Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Nejc Lepen

Avtomatski sistem za serviranje izdelkov znotraj oglasnega prostora

diplomsko delo

univerzitetni študijski program računalništvo in informatika

Mentor: prof. dr. Saša Divjak

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Predstavite problematiko prenosov in serviranja izdelkov v oglasnem prostoru. Razvijte avtomatizirani sistem, ki iz spletne trgovine črpa izdelke preko posebej prirejenega vtičnika. Podatke naj shrani v nadzorni sistem in jih pripravi v obliko za tvorbo oglasov. Uporabnik nadzornega sistema naj ima možnost izbire med izdelki spletne trgovine in tvorbe oglasnih vzorcev, ki jih lahko namesti na želeno spletno stran in jih kasneje administrira preko nadzornega sistema. Predvidite tudi statistično vodenje prikazov in klikov v nadzornem sistemu, kar lahko uporabljamo za optimizacijo spletnih kampanj.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Nejc Lepen, z vpisno številko **63060152**, sem avtor diplomskega dela z naslovom:

Avtomatski sistem za serviranje izdelkov znotraj oglasnega prostora

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Saša Divjaka,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 24. junij 2014 Podpis avtorja:

Zahvaljujem se podjetju iPROM za tehnično in strokovno pomoč pri izvedbi diplomske naloge, družini za poterpežljivost in podporo ter prijateljem za druženje tekom študija.

Diplomsko delo posvečam družini, ki me je potrpežljivo podpirala skozi vsa študijska leta.

Kazalo

Seznam uporabljenih kratic

AJAX (*angl. Asynchronous JavaScript and XML*) Skupina tehnologij za asinhrono obdelavo podatkov.

ASCII (*angl. American Standard Code for Information Interchange*) Ameriški standardni nabor za prestavitev informacij.

Base64 (*angl. Binary data scheme in ASCII string format*) Predstavitev binarnih podatkov v ASCII zapisu.

CDN (*angl. Content Delivery Network*) Omrežje za dostavo vsebine.

CGI (*angl. Common Gateway Interface*) Skupni prehodni vmesnik.

CGP Celostna Grafična Podoba.

CSS (*angl. Cascading Style Sheets*) Stilna predloga za izgled spletne strani.

HTML (*angl. HyperText Markup Language*) Označevalnik jezik za izdelavo spletnih strani.

IAB (*angl. Interactive Advertising Bureau*) Urad za interaktivno oglaševanje.

JS (*angl. JavaScript*) Objektni skriptni jezik za izdelavo interaktivnih sple-

tnih strani.

JSON (*angl. JavaScript Object Notation*) Objektno definiran format podatkov.

MVC (*angl. Model View Controller*) Vzorec za razvoj aplikacij.

MySQL (*angl. My Structured Query Language*) Sistem za upravljanje s podatkovnimi bazami, ki uporablja SQL.

OO (*angl. Object-oriented*) Objektno orientiran pristop.

PHP (*angl. Hypertext Preprocessor*) Je programski jezik za procesiranje spletnih vsebin.

RPK Razmerje med prikazi in kliki.

SQL (*angl. Structured Query Language*) Strukturirani povpraševalni jezik za delo s podatkovnimi bazami.

W3C (*angl. Structured Query Language*) Mednarodna skupnost, ki skrbi za razvoj standardov.

Povzetek

Namen diplomskega dela je predstaviti problematiko prenosa in serviranja izdelkov v oglasnem prostoru. S to težavo se dnevno srečujejo spletni trgovci ter načrtovalci in zakupniki oglasnega prostora. Cilj diplomskega dela je reševanje problema prenosa in serviranja izdelkov z moderno spletno aplikacijo. Zasnovali smo avtomatiziran sistem, ki je sestavljen iz treh ključnih komponent: spletne trgovine, nadzornega sistema in spletne strani z oglasnim prostorom. V sklopu diplomskega dela smo razvili omenjeni avtomatizirani sistem, ki iz spletne trgovine črpa izdelke preko posebej prirejenega vtičnika. Dobljene podatke shranimo in prikažemo v nadzornem sistemu, kjer so pripravljeni v obliki za generiranje oglasnih vzorcev. Uporabnik nadzornega sistema lahko izbira med izdelki spletne trgovine in ustvarja vzorce. Ustvarjene oglasne vzorce lahko namesti na želeno spletno stran in jih naknadno administrira preko nadzornega sistema. Dodana vrednost k diplomskemu delu je statistično vodenje oziroma nadzorovanje prikazov in klikov, ki jih lahko uporabnik sistema uporablja za optimizacijo spletnih kampanj.

Ključne besede: spletna aplikacija, nadzorni sistem, oglasi, oglasni vzorec, avtomatizacija, generiranje kode.

Abstract

The purpose of the thesis is to present the problems of deploying and serving products into advertising space, encountered daily by online marketers, planners and leaseholders of advertising spaces. The aim of the thesis is to solve the problem in question with the help of a novel web application. Therefore, we have designed an automatic system, which consists of three key components: an online store, a surveillance system and websites accommodating advertising space. In the course of this thesis, we have developed the aforementioned automatic system that employs the extraction of the products from an online store through a customized plugin. The acquired data are saved and displayed in the control system, where they are prepared in the form devised to generate advertising samples. The control system user can select among the products presented at the online store and design samples. The prepared samples can then be placed within a desired website, and subsequently administered via the control system. The added value of this thesis is represented by the ability to conduct statistical management, i.e. monitoring banner displays and actual clicks, which the user can employ to achieve a better optimization of an online campaign.

Keywords: web application, control system, ads, advertising sample, automatization, code generation.

Poglavje 1

Uvod

V zadnjih letih smo priča vse večjemu razvoju interneta in posledično tudi vse večjemu razvoju spletnih storitev. Ljudje dnevno uporabljajo internet za iskanje informacij, branje novic ali za preprosto virtualno druženje v družbenih omrežjih. Računalniki prinašajo velik delež k razvoju interneta, a mobilne naprave so naredile pravo revolucijo na tem področju. Zadnji statistični podatki kažejo, da ima v Sloveniji kar 76 odstotkov gospodinjstev dostop do interneta. V prvem četrtletju 2013 se je zanimanje za mobilne širokopasovne internetne povezave povečalo za 5 odstotnih točk v primerjavi z letom prej[5]. Pametni mobilniki pa so postali nekakšen »trend« med mlajšimi uporabniki spleta.

Vse večja uporaba interneta je vzbudila zanimanje tudi pri oglaševalcih. Tako so na primer oglaševalci v ZDA v lanskem letu namenili več sredstev digitalnemu oglaševanju kot televizijskemu[21]. V Sloveniji se vložki v digitalno oglaševanje iz leta v leto sicer tudi povečujejo, a še precej zaostajamo za razvitimi zahodnoevropskimi in ameriškim trgom.

Rečemo lahko, da se »trend oglaševanja« spreminja. Oglaševalci čedalje bolj iščejo točno določene skupine ljudi, ki jih smatrajo za potencialne kupce svojih izdelkov ali storitev. Čedalje več denarja se vlaga v razvoj algoritmov

in spletnih storitev, ki oglaševalcem pomagajo pri iskanju ciljnih skupin in optimizaciji sredstev, ki so namenjena oglaševanju.

Vse hitrejši razvoj interneta pa je omogočil tudi povečevanje spletnega nakupovanja. Primerjava statističnih podatkov[5, 2] zadnjih štirih let kaže, da se ≫trend≪ spletnega nakupovanja v Sloveniji povečuje. Posledica tega je povečanje števila spletnih trgovin in nasičenosti spletnega trga z izdelki.

