ciljnih elemenata
- svojstva
- vrednosti

Nakon što definišemo ciljne elemente, tj. elemente na koje će se trenutni opis odnositi, nizom parova svojstvo-vrednost definišemo izgled svakog ciljnog elementa.

Primer:

```
Ciljni element {  
    svojstvo : vrednost;  
    svojstvo : vrednost;  
    ...  
}
```

CSS podržava i komentare, koji izgledaju kao u programskom jeziku C, dakle navodeći se između znakova `/*` i `*/`.

2.4.1.2 Ciljni elementi

Ciljni elementi se definišu na tri načina:

- navodeći HTML *tag* ciljnih elemenata
- navodeći klasu elemenata
- navodeći direktnu identifikacionu vrednost (ID) elementa

Kada definišemo preko **HTML tag-a**, to znači da će opis uticati na sve elemente u dokumentu koji imaju ovaj tag. Definicija ciljnih elemenata se tada vrši preko direktnog upisa odgovarajućeg HTML *taga*:

```
body {  
    svojstvo : vrednost;  
    svojstvo : vrednost;  
    ...  
}
```

Klasa HTML elementa je reč koju stavljamo kao vrednost argumenta **CLASS** pri definiciji tog elementa. Definicija ciljnih elemenata (svih koji imaju određenu istu klasu) se vrši tako što upišemo znak tachke (.) a zatim naziv klase:

```
.mojaKlasa {  
    svojstvo : vrednost;  
    svojstvo : vrednost;  
    ...  
}
```

Ovaj opis će, dakle, imati uticaja na sve elemente u dokumentu koji su definisani na sledeći način:

```
<p class="mojaKlasa">...</p>  
<div class="mojaKlasa">...</div>
```

Identifikaciona vrednost elementa je vrednost argumenta ID pri definiciji tog elementa. U jednom dokumentu identifikacione vrednosti moraju biti jedinstvene, tj. može postojati samo jedan element sa određenom identifikacionom vrednošću, što znači da ovakvi opisi mogu uticati samo na po jedan element u dokumentu. Definišu se pomocu znaka tarabe (#) , a zatim se navodi identifikaciona vrednost.

```
#mojID {  
    svojstvo : vrednost;  
    svojstvo : vrednost;  
    ...  
}
```

Ovaj će, dakle, opis uticati na sve elemente koji u svojoj HTML definiciji imaju id="mojID" .

2.4.1.3 Organizacija CSS koda

CSS kod se može zadavati na tri standardna mesta:

- direktno u tagu, koristeći argument *style*
- u zaglavlju dokumenta unutar *tag-a style*
- u eksternoj datoteci, koja se u dokument uključuje tagom link

Za jako male dokumente, argument *style tag-a* može biti prihvatljiv, ali za veće dokumente ne može jer se na ovaj stil definiše za svaki element ponaosob. Definicija stilova se najčešće radi grupišući elemente u klase, pomocu argumenta *class tag-a*, a zatim definišući izgled tih klasa bilo u okviru elementa *style* u zaglavlju ili u eksternoj datoteci. Argument *style tag-a* se najčešće koristi ako neki element ne pripada nijednoj grupi i ima posebne zahteve u odnosu na sve ostale elemente.

Stilovi u eksternoj datoteci imaju jednu posebnu prednost u odnosu na preostala dva načina, a to je da na taj način možemo da utičemo na elemente više dokumenata, praktično na čitav jedan sajt. Dovoljno je da pisac bude dosledan u definisanju klasa elemenata, da bi jedan fajl sa stilovima

određivao izgled svih stranica. Na ovaj način jedna promena u fajlu sa stilovima utiče na sve stranice te su stilovi centralizovani.

3 Model Web aplikacije i UML

Model je uprošćavanje realnosti koja omogućava kompletan opis sistema iz određene perspektive. Modeli se prave radi boljeg razumevanja kompleksnih sistema, koje su veoma teški za razumevanje u celini. Pomažu u vizuelizaciji sistema kakav trenutno jeste ili kakav treba da bude, dozvoljavaju definisanje strukture ili ponašanja sistema, daju šablone u konstrukciji sistema i dokumentuju donete odluke.

Najbolji modeli su povezani sa stvarnošću . Svi modeli pojednostavljaju realnost (treba biti siguran da to pojednostavljenje ne sakriva značajne detalje). Ne postoji model ili pogled koji je dovoljan. Da bi opisali strukturu i ponašanje sistema koji posmatramo, treba prikazati čitav skup modela i pogleda na sistem iz različitih perspektiva.

Unified Modeling Language (**UML**) je vizuelni jezik za modelovanje koji se koristi za definisanje, vizuelizaciju, konstrukciju i dokumentovanje softverskih sistema. Omogućava donošenje odluka i bolje razumevanje sistema koji se konstruiše. Koristi se za analizu, dizajn, pretragu, konfigurisanje, održavanje i kontrolu informacija o takvim sistemima. UML obuhvata informacije o statičkoj strukturi i dinamičkom ponašanju sistema. Sistem se modeluje kao kolekcija diskretnih objekata koji su u interakciji.

Statičke strukture definišu vrste objekata važne za sistem i njihovu implementaciju, isto kao i veze između objekata . Dinamičko ponašanje definiše istoriju objekata u vremenu i komunikaciju između objekata da bi se postigli određeni ciljevi.

Modelovanje sistema sa različitih, ali povezanih pogleda omogućava da sistem bude razumljiv za različite namene.

UML nije u osnovi programski jezik. Neki alati omogućavaju generisanje koda iz UML-a u različitim programskih jezicima, isto kao i reverzno inženjerstvo modela iz postojećih programa u UML dijagrame. UML je diskretan jezik za modelovanje koji nije namenjen za modelovanje kontinualnih sistema koji se nalaze u inženjeringu i fizici . Namena UML-a je da bude univerzalni jezik modelovanja za diskretne sisteme – softverske sisteme, organizacijske sisteme ili npr. sisteme koji sadrže logiku.

UML omogućava konstruisanje šema koje modeluju sistem opisujući:

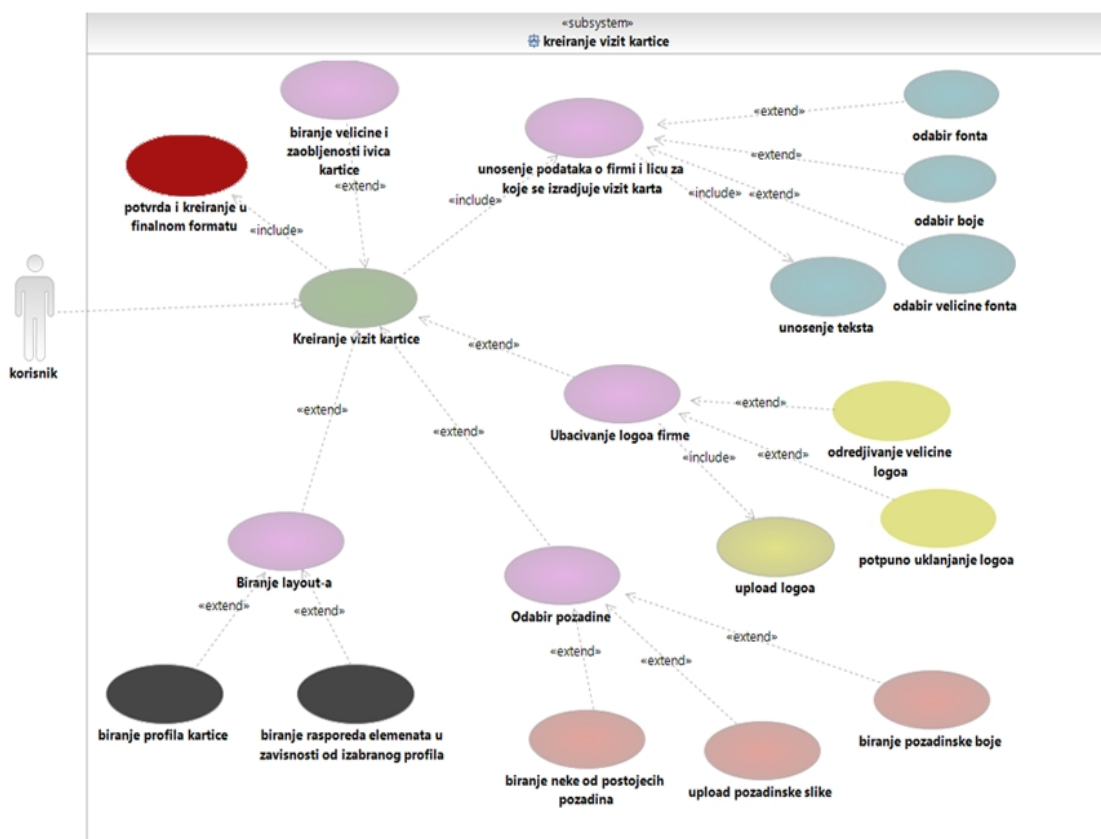
- konceptualne stvari (npr. proces poslovanja i funkcije sistema)
- konkretne stvari (npr. tipove klase, šeme baza podataka , softverske komponente).

3.1 Model slučajeva korišćenja (eng. Use Case)

Modelovanje slučajeva korišćenja u zvanični standard UML-a uveo je Ivar Jacobson ranih '90-ih godina na osnovu iskustva ranije ideje scenarija (eng. Scenario) koji u principu još uvek postoji u UML-u. Od tada su slučajevi korišćenja veoma bitan deo UML-a i u OO zajednici im se pridaje mnogo pažnje.

Predstavljaju tehniku za modelovanje funkcionalnosti i specifikaciju zahteva sistema, odnosno opis funkcionalnosti iz ugla korisnika sistema i opis interakcije između korisnika i sistema.

U njemu se izostavljaju detalji vezani za realizaciju posmatrane funkcionalnosti. Prikazuju se dijagramima slučaja korišćenja (DSK).



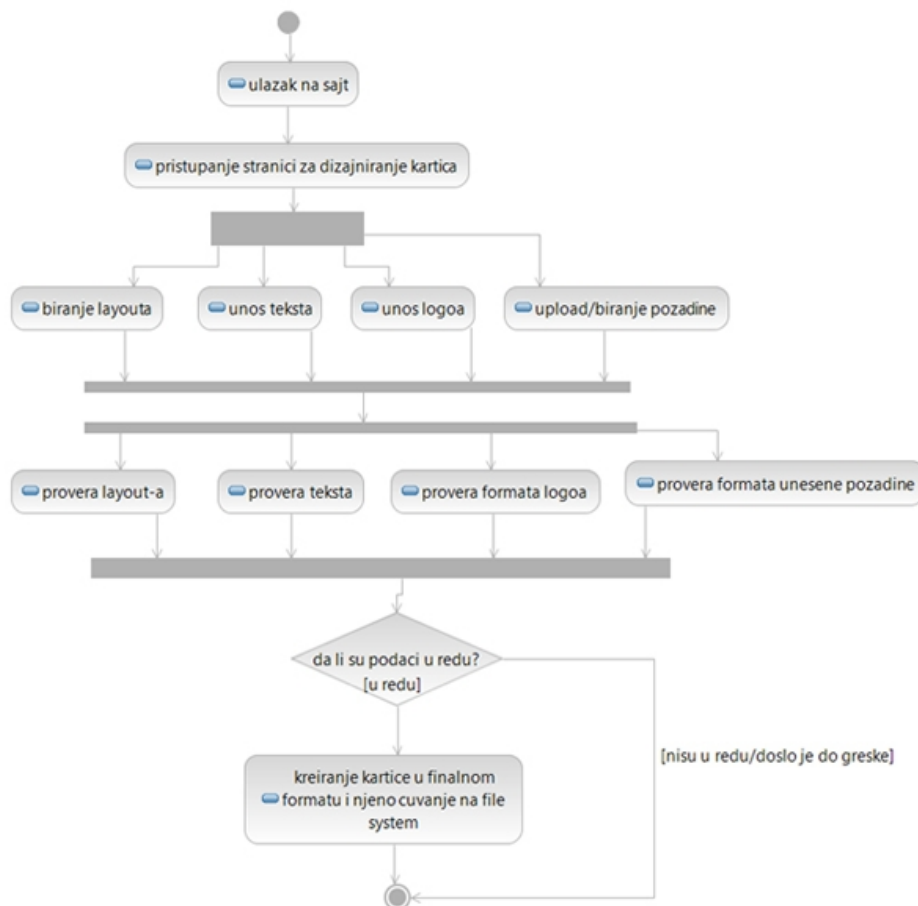
Slika 3.1. use-case dijagram

Elementi DSK:

- Slučajevi korišćenja : elipsa
- Učesnici : stilizovan ljudski lik
- Veze : odgovarajuće linije
- Granice sistema : pravougaonik

3.2 Dijagram aktivnosti

Dijagrami aktivnosti su namenjeni modelovanju dinamičkih aspekata (ponašanja) sistema. Prikazuju sekvencijalne i konkurentne korake u procesu obrade. Mogu se koristiti za opis dinamike skupa objekata i toka neke operacije. Dve najznačajnije primene dijagrama aktivnosti su: modelovanje tokova poslova (tada se kreira dijagram iz ugla posmatranja aktera koji sarađuje sa sistemom) i modelovanja operacija (ovde se više vodi računa o detaljima proračuna).



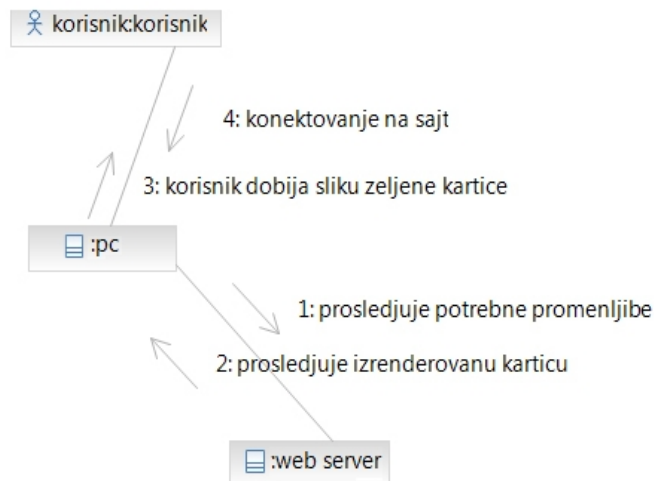
Slika 3.2. dijagram aktivnosti

Aktivnost reprezentuje neatomsku obradu koja se dekomponuje na jedinice – elementarne podaktivnosti – pojedine akcije.