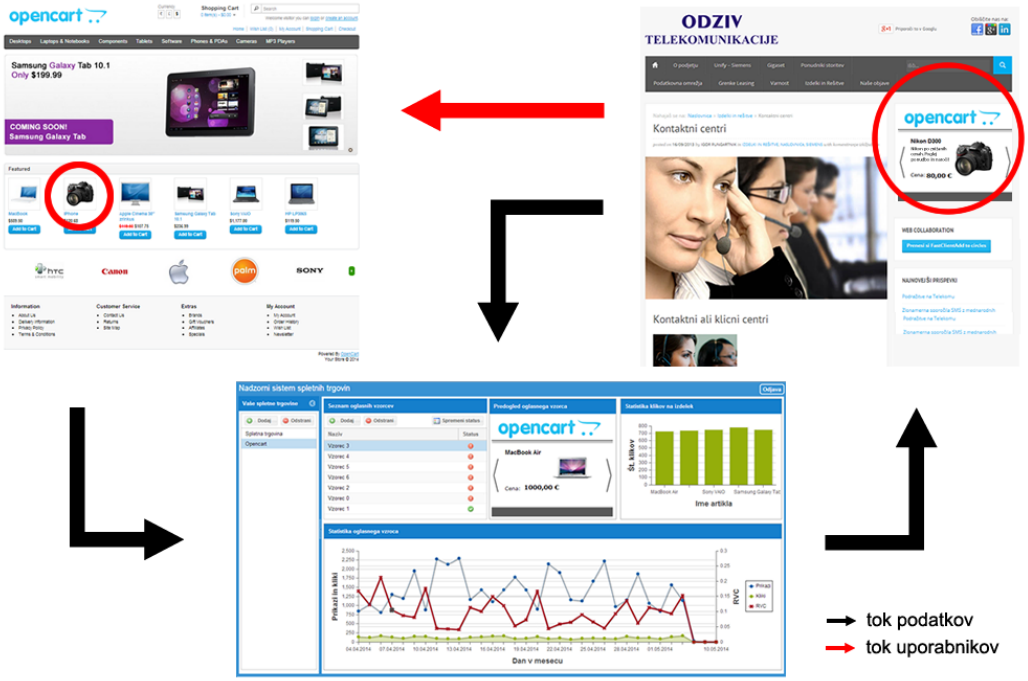
Izziv, pred katerega so danes postavljeni lastniki spletnih trgovin je, kako na čim bolj privlačen način predstaviti svoje izdelke potrošnikom. Eden od načinov je spletno oglaševanje, ki lahko za seboj povleče veliko dela. Problem je lahko že pri velikih količinah izdelkov, vnosu teh v različne servirne sisteme ter nadzoru in kontroli potencialno zanimivih izdelkov. Vse to je rešljivo, a za to je potrebno veliko znanja, usposobljenih programerjev, nadzornikov oglasnega prostora in oblikovalcev. To lahko lastnika spletne trgovine stane precej denarja. V vse to pa še ni vključena investicija, ki jo mora oglaševalec plačati za zakup oglasnega prostora.

Težav pa s tem še ni konec. Naslednja težava se lahko pojavi med samim potekom oglaševalske kampanje. Nekateri oglasi oziroma izdelki imajo boljši odziv od drugih, torej lahko za enako število prikazov dobimo več potencialnih kupcev. Za tovrstno optimizacijo pa spet potrebujemo nadzornika oglaševanja, programerje, dogovore z mediji, odstranjevanje oglasov in ponovno izdelavo novih (izboljšanih) oglasov. Na tej točki se torej pojavi potreba po nadzornem sistemu in avtomatiziranem prenosu izdelkov v spletno oglasno mesto.

Namen diplomskega dela je načrtovati in izdelati avtomatiziran sistem za prikaz izdelkov spletne trgovine v spletnemu medijskemu prostoru. Uporabnik, ki bi uporabljal takšen nadzorni sistem, ne bi le optimiziral prikazov in klikov in znotraj spletnega oglasa (ki je namenjen enemu izdelku), ampak bi

obšel tudi zgoraj omenjen problem klasičnega oglaševanja in ga nadomestil z avtomatiziranim prenosom.

Avtomatiziran sistem je nadzorna aplikacija, ki iz spletne trgovine pridobi izdelke in jih pripravi za ustvarjanje oglasnih vzorcev. Za delovanje sistema je potrebno imeti spletno trgovino z vtičnikom za prenos izdelkov ter dostop do želene spletne strani za integracijo oglasnih vzorcev. Avtomatiziran sistem je sestavljen iz treh ključnih delov: spletne trgovine, nadzornega sistema in spletne strani z oglasnim prostorom. V nadaljevanju tega poglavja bi vas radi seznanili s temi tremi ključnimi deli, ki so bistvenega pomena za razumevanje avtomatiziranega sistema.



Slika 1.1: Avtomatiziran sistem

Slika 1.1 prikazuje osnovne tokove podatkov med spletno trgovino, nadzornim sistemom in spletno stranjo. Prikazana dva različna tokova. Prvi tok (označen je s črno puščico) je tok podatkov, drugi tok (označen z rdečo

puščico) je tok uporabnikov. Tok podatkov teče iz spletne trgovine proti nadzornemu sistemu in iz nadzornega sistema preko implementirane kode oglasa na želen medij oz. spletno stran z oglasnim prostorom in nazaj v nadzorni sistem. Rdeča puščica prikazuje klik uporabnika na spletni strani in prihod v spletno trgovino na izbrani izdelek.

1.1 Spletna trgovina

Spletna trgovina je aplikacija, ki potrošnikom preko spleta nudi nakup izdelkov. Njene glavne prednosti pred običajno trgovino so večja dostopnost in boljša preglednost nad cenami izdelkov. Vsak posamezen izdelek ima naslov, opis, sliko, ključne besede ter definirano kategorijo.

Običajen potek nakupa izdelka se prične z izbiro izdelka, ki ga uporabnik vnese v t. i. spletno košarico. Uporabnik spletni nakup zaključi s plačilom izbranega izdelka in z izbranim plačilnim sistemom. Najpogostejša plačilna sistema za plačevanje preko spleta sta plačila po povzetju in plačilo s kreditno kartico.

Cilj avtomatiziranega sistema je trgovcem s spletno trgovino ponuditi orodje, s katerim lahko avtomatizirajo prenos izdelkov na želene spletne medije oz. na spletne strani z oglasnim prostorom.

Za potrebe diplomske naloge smo implementirali odprtokodno spletno trgovino OpenCart[7]. Izbira izvira iz dveh ključnih razlogov: OpenCart je ena izmed najbolj razširjenih spletnih trgovin na spletu, poleg tega je odprtokodna, kar pomeni, da lahko vsak pridobi celotno izvorno kodo. S pridobitvijo izvorne kode je preučevanje postalo mnogo lažje in enostavnejše. Naslednji korak pa je implementacija spletnega vtičnika za pridobitev izdelkov iz spletne trgovine. Cilj vtičnika je, da deluje na vseh OpenCart trgovinah. V

kolikor bi želeli uporabljati pričujočo diplomsko delo v drugi spletni trgovini OpenCart, bi na strežnik naložili t. i. vtičnik oziroma datoteko, ki skrbi za izvoz vseh izdelkov spletne trgovine v strukturiran zapis JSON, prirejen za potrebe nadzornega sistema izdelkov.

Spletna trgovina je ključni del in dejanska destinacija uporabnika, ki želi opraviti spletni nakup. Naš cilj je, uporabnika na čim bolj enostaven način pripeljati do izdelka v spletni trgovini.

1.2 Nadzorni sistem izdelkov

Nazorni sistem je potreben za delo s spletnimi trgovinami, oglasnimi vzorci in prikazovanjem informacij o uspešnosti ustvarjenih vzorcev. Vsak uporabnik nadzornega sistema od administratorja sistema pridobi uporabniško ime in geslo, ki mu omogoči prijavo v sistem. Po prijavi v sistem lahko uporabnik dodaja, ureja ali izbriše spletne trgovine, ustvarja različne oglasne vzorce in pridobi servirno kodo oglasnih vzorcev. Vse potrebne podatke o statistikah, kot so prikazi in kliki na oglas, uporabnik spremlja z izbiro oglasnega vzorca, kar lahko pripomore k boljši optimizaciji razmerja med prikazi in kliki na posamezen oglas tekom kampanje. Slednje je tudi ključnega pomena, če ima uporabnik (oglaševalec) z mediji dogovorjen zakup oglasnega prostora po prikazih.

Po uspešnem analiziranju lahko uporabnik sam ugasne ali izbriše oglasne vzorce, ki mu prinašajo slabo razmerje med kliki in prikazi. S tem pripomore k izboljšanju tega razmerja in prenosu prikazov na bolj priljubljene izdelke, kar se lahko odraža kot boljša prodaja izdelkov v spletni trgovini.

1.3 Spletna stran z oglasnim prostorom

V kolikor želimo izdelke spletne trgovine prikazati na spletni strani, potrebujemo spletno stran, ki ima določen oglasni prostor. Dobro je, da ima spletna stran oglasni prostor za oglasne pasice določene po standardu IAB[6]. Znotraj oglasov je najbolje, da serviramo oglase preko servirne kode, ki pridobi oglase iz servirnega strežnika.

Za potrebe aplikacije smo naredili servirne kode, ki določajo oglasno polje po standardu IAB[8] (formata velikosti 300x250). Vse, kar mora narediti upravljalec spletne strani, da zadovolji sistemu za prenos izdelkov na njegovo spletno stran, je implementacija servirnih kod, ki mu jih posreduje upravitelj sistema za ustvarjanje oglasnih vzorcev. Pridobljene servirne kode lahko upravljalec spletne strani vgradi neposredno v samo spletno stran ali pa jih preprosto poservira z drugim orodjem za serviranje oglasov. Za prenos in prikaz izdelkov pa poskrbi servirna koda.

Poglavje 2

Predstavitev spletnih tehnologij in orodij

Danes je vse več poudarka na tehnologiji, v kateri bo vaš sistem deloval. Zato je treba že na samem začetku razmišljati o tem, kako postaviti sistem, da bo lahko deloval na vseh platformah. Pametno je načrtovati sistem tako, da bo deloval na vseh operacijskih sistemih, vseh brskalnikih in vseh mobilnih napravah. To je nekakšen velik cilj današnjih spletnih razvijalcev. Že s samo izbiro napačnih knjižnic na primer lahko odpišemo veliko število mobilnih naprav, kar pa ima lahko velike finančne posledice. Večkrat se izkaže za pravilno pot prav ta fleksibilnost izbire pri načrtovanju sistemov.

2.1 Uporabljene tehnologije

2.1.1 PHP

Programski jezik PHP ni bil vedno takšen, kot ga poznamo danes. Čez leta je doživljal spremembe in izboljšave. Razvoj se je začel s PHP/FI, katerega prvotni namen je bil orodje za izvajanje stavkov SQL, procesiranje form in kontroliranje podatkovnih tokov. Razvoj jezika se je nadaljeval z različico PHP/FI 2.0. Ta ni vsebovala objektnega pristopa in je imela drugačno sintakso, kot jo poznamo v novejših različicah. Z verzijo PHP3 je bil narejen

celoten prepis kode iz verzije PHP/FI2.0. S PHP3 je šel ta jezik v smer objektnega pristopa, pravilne inicializacije pa še ni bilo. Ta problem so rešili tako, da so dodali objekte kot nov način definiranja polj. PHP4 je doživela pravi preboj zaradi usmirjanja jezika v smeri objektnega programiranja. Z verzijo PHP5 pa je objektni pristop postal temelj programskega jezika PHP[4].

PHP je programski jezik, ki se izvaja na spletnemu strežniku. Uporabljamo ga za obdelavo podatkov, povezovanje na podatkovno bazo, velikokrat pa kar za izpis elementov HTML. Za razliko od drugih programskih jezikov, kot je na primer JavasScript, se PHP v celoti izvaja na spletnem strežniku. To prepričuje uporabnikom pridobivanje izvorne kode PHP, vidijo samo njen rezultat. V primerjevi z programskim jezikom JavaScript, kjer se izvorna koda prenese na stran brskalnika in procesira.

PHP ni uporaben samo za zgoraj omenjeno delo, ampak z njim lahko upravljamo vsa dela klasičnih CGI programov. Predvsem pokriva tri področja uporabe:

- strežniško procesiranje,
- izvajanje skript v konzolnem načinu,
- pisanje namiznih aplikacij.

PHP se danes uporablja skoraj na vseh večjih operacijskih sistemih. Najdemo ga tako na Linuxu, Unixu, Microsoft Windowsu, Mac OS X-u, RISC OS-u itd. Z njim so podprti tudi vsi večji spletni strežniki, kot so na primer Apache, IIS, Nginx itd.[18].

2.1.2 HTML in CSS

HTML in CSS sta dve vodilni tehnologiji za izdelavo spletnih strani. HTML skrbi za strukturo spletne strani, CSS pa skrbi za videz oziroma vizualno podobo elementov na spletni strani. Obe omenjeni tehnologiji sta priznani s

strani skupnosti za razvoj standardov W3C[20].

HTML je označevalni jezik, ki ga uporabljamo predvsem za:

- objavo spletnih dokumentov,
- preusmerjanje preko hiperpovezav,
- upravljanje komunikacij z drugimi storitvami oziroma pridobivanje ali iskanje želenih informacij,
- prikaz multimedijskih vsebin, kot sta na primer video in zvok.

CSS je jezik, s katerim definiramo stil spletne strani. Pomemben je zaradi ločitve oblikovne podobe od dejanske strukture spletne strani, ki jo določa HTML. Uporablja se za definiranje oblike posameznih elementov. Z njim povemo, kako bo element izgledal, kašna bo njegova velikost, oblika, barva, kakšen font bo vseboval itd. Uporabljamo ga za prilagajanje oblike spletne strani za različne velikosti ekranov. To je še posebej priročno, če želimo uporabiti isto HTML strukturo in jo želimo uporabiti na različnih mobilnih napravah oziroma računalnikih z različno velikostjo zaslona[19].

2.1.3 JavaScript

Začetnik skriptnega jezika JavaScript je bil Brendan Eich. Njegov prvotni namen je bil narediti skriptni jezik, ki bo kot nekakšen dodatek oziroma lepilo omogočil skriptne lastnosti v spletnem brskalniku Netscape. Prvotno so jezik poimenovali Mocha, pri naslednji izdaji brskalnika je imel ime LiveScript, šele pri izdaji različice Netscape Navigator 2.0B3 pa je dobil ime JavaScript.

S širitvijo interneta je JavaScript postal med spletnimi razvijalci čedalje bolj priljubljen. Pravi preboj je doživel z uvedbo t. i. Ajax zahtevkov in rabe XMLHttpRequest. S tem je razvijalcem spletnih aplikacij dal možnost sinhronega in asinhronega prenosa podatkov med strežnikom in brskalnikom[3].

Z rastjo priljubljenosti se je jezik JavaScript začel uporabljati v raznih JavaScript knjižnicah, ki so razvijalcem spletnih strani oziroma aplikacij pomagali k preprostejši rabi in hitrejšemu razvoju. Ena izmed bolj znanih takšnih knjižnic je jQuery[13] knjižnica.

Delovanje oziroma izvajanje JavaScripte je za razliko od drugih skriptnih jezikov, ki se izvajajo na strežnikih, precej drugačno. JavaScript se najprej prenese v brskalnik in šele potem izvede. To je njena ključna razlika od drugih jezikov, kot so na primer PHP, ki se izvaja na strani spletnega strežnika.

Danes JavaScript zasledimo skoraj na vseh spletnih straneh. Uporabljamo jo za različne namene, kot so obdelovanje elementov HTML, spreminjanje CSS, pošiljanje Ajax zahtevkov itd.