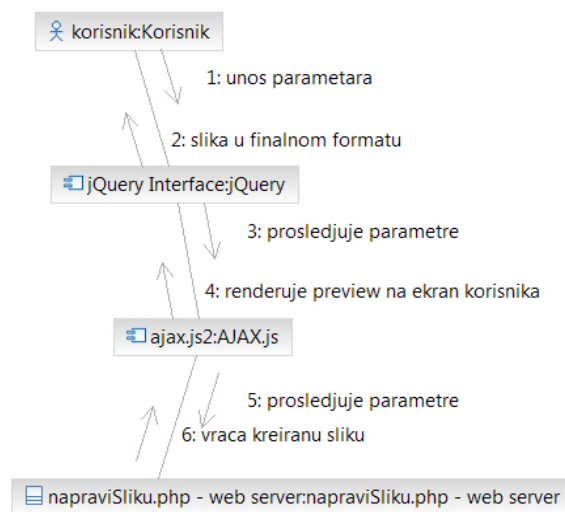
Akcija je osnovna jedinica specifikacije ponašanja koja reprezentuje neku transakciju ili obradu u modelovanom sistemu. Akcija je osnovni izvršni element aktivnosti i predstavlja jedan korak u aktivnostima koji se obično dalje ne dekomponuje. Aktivnost definiše ponašanje koje se može ponavljati na više mesta, dok se akcija dešava samo jednom na posebnom mestu unutar aktivnosti.

3.3 Dijagram komunikacije

Predstavlja dijagram interakcije i koristi se da modeluje ponašanje slučaja upotrebe. Za razliku od sekvencijalnih dijagrama, koji takodje spadaju u dijagrame interakcije, dijagram komunikacije je više fokusiran na prikazivanje kolaboracije između objekata nego na vremenske sekvence.



Slika 3.3.1 dijagram komunikacije



Slika 3.3.2 dijagram komunikacije

Osobine dijagrama komunikacije:

- Naglašavaju povezanost različitih učesnika u interakciji
- Omogućavaju proizvoljan raspored učesnika i prikazuju kako su učesnici povezani pomoću linija
- Redosled se označava numerisanjem poruka
- Osim veza asocijacija, na dijagramu se prikazuju i privremene veze u kontekstu interakcije.

3.4 Principi rada aplikacije

3.4.1 Opšti princip rada

Korisnik pristupa aplikaciji (sajtu) putem interneta i preko interfejsa izradjenog u jQuery-ju podešava parametre potrebne za izradu slike.

Sa svakom izmenom vrednosti parametara poziva se AJAX funkcija koja prikuplja sve promenljive potrebne za izradu slike i preko *XMLHttpRequest* objekta ih šalje PHP fajlu zaduženom za pravljenje slike.

Potom PHP prihvata poslate parametre, instancira zadatu klasu i poziva njen metod za izradu slike. Taj metod vraća sliku u .png formatu , kodovanu u base64 zapis, koja je ujedno i izlaz iz PHP fajla (prosledjuje se natrag AJAX-u).

AJAX f-ja potom prihvata novu sliku i postavlja je na odgovarajuće mesto na stranici , pri čemu se zamenjuje stara slika i korisnik dobija uvid u izmene koje je napravio.

3.4.2 Prednosti AJAX-a u odnosu na POST/GET metode

Kao sto je već detaljno objašnjeno u poglavlju 2.3.3 , koje obrađuje AJAX , ova tehnologija je korišćena da bi se omogućilo slanje i primanje sadržaja bez ponovnog učitavanja stranice.

Tako je postignuto da se renderovanjem nove slike menja samo sadržaj *div*-a u kome se nalazi slika , a ne dolazi do učitavanja kompletne stranice, čime se znatno poboljšavaju performanse i ukupni doživljaj korisnika.

3.4.3 Razlozi uvođenja base64 kodovanja slika

Base64 je vrsta zapisa, koja binarni prikaz konveruje u ASCII. Bez upotrebe base64 zapisa, sve privremene (eng. *temporary*) slike, koje bivaju kreirane pri svakoj izmeni parametara bi morale biti čuvane na serveru, a potom povremeno brisane, kako ne bi popunile sav fizički prostor na serveru.

Međutim korišćenjem base64 zapisa izbegnuto je čuvanje slike , tako da PHP f-ja za pravljenje slike vraća samo kod slike u tekstualnom formatu, a ne smešta sliku na server čime ga rasterećuje u velikoj meri.

3.4.4 Dodavanje novih pozadinskih slika

Dodavanje novih pozadina vrši se kopiranjem istih na server i to u sledeće foldere.

Prednja strana:

- uspravni profil – images/backgrounds/front/x
- horizontalni profil – images/backgrounds/front/y

Zadnja strana:

- uspravni profil – images/backgrounds/back/x
- horizontalni profil – images/backgrounds/back/y

Pored slika u punoj veličini potrebno je prekopirati na server i umanjene prikaze istih u podfolder *thumbs* (primer : images/backgrounds/back/x/thumbs). Nakon toga umanjeni prikazi će biti automatski renderovani na stranu.

Napomena: ime umanjenog prikaza mora imati isti naziv , kao i slika u punoj veličini, s tim što mora imati prefix tn_ (pozadina – tn_pozadina).

3.4.5 Dodavanje novih rasporeda elemenata

Kako su predefinisani rasporedi ugrađeni na nivou klase, njihovo dodavanje, za razliku od dodavanja novih pozadina znatno je komplikovanije i zahteva određeno programersko iskustvo. Osim modifikacije klase potrebno je prekopirati sliku umanjenog prikaza *layout*-a na server i to u sledeće foldere:

Prednja strana:

- uspravni profil – images/layouts/front/x
- horizontalni profil – images/layouts/front/y

Zadnja strana:

- uspravni profil – images/layouts/back/x
- horizontalni profil – images/layouts/back/y

Posto se *layout*-i razlikuju kako za prednju i zadnju stranu, tako i za oba profila osnove , izmenom profila , poziva se AJAX f-ja koja učitava specifične prikaze i omogućava njihovo biranje .

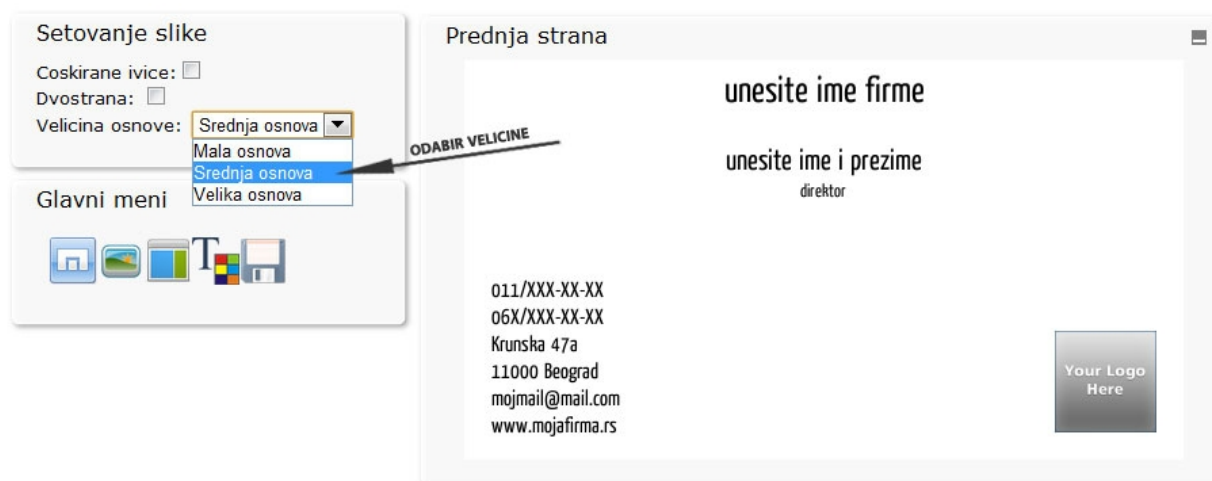
4 Primer upotrebe aplikacije

4.1 Podešavanje osnovnih opcija

4.1.1 Veličina vizit karte

Promena veličine vizit karte vrši se preko padajućeg menija obeleženog na slici. Ponuđene su tri standardne veličine:

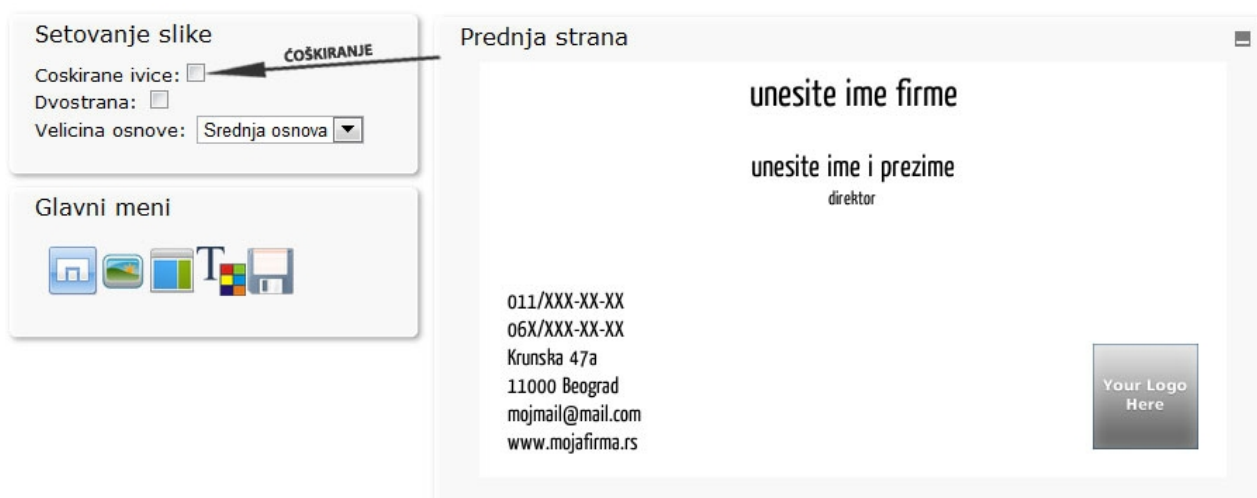
- Mala – 8,5 cm x 5,5 cm
- Srednja – 9 cm x 5 cm
- Velika – 9 cm x 5.5 cm



slika 4.1.1 odabir veličine slike

4.1.2 Čoškiranje

Ukoliko želite da ivice vaše vizit karte budu čoškirane štiklirajte obeleženi *checkbox*.



Slika 4.1.2. coskiranje

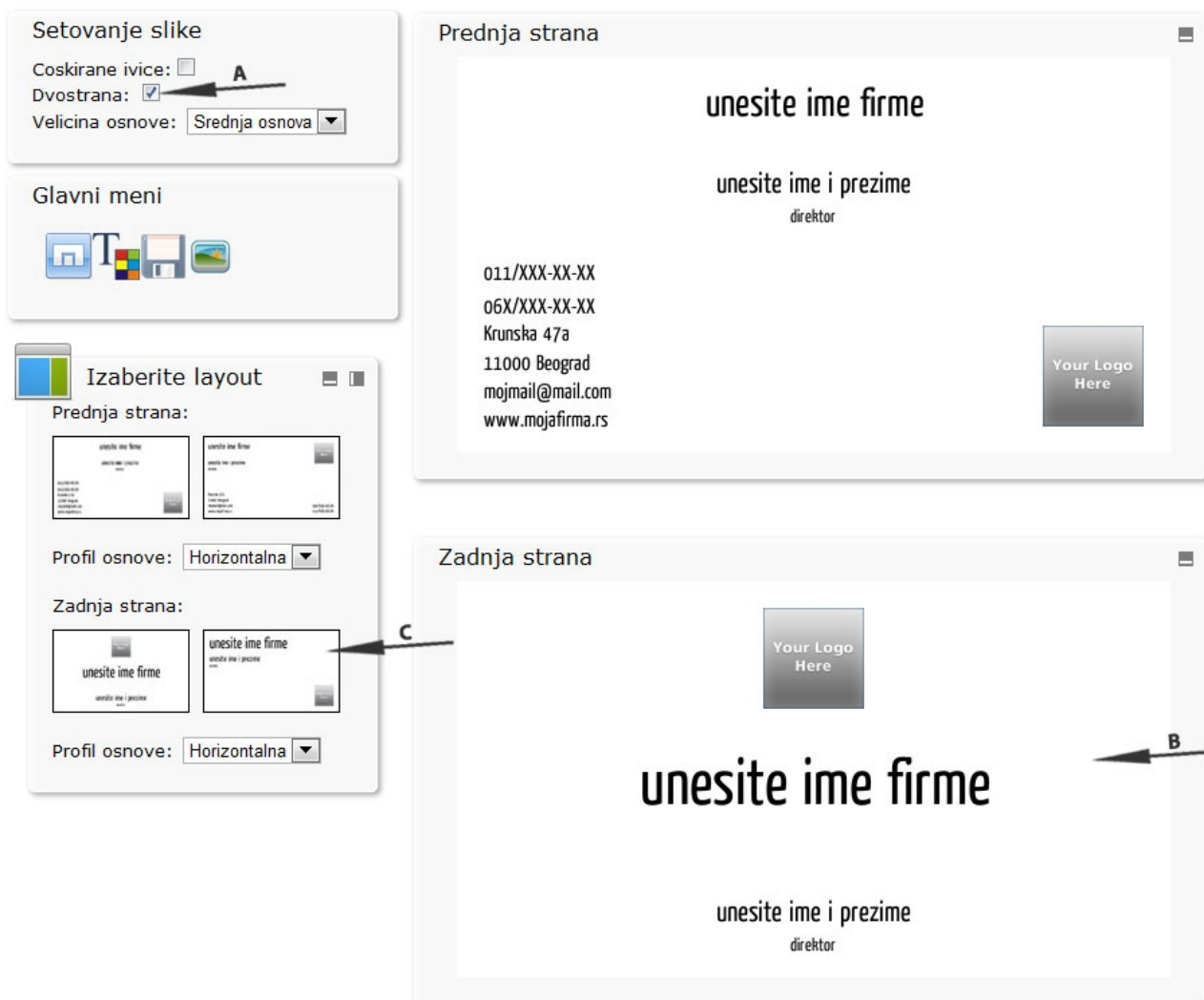
Napomena : Čoškiranje se vrši na štamparskoj mašini, tako da je samo prikaz u dizajneru čoškiran, a ne i finalna slika, koja će vam biti poslata na željenu e-mail adresu.