2.1.4 SQL

SQL je strukturirani povpraševalni jezik, ki ga uporabljamo za delo s podatkovnimi bazami. Določen je s standardom ANSI/ISO SQL. Uporabljamo ga predvsem za pridobivanje informacij iz podatkovne baze oziroma za posodabljanje in zapisovanje v podatkovno bazo. Uporabljajo ga vsi večji sistemi za delo s podatkovnimi bazami, kot so: MS Access, SQL Server in MySQL.

2.2 Razvojne metode in pristopi

2.2.1 MVC

Pravilna izbira arhitekturnega vzorca lahko drastično vpliva na izdelavo spletne aplikacije. Kot dober primer bi vam predstavili razvojni pristop po modelu MVC. Ideja sloni na preprosti miselnosti, ki pravi, da moramo aplikacijo razdeliti na tri ključne komponente (model, pogled, kontroler), od katerih vsaka skrbi za svoj del. Te komponente povežemo v celoto in dobimo zelo transparentno zgradbo aplikacije[17].

Tri ključne komponente MVC pristopa so:

1. model (M-model): namenjen povezovanju na podatkovno bazo in načrtovanju SQL poizvedb;
2. kontroler (C-controller): zajema logiko delovanja programa ter skrbi za prenos in obdelavo podatkov med pogledom in modelom;
3. pogled (V-view): namenjen prikazu podatkov ali izpisu elementov HTML.

Prednosti pristopa MVC:
Pravilna zastavitev modela nam omogoča, da lahko z enim ukazom zamenjamo celotno podatkovno strukturo na primer iz MySQL na PostgreSQL.

Odzivnost aplikacije se lahko izboljša, če pravilno načrtujemo prenose podatkov med kontrolerjem in pogledom. Posebej problematični so prenosi, ki jih zahteva končni uporabnik oziroma klient. V takšnih primerih predlagamo uporabo formatiranega in definiranega prenosa podatkov, ki lahko drastično zmanjša velikosti samih prenesenih podatkov. Vse bolj je popularna t. i. JSON oblika podatkov.

Pravilna raba in gnezdenje pogledov lahko precej pripomore k učinkovitosti spreminjanja videza programske opreme. S pravilnim načrtovanjem pogledov lahko programer spremembe vključi na enem mestu, posledično pa popravek prenese na celotno aplikacijo.

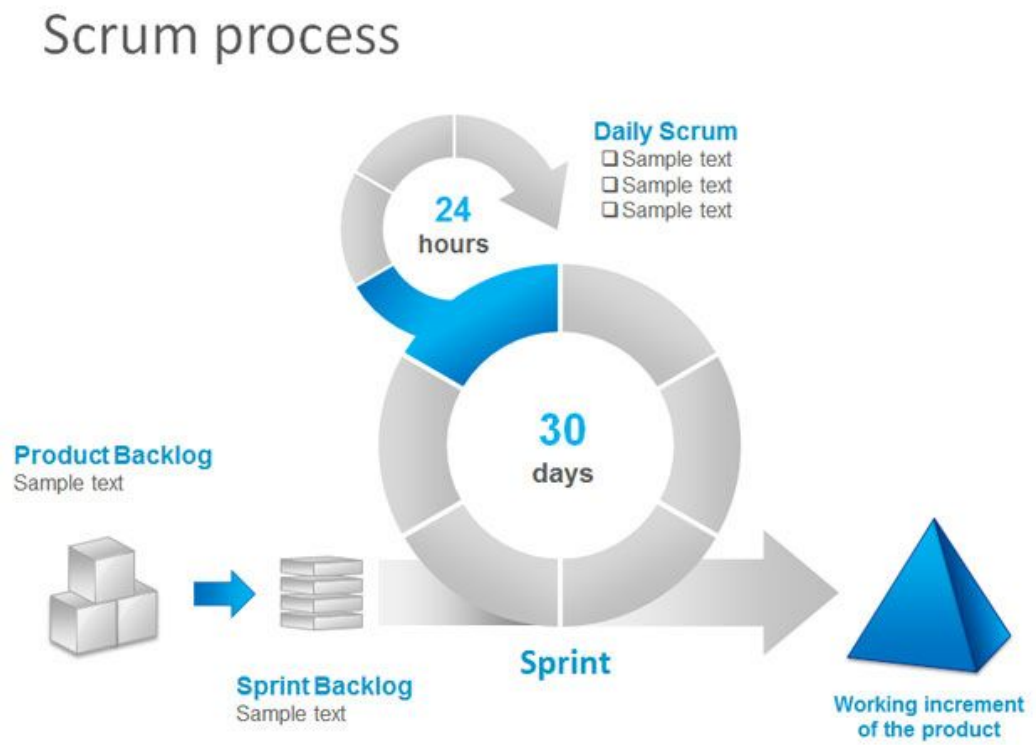
2.2.2 Agilne metode in metoda Scrum

Danes se vse več podjetij srečuje s problematiko menedžmenta in vodenja projektov. Problemi se pojavijo predvsem zaradi dinamičnosti trga, na katerem delujejo. Trgi se hitro spreminjajo, kar za podjetje pomeni, da mora biti prilagodljivo, odzivno in v koraku s časom. Agilnost podjetja pa ne pomeni samo prilagodljivosti na trg, ampak tudi agilnost znotraj podjetja. Poudarili bi, da morajo podjetja danes paziti predvsem na dolgoročnost načrtovanja in učinkovitost nad upravljanjem s sredstvi. Projekti v podjetju naj bodo

vodeni transparentno, fleksibilno in odgovorno s strani vodij projektov.

Projektno vodenje v podjetju mora biti toliko agilno, da se lahko potek projekta spremeni tako zaradi izboljšanja rezultatov kot zaradi učenja, ki smo ga osvojili skozi projekt. Poudarili bi, da spreminjanje načrta oziroma prilagajanje projekta ne sme vplivati na končne roke projekta. Lahko pa izboljšajo proces dela in pozitivno vplivajo na finančno stanje projekta[4].

Dober primer agilne metode je metoda Scrum. Scrum je proces, ki ga lahko razdelimo na tri dele. V prvem delu mora vodja projekta izdelati seznam zahtev naročnika, določiti cilje in pripraviti uporabniške zgodbe. Uporabniške zgodbe predstavi razvijalcem, ki ovrednotijo vsako zgodbo posebej. Vodja nato nadaljuje z oceno uporabniških zgodb in jih na podlagi zaključenih sklopov in pomembnosti uvrsti v t. i. »sprinte«. Drugi del procesa se izvaja v 30-dnevnih iteracijah(slika 2.1), med katerimi vodja skrbi za sestanke, usmerja razvojno ekipo oziroma skrbi za pravilno izvajanje iteracije. Zadnji del pa je namenjen poročilu iteracije oziroma pregledu rezultatov. Vodja mora biti strikten do pravilno zaključenih nalog, saj nepravilno izvedene naloge lahko pripeljejo do napake pri končnem izdelku. Zelo pomembno je, da v zadnjem delu procesa sodeluje tudi naročnik in tako dobi vpogled v sam proces in rezultate.