4.1.3 Jednostrana ili dvostrana vizit karta

Ukoliko želite dvostranu vizit kartu potrebno je štiklirati obeleženi checkbox (A).

Kada ste to uradili pojaviće se novi prozor sa prikazom zadnje strane (B) .

Pored toga dupliraće se i sve opcije u dizajneru , tako da će biti omogućena sva podešavanja za zadnju stranu - na primeru (C) su vidljivi *layout*-i za zadnju stranu kojih ne bi bilo da ova opcija nije uključena .



Slika 4.1.3. jednostrana / dvostrana vizit karta

Naravno moguće je i vraćanje na jednostranu vizit kartu i to samo odštikliranjem checkbox-a, pri čemu će nestati kako prikaz zadnje strane, tako i sve opcije vezane za njeno podešavanje.

4.2 Ubacivanje logotipa

4.2.1 Otvaranje panela za ubacivanje i podešavanje logotipa

Kako bi pristupili opcijama za podešavanje i ubacivanje logotipa potrebno je otvoriti „logo panel“, klikom na dugme (ikonu) obeleženo na slici.

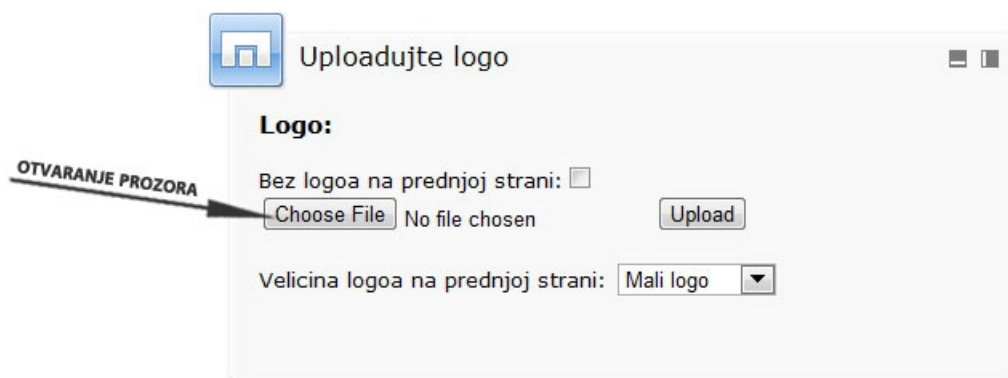


Slika 4.2.1 otvaranje panela za ubacivanje logo-a

4.2.2 Odabir slike logotipa sa korisnikovog računara

Kako bi na vizit kartu ubacili logotip, potrebno je izabrati željenu sliku sa fajl sistema. Prozor za odabir slike se otvara klikom na dugme *Choose File* (oznaceno na slici). Ovaj prozor predstavlja standardni windows-ov *open* panel u kome je potrebno otići na lokaciju na kojoj se nalazi slika, a potom selektovati istu i kliknuti na dugme *open*.

Slika mora biti u nekom od podržanih formata - .png, .jpg ili .jpeg.

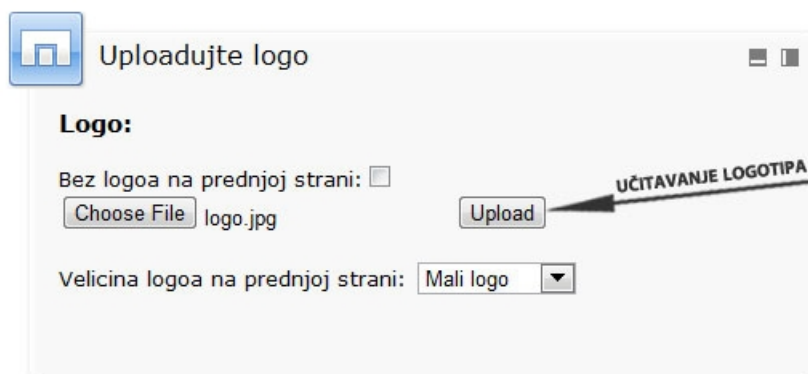


Slika 4.2.2 panel za ubacivanje logotipa

Napomena : Ukoliko je logotip nepravilnog oblika predlaže se korišćenje .png formata kako bi se sačuvala transparentost.

4.2.3 Učitavanje odabrane slike na server i njeno postavljanje na karticu

Nakon odabira slike sa fajl sistema, potrebno je kliknuti na dugme *upload* (oznaceno na slici) kako bi je poslali na server i postavili na vizit kartu.



Slika 4.2.3 postavljanje odabrane slike logotipa

Napomena: Potrebno je dva puta kliknuti na dugme *upload* kako bi postavili logotip, jednom da bi snimili sliku logotipa na server i drugi put da bi ga postavili na umanjeni prikaz vizit karte.

4.2.4 Podešavanje veličine i kompletno uklanjanje logotipa

Nakon odabira i učitavanja slike na server, ostalo je jos samo odrediti veličinu logotipa odabirom neke od tri ponuđene veličine (mali, srednji, i veliki).



Slika 4.2.4 odabir velicine i uklanjanje logotip-a

Kako bi kompletno uklonili logotip, potrebno je uključiti *checkbox* obelezen na slici.

Napomena : Ukoliko želite da logotip zadrži transparentnost kako ne bi bio pravougaonog oblika ubacite sliku logotipa u .png formatu.

4.3 Pozadinska slika i boje

4.3.1 Otvaranje panela za odabir pozadine

Da bi pristupili opcijama za podešavanje pozadine vizit karte, potrebno je otvoriti određeni panel klikom na dugme (ikonu) obeleženo na slici.



Slika 4.3.1. otvaranje panela za odabir pozadinske slike

4.3.2 Ubacivanje Vaše slike za pozadinu

Kako bi ubacili Vašu pozadinsku sliku potrebno je izabrati željenu sliku sa fajl sistema. Prozor za odabir slike sa fajl sistema otvara se pritiskom na dugme *Choose File* (obeleženo na slici).



Slika 4.3.2. odabir pozadinske slike sa fajl sistema

Napomena : Slika mora biti u nekom od podržanih formata - .png , .jpg ili .jpeg.

4.3.3 Učitavanje pozadinske slike na server i njeno postavljanje na karticu

Nakon odabira slike sa fajl sistema , potrebno je kliknuti na dugme *upload* (oznaceno na slici) kako bi poslali istu na server i postavili je kao pozadinu kartice.

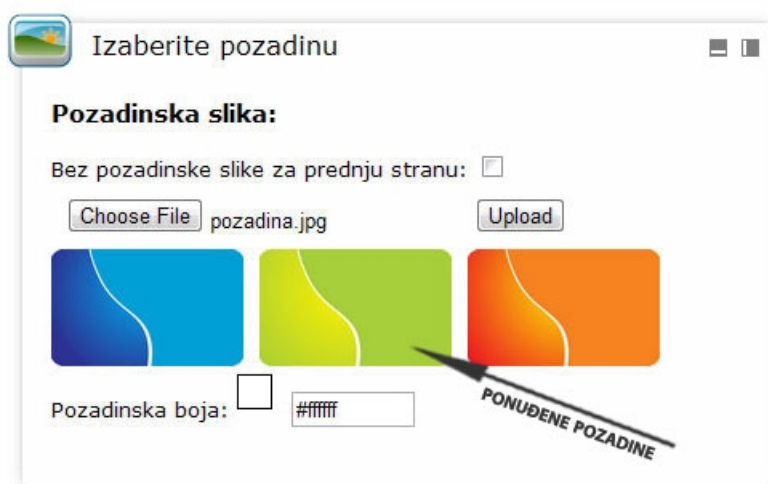


Slika 4.3.3. postavljanje odabrane pozadine

Napomena: Potrebno je dva puta kliknuti na dugme *upload* kako bi postavili pozadinu, jednom da bi snimili sliku na server ,a drugi put da bi je dodali na sliku.

4.3.4 Odabir neke od ponuđenih pozadinskih slika

Da bi izabrali neku od ponuđenih slika ,potrebno je kliknuti na umanjeni prikaz iste i ona će biti automatski postavljena kao pozadina vizit karte.

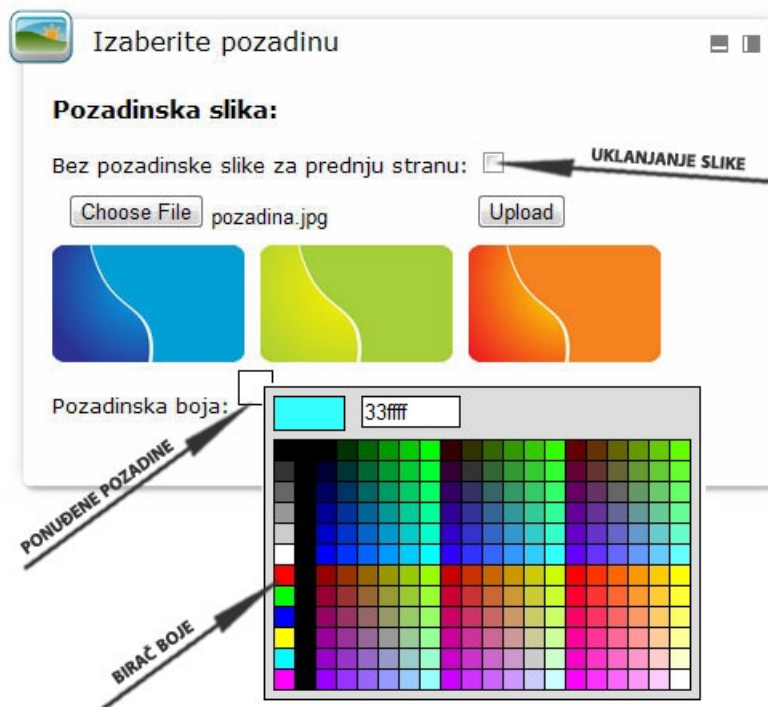


Slika 4.3.4. odabir neke od ponudjenih pozadina

4.3.5 Jednobojna pozadina – uklanjanje slike i odabir boje

Ukoliko želite jednoboju pozadinu vizit karte , a vec ste ubacili Vašu sliku ili ste odabrali neku od ponudjenih, potrebno je štiklirati *checkbox* obelezen na slici.

Nakon toga potrebno je otvoriti birač boje klikom na obeleženi kvadratić sa prikazom trenutne pozadinske boje, a zatim iz birača izabrati željenu boju.



Slika 4.3.5. birač boje

Napomena : Ukoliko u biraču ne postoji određena nijansa boje , moguće je i uneti kod boje u heksadecimalnom zapisu (#RRGGBB) , gde su RR, GG i BB heksadecimalne vrednosti crvene (eng. *Red*) , zelene (eng. *Green*) i plave (eng. *Blue*).

4.4 Rasporedi elemenata (layout-i)

4.4.1 Otvaranje panela za odabir layout-a

Da bi pristupili opcijama za podešavanje *layout-a* potrebno je otvoriti određeni panel klikom na dugme obeleženo na slici.

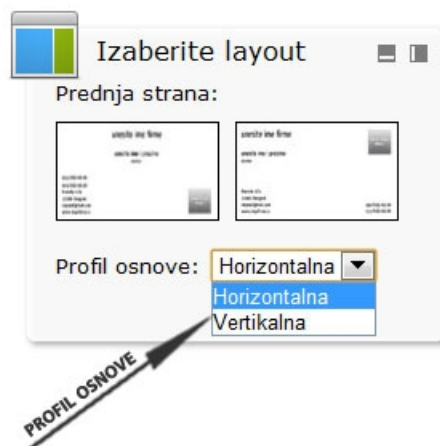


Slika 4.4.1. otvaranje layout panela

4.4.2 Menjanje profila vizit karte

Menjanje profila vizit karte vrši se preko padajuće liste obeležene na slici.

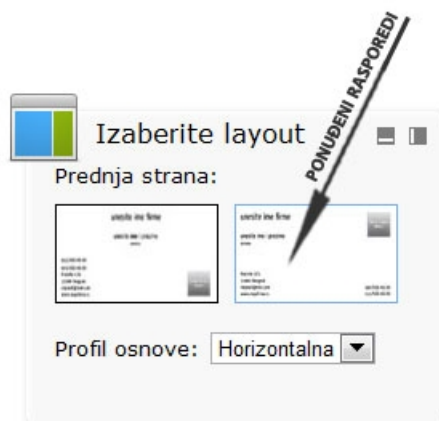
Ukoliko želite da vaša vizit karta stoji uspravno izaberite opciju *Vertikalna*, u suprotnom izaberite opciju *Horizontalna*.



slika 4.2. menjanje profila vizit karte

4.4.3 Izbor nekog od ponuđenih rasporeda

Kako bi promenili raspored elemenata na vizit karti potrebno je izabrati neki od ponuđenih izgleda, klikom na njegov umanjeni prikaz (obeleden na slici).



slika 4.3. odabir rasporeda elementata

Napomena: Uspravna i vertikalna vizit karta imaju različite predefinisane rasporede.

4.5 Stilizacija i formatiranje teksta

4.5.1 Otvaranje panela za stilizaciju i formatiranje teksta



Slika 4.5.1. otvaranje panela za stilizaciju teksta

Kako bi pristupili opcijama za stilizaciju i formatiranje teksta potrebno je otvoriti određeni panel klikom na dugme obeleženo na slici.