Slika 2.1: Scrum proces
vir slike: www.slidehunter.com

2.3 Uporabljene knjižnice in ogrodja

Pametna izbira PHP ogrodja pri razvijanju spletnih aplikacij lahko prihrani veliko časa pri pisanju že dodelanih modulov. Zato bi poudarili, da je pomembno v najprej dobro premisliti, katero ogrodje boste izbrali za svojo spletno aplikacijo. Na spletu lahko zasledite več PHP ogrodij. Med najbolj aktualnimi so naslednji:

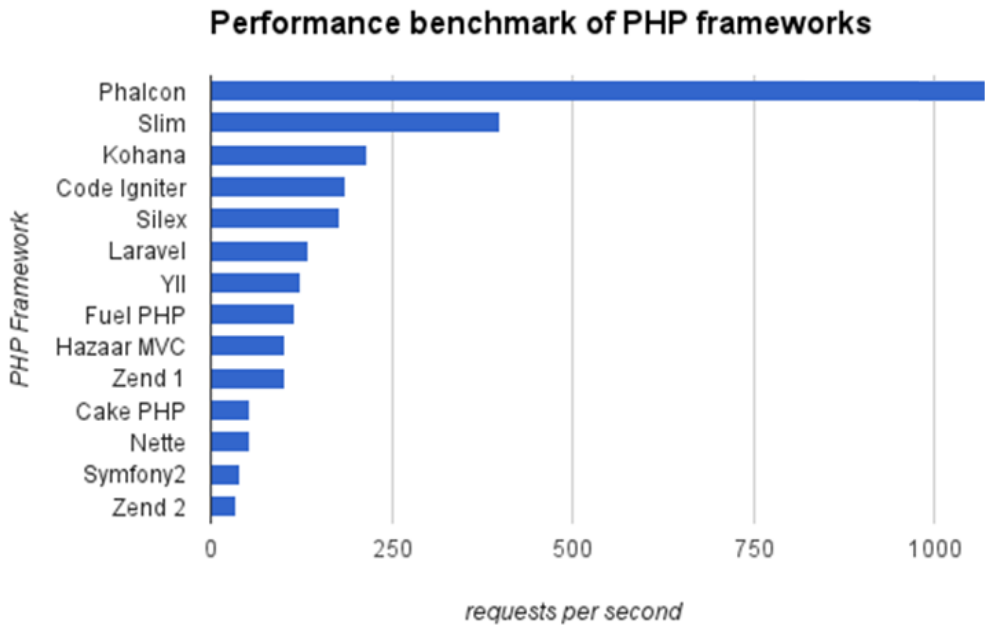
- Kohana: odprtokodno PHP ogrodje za razvoj OO MVC aplikacij. Ogrodje temelji na verziji PHP5 in strmi k hitremu in varnemu razvoju PHP aplikacij[9].

- Symfony: izjemno močno PHP ogrodje, ki je še posebej aktualno med razvijalci poslovnih aplikacij. Je eno izmed najbolj popolnih PHP ogrodij na trgu[10].

- Phalcon: ogrodje, namenjeno za obvladovanje velikega števila zahtevkov, deluje po principu kot dodatek C, vendar se koda piše v PHP[11].

- CakePHP: ogrodje, namenjeno predvsem hitremu razvoju PHP aplikacij. Če ga podrobno pogledamo, vidimo, da ima veliko modulov že vgrajenih. S tem si lahko prihranimo veliko časa in hitreje razvijemo aplikacijo[12].

Seveda pa je izbira PHP ogrodja odvisna predvsem od zahtev aplikacije in omejitev, ki jih s seboj prinese uporaba drugih dodatkov in povezovanje v druge sisteme kot pa preglednost in hitro delovanje. Zato je potrebno izbrati ogrodje, ki bo prilagojeno vašim potrebam pri razvoju aplikacije. Kot pomemben parameter bi tukaj izpostavili zahtevke na sekundo. Na sliki(2.2) si lahko ogledate graf zahtevkov na sekundo, ki vam sporoča prepustnost trenutno aktualnih PHP ogrodij.



Slika 2.2: prepustnost PHP ogrodij
vir slike: www.systemsarchitect.net

Iz grafa(slika 2.2) je razvidno, da je Phalcon eden izmed najhitrejših PHP ogrodij na trgu. Kot smo že zgoraj omenili, pa sama hitrost zahtevkov na sekundo ni edini pomemben podatek. Lahko imamo aplikacijo, kjer pride zelo malo zahtevkov na sekundo in je zato ta hitrost zanemarljivega pomena. Dejansko moramo iskati kompromis med uporabnostjo ogrodja in hitrostjo delovanja. Z uporabnostjo ogrodja lahko bistveno zmanjšamo stroške razvoja.

Danes se težko izmuznemo uporabi JavaScript knjižnic na strani klienta. Knjižnice so vse bolj priljubljene pri izgradnji uporabniških vmesnikov, razvijalci pa se nagibajo k uporabi knjižnic, ki dobro podpirajo tako obdelovanje kode HTML kot tudi njej pripetega stila. Trenutni trendi uporabe naka-

zujejo prevlado knjižnic, kot so jQuery, ExtJS, Dojo Toolkit in DHTMLX. Knjižnica, kot je na primer jQuery[13], je postala že del skoraj vsake spletne strani. To mesto si je prislužila zaradi dovršenosti in hitrosti.

Izbira JavaScript knjižnic je zelo odvisna od tega, kakšna bo naša končna rešitev. Tukaj bi mogoče kot problem navedli stil CSS. Zgoraj naštete JavaScript knjižnice imajo namreč vgrajen CSS stil, kar pa večkrat razvijalcu omejuje konsistentnost pri razvoj spletne strani z uporabljeno knjižnico. Po drugi strani pa razvijalci knjižnic to rešujejo z različnimi temami ter iščejo kompromis med stilom in razvojem spletnih aplikacij.

2.3.1 PHP ogrodje CodeIgniter

Kot smo že omenili, je izbira PHP ogrodja odvisna od zahtev aplikacije. CodeIgniter smo izbrali na podlagi izkušenj, ki jih imamo z izdelavo spletnih aplikacij. Za potrditev izbire ogrodja sledijo prednosti tega pred ostalimi konkurenčnimi ogrodji.

Prednost CodeIgniter-ja je v zmogljivosti pri obvladovanju velikega števila zahtevkov. Je zelo dobro konfigurirano ogrodje, pri katerem je malo potrebno za definiranje podatkovnih baz, spletnih naslovov in t. i. vgrajenih aplikacij. Je ogrodje, ki je vodljivo brez dodatne konzolne navigacije. CodeIgniter ima tudi veliko skupnost, ki je podprta z več tisoč razvijalci. Pregledna je tudi njegova dokumentacija, kar pa je bistveno za razumevanje delovanja celotnega ogrodja[14].

Kot pomemben parameter bi izpostavili še njegov dolgoleten obstoj na trgu. Vsako leto namreč izide nekaj novih PHP ogrodij, nekatera pa se prosto ne razvijajo več. CodeIgniter pa je na trgu že 12 let, kar mu daje prednost pred ostalimi novejšimi ogrodji, saj ima večjo podporo pri razvijalcih, ki mu sledijo že ves ta čas.

2.3.2 JavaScript knjižnica jQuery

jQuery je brezplačna odprtokodna JavaScript knjižnica. Namenjena je predvsem za manipulacijo z elementi HTML, omogoča pa tudi preproste animacije in druge razvijalcu prijetne lastnosti, kot so delo z zahtevki Ajax. Njena implementacija je zelo preprosta kot tudi samo delo z jQuery objekti. Na spletu najdete veliko dokumentacije in dobro skupnost, ki skrbi za njen hiter in odličen razvoj.

Izbira jQuery knjižnice je dobra izbira predvsem zaradi njene podpore pri skladnosti med brskalniki in zaradi njene majhnosti. Njena uporaba, v nadaljevanju diplomskega dela, bo sledila v vsaki servirni kodi za prikazovanje izdelkov, ki jih bomo generirali v nadzornem sistemu.

Kot zanimiv trend bi izpostavil storitev podjetja Google, ki ponuja gostovanje JavaScript aplikacij[15]. Med gostovanjem lahko najdemo ExtJS in jQuery knjižnici. Tri glavne prednosti, ki jih s tem pridobi razvijalec programske opreme so: (1) zmanjševanje latence (vsebina JavaScript datoteke se naloži iz najbližjega strežnika, kar je lahko dejansko hitreje v kolikor imamo en strežnik in zahtevo od oddaljenega klienta)), (2) povečevanje paralelnosti (podatki se črpajo iz več strežnikov) in (3) boljše pred-pomnjenje (lahko se zgodi, da je uporabnik JavaScripto že prinesel na drugi spletni strani in je na vaši spletni strani ne bo potrebno po prenesti, ampak se bo knjižnica naložila iz brskalnikovega predpomnilnika. V kolikor pa imate jQuery shranjen na vaši spletni strani, se bo morala prenesti vsaj enkrat.)

Primer izpisa vsebine in implementacija jQuery knjižnice iz Googlovega strežnika:

```
<!doctype html>
<html>
<head>
   <title>Preprosta implemetacija</title>
   <meta charset="utf-8" />
   <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js">
   </script>
   <script>
       $(function(){
          var content = $("#content").html();    //Pridobivanje vsebine
          console.log(content);                  //Izpis vsebine
       });
   </script>
</head>
<body>
   <div id="content">Hello world!</div>
</body>
</html>
```

2.3.3 Uporaba ogrodja Ext JS za nadzorni sistem

Ext JS je JavaScript knjižnica, ki je namenjena predvsem izdelavi spletnih aplikacij. Poleg zelo dobro definiranega objektnega modela ima vgrajen še CSS stil in več predlog, s katerimi lahko razvijalec v kratkem roku zgradi dovršeno in vizualno privlačno spletno aplikacijo.

Prednosti uporabe knjižnice Ext JS[16]:

- moderno aplikacijsko JavaScript ogrodje, z že v naprej narejenimi modeli kot so: podatkovni shranjevalniki, napredna obdelava podatkov in determinirane forme,
- napreden konsistenten grafični vmesnik z vgrajenimi grafi za boljšo vizualno predstavo podatkov končnemu uporabniku,
- orodje za izdelavo grafičnega vmesnika, kar pripomore k hitri vizualizaciji programske opreme in h krajšemu razvoju uporabniškega vmesnika,

- skladnost med brskalniki ter podpora za starejše brskalnike in operacijske sisteme,
- zelo dobra dokumentacija in podpora skupnosti.

Zahteve nadzornega sistema aplikacije, ki smo jih hoteli zadovoljiti, so bile:

- implementacija seznamov,
- implementacija oken,
- implementacija poslušalcev akcij nad elementi HTML,
- vizualizacija rezultatov,
- vzporeden prenos in obdelava podatkov brez osveževanja.

Vse zgoraj zahtevane implementacije ima Ext JS že vgrajene, zato je bilo to ogrodje najbolj primerno za izdelavo nadzornega sistema. Alternativna rešitev bi bila jQuery UI, s katero bi tudi lahko izpolnili zahteve, a smo mnenja, da tega ne bi mogli narediti na tako eleganten in pregleden način.

Seveda pa ima Ext JS kot vse ostale aplikacijsko orientirane JavaScript knjižnice svoje omejitve. Velik problem lahko nastane pri konsistentnosti, če spletna aplikacija odstopa od podane teme, ki je vgrajena v samo ogrodje knjižnice. Kot smo navedli že zgoraj, ta problem razvijalci rešujejo z več vizualnimi predlogami, vendar pa tak pristop ne more v celoti rešiti problema.

Poglavje 3

Razvoj avtomatiziranega sistema

Uporabnikom (oglaševalcem) želimo ponuditi orodje, s katerim bodo lahko svoje izdelke preprosto prenesli iz spletne trgovine preko nadzornega sistema v oglasni prostor. Da bi zadovolili omenjene zahteve, moramo posamezne komponente sistema graditi v pravilnem zaporedju saj bomo le tako lahko povezali vse komponente v celoto in zgradil avtomatiziran sistem. V naslednjih poglavjih bomo predstavili posamezne komponente sistema, ter prikazali kako načrtovati in izdelati avtomatiziran sistem.

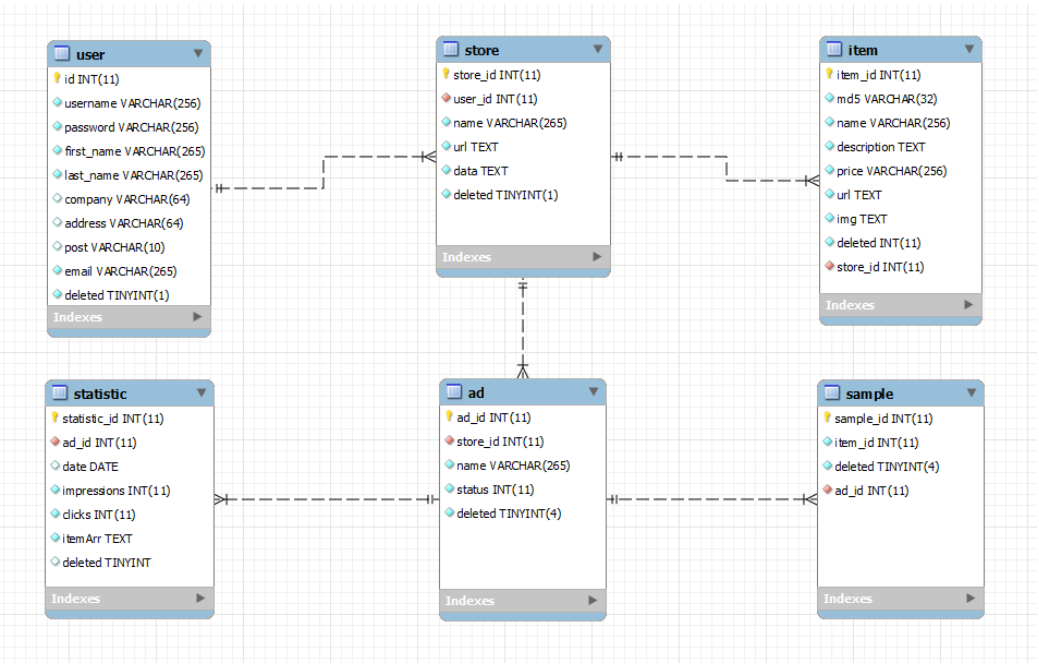
3.1 Razvoj aplikacije

Razvoj celotnega programerskega dela je potekal po metodi Scrum. Za videz grafičnega vmesnika smo uporabili minimalističen pristop, saj ne želimo uporabnika aplikacije zavajati z nepotrebnimi animacijami, barvami itd., ampak mu aplikacijo prestavimo kot koristno delovno orodje. Izpostavimo mu le funkcije, ki so za njegovo delo ključnega pomena. Menimo namreč, da ima dobra aplikacija premišljeno delovanje s strani uporabnika, in ga sili v znan tok dogodkov.

3.1.1 Načrtovanje podatkovne baze

Načrtovanje, optimizacija in minimizacija podatkovne baze so velikega pomena, kajti s tem lahko pridobimo tako pri izdelavi alikacije kot tudi pri uporabnosti oziroma hitrosti delovanja. Z optimizacijo in minimizacijo ne pridobimo le časa, ampak tudi bolj logično prestavo podatkovnega modela. Za podatkovno bazo smo izbrali MySql, načrtovanje pa smo opravili v orodju Workbench.

Pristop k našemu načrtu podatkovne baze je bil z vrha navzdol oziroma z bolj abstraktnega pogleda proti konkretnim atributom.



Slika 3.1: shema podatkovne baze

Na sliki 3.1 je prikazana shema podatkovne baze. Sestavljena je iz šestih tabel, vsaka od njih pa vsebuje primarni ključ, atribute in povezave.

Razčlenitev tabel in razlaga atributov:

- »user« – tabela je namenjena shranjevanju podatkov o uporabnikih in o njihovi prijavi v sistem. Tabela »user« je neposredno povezana s tabelo »store«, kar pomeni, da ima lahko en uporabnik v lasti več trgovin. Slednja povezava je »ena proti več«, kar lahko razberete iz zgornje slike(3.1) s črtkano črto med tabelo »user« in tabelo »store«.