4.5.2 Unos teksta



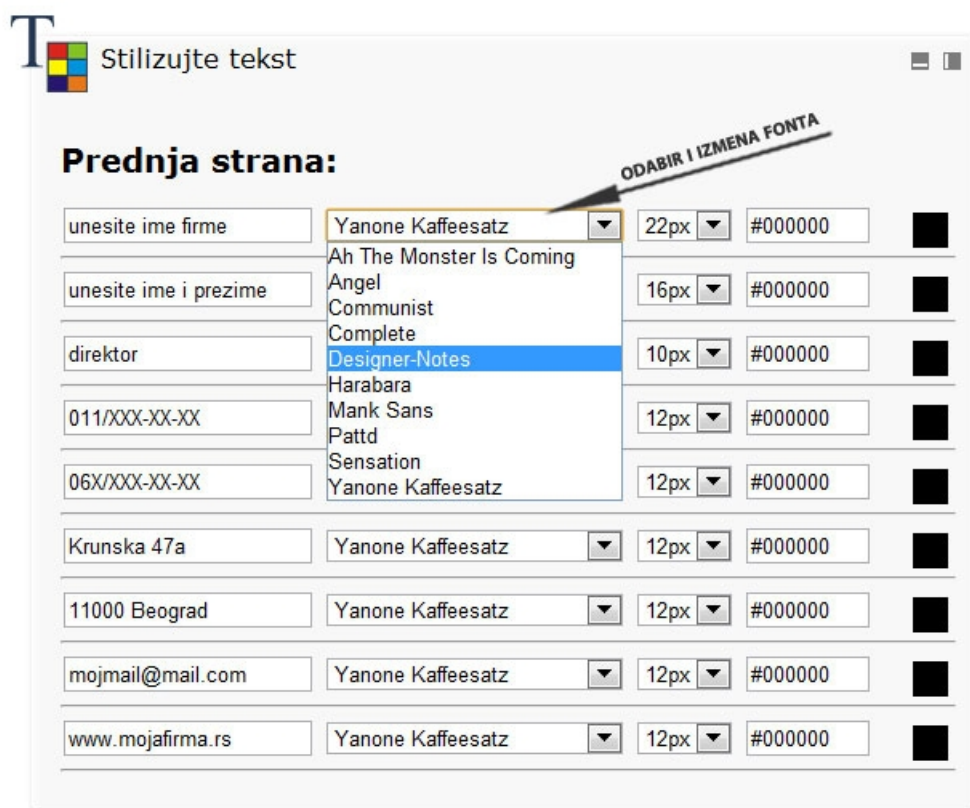
Slika 4.5.2. unos teksta

Unos i izmena teksta vrši se menjanjem vrednosti polja označenih na slici.

Napomena: Da bi izmene postale vidljive na slici potrebno je skloniti fokus sa polja čija se vrednost menja.

4.5.3 Izmena vrste fonta

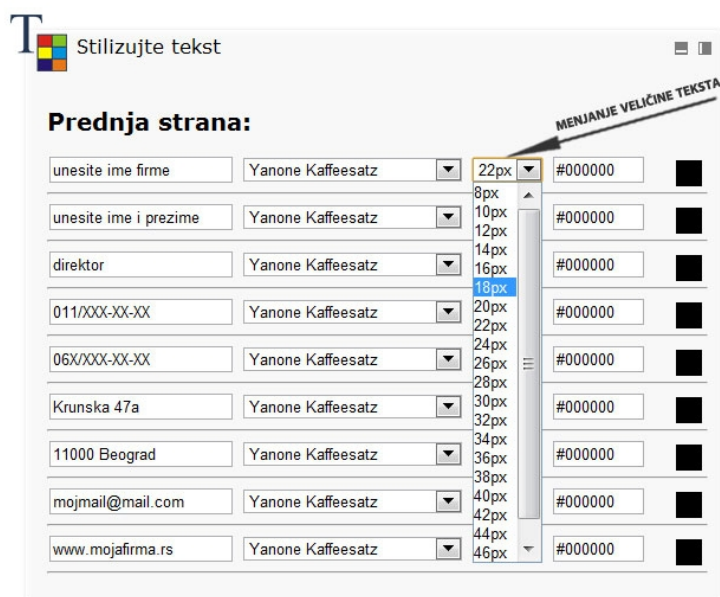
Kako bi promenili vrstu fonta potrebno je otvoriti padajući meni (obeležen na slici) klikom na strelicu pored njega, a potom iz liste odabrati neki od ponuđenih fontova.



Slika 4.5.3. izmena fonta

4.5.4 Izmena veličine teksta

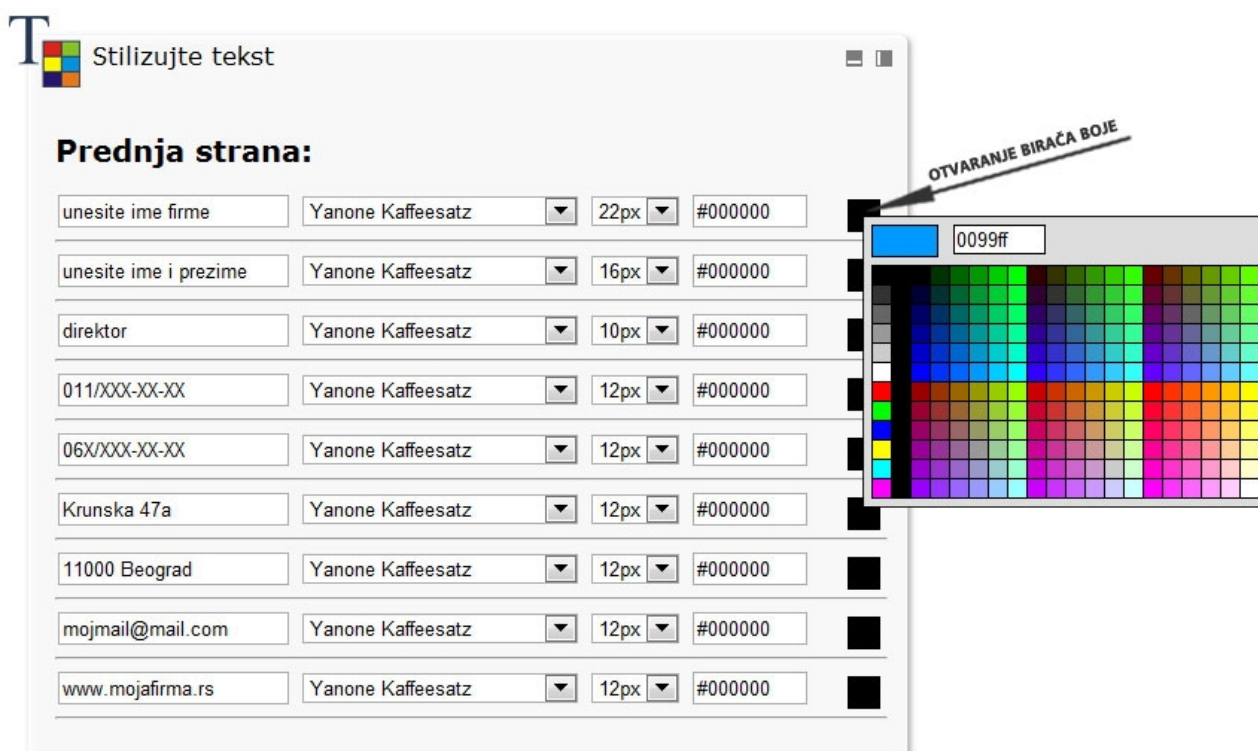
Kako bi promenili veličinu fonta potrebno je otvoriti padajući meni klikom na strelicu pored njega (obeležen na slici), a potom iz liste odabrati neku od ponuđenih veličina. Svi *combo box*-ovi sa veličinama se odnose samo na određene delove teksta i izražene su u pikselima (px.) .