- »store« – tabela je namenjena shranjevanju podatkov o spletnih trgovinah. Vsebuje podatke, kot so url do izdelkov trgovine, ime trgovine in podstrukturo, ki je kodirana v obliki JSON pod atributom »data«. Ta atribut pa vsebuje podatke o sami grafiki oglasa in ostale pomembne parametre za vizualizacijo oglasnega vzorca. Tabela je neposredno povezana s tabelo »user« v zgoraj omenjeni povezavi in »ena proti več« s tabelo »ad«. Iz tega izvemo, da ima trgovina enega lastnika in več oglasov.

- »ad« – tabela je namenjena shranjevanju oglasnih vzorcev. V njej hranimo podatke o imenu oglasnega vzorca in trenutnem stanju vzorca. Vsak vzorec je lahko v neaktivnem ali aktivnem stanju, kar se odraža kot prikazovanje oziroma ne-prikazovanje spletnega oglasa. Vsaka oglas ima svojo nadrejeno trgovino in je z njo povezan v razmerju »več proti ena«. Vsebina oglasnega vzorca pa je znana preko povezovalne tabele »sample«, ki določa posamezne informacije o izdelkih glede na oglas.

- »sample« – povezovalna tabela, ki skupaj poveže oglasni vzorec in izdelke iz tabele »item«. V obeh primerih gre za povezavo »več proti ena«, kar pomeni, da ima lahko vsak oglas več izdelkov.

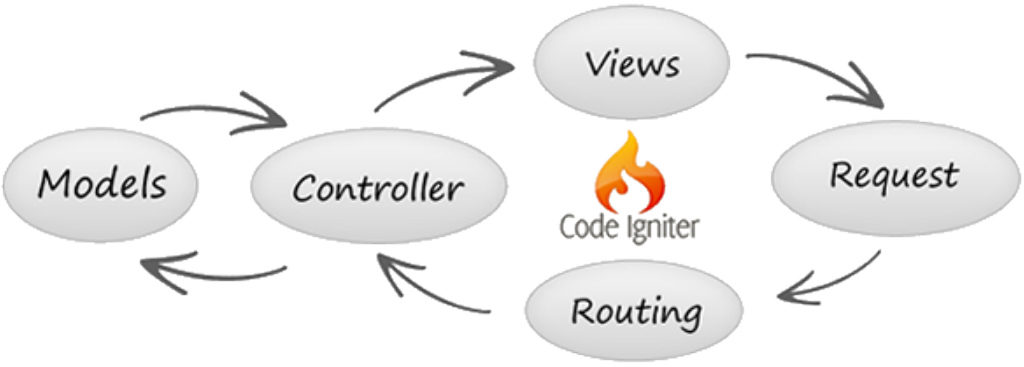
- »item« – tabela vsebuje podatke o prenesenih izdelkih iz spletne trgovine. V njej najdemo med podatki o oglasu tudi tarčo oglasa, ki je dejanska destinacija oziroma klik na oglasno mesto. Tabela je v razmerju s povezovalno tabelo »sample« in »statistic«.

- »statistic« – tabela vsebuje statistične podatke o prikazih in klikih oglasa. Preko nje dobimo povratno informacijo iz celotnega sistema. Uporablja se za beleženje in pomnjenje rezultatov. Neposredno je povezana s tabelo »ad« in tabelo »item«.

Baza je v strukturi, ki pelje od tabele »store«, ki prestavlja spletno trgovino, do tabele »item«, ki je namenjena beleženju podatka o izdelku. Gre za trostopenjsko globino. Sama ideja strukture pa izvira iz optimizacije podatkovne baze.

3.1.2 Načrtovanje PHP ogrodja

Osnova ogrodja bazira na modelu MVC, ki je bil privzet iz PHP ogrodja CodeIgniter. Struktura je razdeljena na tri osnovne dele in ima dva pomožna dela, ki skrbita za delo z zahtevki in njihovo usmeritvijo.



Slika 3.2: MVC pristop
vir slike: www.mint.rs

1. Model je PHP Class, ki je namenjen delu s podatkovno bazo. Z njim definiramo strukturo funkcij in spremenljivk, ki jih bomo kasneje uporabili za delo s tabelami v podatkovni bazi. Vse modele smo načrtovali tako, da prevzamejo minimalistično število funkcij. Izpostavili bi pet

osnovnih funkcij, ki smo jih uporabili v večini modelov, njihova glavna razlika pa je v klicu SQL stavka, ki vrne poizvedbo iz podatkovne baze.

Poudarili bi še nekaj zasnovnih vzorcev[4] modelov in imen funkcij. Ker gre za osnovne modele, se preprosto poimenujejo s kratkimi imeni: »get«, »insert«, »update«, »delete«. S tem poimenovanjem, se v nadaljevanju lažje orientiramo in razberemo katere funkcije opravljamo z določenim sektorjem baze. Primer: slaba izbira imena je getCampaign, saj s to funkcijo kličemo iz krmilnika model in šele nato ime funkcije. V tem primeru je slabo poimenovanje funkcij to, da kar dvakrat omenjamo »campaing«. Slednje je nesmiselno, saj že na začetku z izbiro modela povemo, iz katere tabele bomo pridobivali podatke.

Primer lepopisa klica funkcije modela iz krmilnika:

```
$this->campaign->get();
```

Poudarili bi, da pri nalaganju modela ne vzpostavimo povezave na podatkovno bazo, ampak to storimo v konfiguracijskem delu. Tu lahko definiramo tudi več podatkovnih baz, kar nam omogoča delo z dvema čisto različnima podatkovnima bazama. Na primer: v istem modelu lahko uporabimo podatkovno bazo MySql In PostgreSQL, kar daje programerju veliko moč nad povezovanjem in črpanjem podatkov.

2. Pogled uporabimo za prikazovanje manjših segmentov kode HMTL, kar nam omogoča, da lahko spletno stran razdelimo na več delov. Ker pogled podpira ugnezdeno strukturo, lahko znotraj enega kličemo več drugih pogledov. To hierarhijo lahko programer razdeli tako globoko, da definira vsak element posebej. Slednje mu omogoča, da s spremembo enega manjšega segmenta zamenja celotno strukturo oziroma obliko segmenta. Dober primer je kreiranje segmenta za gumb, ki se uporablja na več delih aplikacije, sprememba tega pa bo vplivala na

vse dele oziroma v vseh pogledih aplikacije. Programer lahko s pravilnim segmentiranjem pridobi veliko časa in vizualno podobo spreminja samo v enem pogledu. V pogledu lahko uporabljamo tudi kodo PHP, to pa lahko izkoristimo tudi za generiranje kode JavaScript in elementov HTML.

V diplomskem delu smo uporabili osnovno segmentacijo za prikazovanje pogledov in pridobivanje podatkov preko zahtevkov. Naša pot črpanja podatkov v pogled pa je bila preko modela v krmilnik in nato nazaj v pogled.

3. Krmilnik je PHP Class, ki poskrbi za logiko delovanja aplikacije. Neposredno je povezan z URL-jem, ki ga kličemo iz brskalnika. V njem nalagamo in kličemo modele za delo s podatkovno bazo ter funkcije za obdelavo podatkov. Pridobivanje podatkov s strani uporabnika poteka preko metod POST in GET. Pravilno delovanje krmilnika je v klicu URL-ja v naslednjem vrstnem redu:

   - zahteva preko URL-ja na krmilniku s podatki preko POST ali GET metode,
   - inicializacija krmilnika in modelov,
   - pridobivanje podatkov preko metode POST ali GET,
   - obdelava podatkov in morebiten zapis v podatkovno bazo,
   - klic preprostega ali naprednega pogleda s prenosom podatkov vanj.