Slika 4.5.4. promena velicine teksta

4.5.5 Izmena boje teksta

Kako bi promenili boju teksta potrebno je otvoriti birač boje klikom na kvadratić sa prikazom trenutne boje (vidi sliku 4.5.5 *promena boje teksta*), a zatim odabrati neku od ponuđenih boja.



Slika 4.5.5. promena boje teksta

4.6 Čuvanje slike

4.6.1 Otvaranje panela za čuvanje slike

Kako bi sačuvali sliku prvo je potrebno otvoriti panel za čuvanje (slanje) slike.

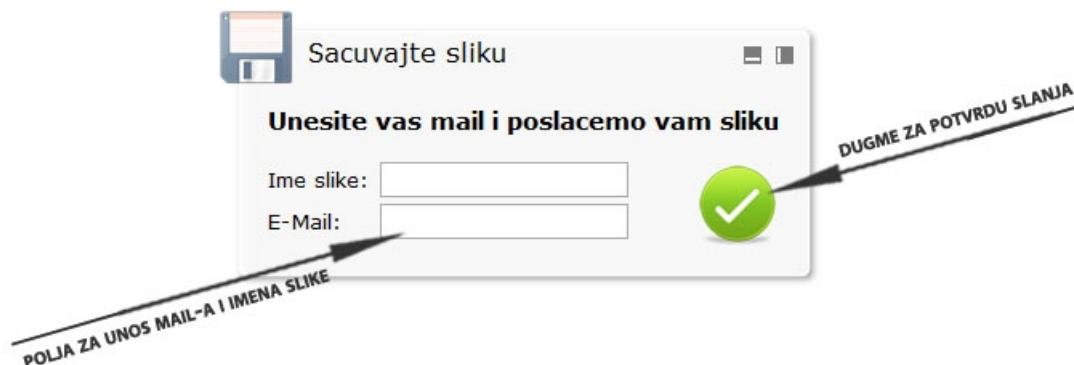


Slika 4.6.1. otvaranje panela za cuvanje slike

Ovaj panel se otvara klikom na dugme (ikonu) obeleženo na slici.

4.6.2 Slanje slike na E-Mail

Nakon što ste otvorili ovaj panel, potrebno je uneti ime slike i Vašu E-mail adresu u polja označena na slici, a zatim klikom na dugme potvrditi slanje slike. Potom će finalna slika spremna za štampu, u rezoluciji od 300dpi biti automatski poslata kao dodatak (eng. *attachment*) poruke.



slika 6.2. slanje slike na mail

Napomena: Ukoliko u roku od nekoliko minuta niste dobili poruku, a sigurni ste da ste dobro uneli e-mail adresu , proverite sanduče sa *spam*-om.

5 Zaključak

Ubrzani razvoj web tehnologija otvorio je mogućnost razvoja najrazličitijih web aplikacija. Zbog svih prednosti ovih aplikacija, do kraja tekuće decenije se očekuje potpuni prelazak na računarstvo u oblacima (eng. *Cloud-computing*). Ono predstavlja lokacijski nezavisno računarstvo, gde serveri obezbeđuju resurse, programe i podatke računarima i drugim uređajima na zahtev korisnika. Ovakav pristup računarstvu donosi mnogo prednosti, kao što su savršena administracija i maksimalna bezbednost podataka, mogućnost pristupanja „svom računaru” sa bilo koje tačke u svetu i td. , međutim povlači sa sobom i problem privatnosti, kao i neophodnost veze sa serverom.

Iako trenutno skromnih mogućnosti, naredne verzije ove aplikacije treba da omoguće kreiranje kako vizit karti, tako i drugih likovnih elemenata vizelnog identiteta, kao što su memorandumi, brošure, koverte i dr. Pored toga planira se i unapređenje funkcionalnosti i to ubacivanjem većeg broja predefinisanih rasporeda elemenata (eng. *Layout*), kao i uvođenjem mogućnosti kreiranja novih rasporeda od strane korisnika pomeranjem elemenata na željene pozicije. Pored ovoga planira se i uvođenje tema, kako bi se postigla što veća sličnost svih likovnih elemenata vizuelnog identiteta (memoranduma, brošura i td.). Ove teme bi trebalo da sadrže određene pozadine, kao i predefinisane rasporede koji se savršeno uklapaju u iste.

Pošto ova aplikacija treba prvenstveno da bude namenjena štamparijama neophodno je i uvođenje sistema naplate i poručivanja. Kako ova usluga neće biti besplatna, potrebno je i iscrtavanje žiga (eng. *Watermark*) preko prikaza u dizajneru, kao i onemogućavanje čuvanja slike, kako bi se sprečila zloupotreba aplikacije. Takođe planira se i uvođenje sistem registracije i prijavljivanja, kako bi korisnici na svom nalogu uvek imali sačuvane one elemente vizuelnog identiteta koje su ranije dizajnirali i kako bi mogli ponovo da ih poruče. Za neiskusnije korisnike planira se i dodavanje čarobnjaka (eng. *Wizard*). On predstavlja element korisničkog interfejsa, koji korisnika u vidu dobro definisanih koraka vodi kroz složeni proces izrade nekog od likovnih elementata vizuelnog identiteta, čime ga dosta olakšava i ubrzava.

Pored navedenih unapređenja, nova verzija će doneti i kompletan redizajn sajta (aplikacije) i to u čistom, minimalističkom stilu, sa glavnim akcentom na funkcionalnosti, brzini i što boljoj preglednosti.

6 Literatura

- [1] Suzana Marković, **Rečnik UML-a** , materijali za učenje na daljinu, LINK Group.
- [2] Wikipedia – Online enciklopedija, dostupna na <http://www.wikipedia.org/> .
- [3] Brendovi.ba - servis dizajnera, marketinških kuća, institucija i vlasnika domaćih brendova, dostupan na <http://brendovi.ba/> .
- [4] Ilija Studen, **Web aplikacije – juče, danas, sutra**, tutorial.
- [5] NAPRAVISAJT.COM – prezentacija firme koja se bavi izradom web sajtova, vizuelnog identiteta, E-marketingom i štampom, dostupna na <http://www.napravisajt.com/> .
- [6] Web Arena – blog namenjen web dizajnerima i web programerima, dostupan na <http://webarena.rs/> .