   Kot smo omenili že zgoraj, imamo še dva pomožna dela. Prvi del skrbi za t. i. krmiljenje, ki je namenjeno definiranju poti in preusmerjanju. Vsak URL lahko preusmerimo na želeno lokacijo ali poljuben krmilnik in tako preusmerjamo aplikacijo.

   Drugi del, ki skrbi za zahtevke, delimo na dva načina: (1) t. i. klici na URL, s katerimi želimo vsebino spletne strani z obliko in z ostalimi JavaScript datotekam, ter (2) klice, ki jih potrebujemo za naknadno obdelavo in pridobivanje podatkov. Te klice imenujemo zahtevki Ajax, ki

v osnovi niso nič drugačni od pravih klicev, vendar je njihova bistvena prednost v tem, da se spletna stran pri takšnem klicu ne osveži. Namenjeni so naknadnemu pridobivanju podatkov za posebne komponente, ki se lahko izvajajo asinhrono.

3.1.3 Načrtovanje grafičnega vmesnika

Grafični vmesnik želimo načrtovati tako, da bo uporabnika vodil skozi jasen potek dogodkov in ga bo hkrati spoznal s funkcionalnostjo sistema. Na podlagi žičnega izrisa bomo predstavili tok dogodkov in delovanje aplikacije.

Slika 3.3: žični izris aplikacije

Grafični vmesnik smo zasnovali tako, da tok dogodkov poteka iz levega zgornjega kota in se širi proti desnemu spodnjemu kotu. Predstavljamo si ga lahko tudi kot nadzorno ploščo, iz katere lahko s preprostim dvoklikom razberemo uspešnost oglasnega vzorca.

Žični izris(slika 3.3) predstavlja nadzorni sistem, razdeljen na štiri glavne segmente:

1. **Nadzor nad spletno trgovino:** v zgornjem desnem kotu dodamo

novo spletno trgovino, izpolnimo vse potrebne podatke in jih potrdimo. Novo dodana trgovina se nam bo pokazala na »Seznam spletnih trgovin«. Če jo želimo naknadno urediti, jo odpremo v formi z dvoklikom.

2. **Ustvarjanje oglasnih vzorcev:** zaradi pomembnosti smo ga uvrstili na sredino nadzornega sistema, tako je uporabnik osredotočen na ustvarjanje in testiranje vzorcev. Nov oglasni vzorec se doda preko zgornjega gumba »Dodaj« pod segmentom »Seznam oglasnih vzorcev«. Ob potrjevanju se novo kreirani vzorec prikaže na »Seznam oglasnih vzorcev«. Uspešno shranjeni oglasni vzorec povzroči prikaz okna, iz katerega pridobimo servirno kodo. Seznam urejamo na enak način kot »Seznam spletnih trgovin«.

3. **Vizualizacija ustvarjenih vzorcev:** v levem sredinskem delu lahko vidite trenutno izbran oglasni vzorec in njegov videz. Uporabnik lahko pregleda ustvarjene vzorce in testira povezave. Ta segment je pomemben, saj si lahko prek njega lažje predstavljamo rezultate ustvarjanja vzorcev. Predogled oglasa je instanca kreirane servirne kode.

4. **Poročilo o uspešnosti oglasnih vzorcev:** iz spodnjega grafa lahko razberemo odzivnost posameznega vzorca skozi čas. Predstavljena sta dva ključna parametra: prikazi in kliki. Prikaz je servirani oglas, klik pa definiramo kot uporabnikov klik na enega izmed izdelkov. Njuno razmerje se imenujemo RPK. Levi zgornji kot je namenjen prikazu uspešnosti posameznih izdelkov (graf klikov na posamezni izdelek). Ta prikazuje, kateri izdelek je v posameznem vzorcu najbolj odziven in kateri najmanj.

Vse segmente smo povezali v arhitekturo, ki deluje po principu filtra izbire. Nadzorni sistem si lahko predstavljate kot tok dogodkov, ki poteka iz levega zgornjega kota proti desni strani aplikacije.

Klasičen primer uporabe poteka takole: na seznamu »Vaše spletne trgovine« izberemo želeno spletno trgovino in osveži se nam segment 2, ki

prikazuje seznam spletnih vzorcev izbrane spletne trgovine. Klik na oglasni vzorec v segmentu 2 pa prikazuje informacije v segmentu 3 in 4. S tem osveževanjem lahko uporabnik v dveh korakih pridobi želene informacije o trenutnem stanju oglasnega vzorca.

3.1.4 Komunikacija in povezovanje

Načrtovanje povezovanja je šlo predvsem v smer, da imamo nadzorni sistem za centralno aplikacijo, ki nadzoruje tako zahteve na vtičnik spletne trgovine kot tudi spreminjanje servirnih kod na posamezno spletno mesto. Načrtovanje prenosa je definirano tako, da mora spletni vtičnik poskrbeti za izvoz vseh izdelkov spletne trgovine in na definiran url izpisati izdelke v dogovorjenem formatu izdelkov v obliki JSON. Format izdelkov si lahko predstavljamo kot polje objektov, pri čemer objekti predstavljajo zapise posameznih izdelkov.

Primer objekta izdelka:

```
{
  "price":"Cena izdelka",
  "img"  :"Pot slike izdelka",
  "name" :"Ime izdelka",
  "description":"Opis izdelka",
  "url":"Tarča izdelka"
}
```

Nadzorni sistem izda zahtevo na omenjeni URL in prenese vse kodirane izdelke. Izdelke je potrebno najprej dekodirati in jih zapisati v podatkovno bazo, kjer se zapišejo v pravilen format za kasnejše obdelovanje oziroma jih pripravimo za kasnejše ustvarjanje vzorcev.

Drugi del povezovanja je namenjen prenosu kode javaScript na želeno spletno mesto. Poudariti je treba, da se generirane kode spreminjajo na uporabniško zahtevo. Vsaka koda, ki jo uporabnik ustvari, se zapiše na strežnik in do nje poda pot. Tako imamo en oglasni vzorec in eno servirno kodo na

točno določenem naslovu strežnika. S tem rešimo problem osveževanja vzorcev na več spletnih straneh hkrati. Če namreč spreminjamo oglasni vzorec že v vnaprej določenem naslovu, se spremeni vsebina generirane kode na tem naslovu, posledica tega pa je osvežitev oglasnega vzorca na vseh spletnih straneh, ki imajo vstavljeno to kodo. Zadnji del nadzornega sistema je namenjen komunikaciji in meritvam. Cilj je prikazati uporabniku povratno informacijo o prikazih in klikih ustvarjenih vzorcev. Da bi zadostili potrebam, je treba vzpostaviti sistem, ki bo zabeležil vsak prikaz oglasa in vsak klik na izdelek v oglasu. Ob pravilno serviranem oglasu se zabeleži zahtevek v statični tabeli, enako velja za klik na posamezen izdelek. Obdelujejo se podatki vseh zahtev o serviranju in klikanju, po končanem procesiranju pa se prikažejo v nadzornem sistemu. Klik na izdelek bi lahko z drugimi besedami opisali kot uvod v nakupni proces uporabnika, saj s klikom na izdelek potencialnega kupca preusmerimo na želeni izdelek znotraj spletne trgovine.

3.1.5 Vizualizacija rezultatov

Vizualizacijo rezultatov lahko razdelimo na dva skopa. Prvi sklop bi bila t. i. vizualizacija oglasa. Vsak ustvarjeni oglas lahko vidimo v predogledu, kar nam prikaže vzorec, ki ga bomo servirali na spletni strani. S tem imajo uporabniki nadzornega sistema takojšen vpogled in lažjo predstavo, kakšen oglas so ustvarili.

Drugi sklop zajema vizualizacijo rezultatov odzivnosti posameznega oglasnega vzorca. Prikaz smo načrtovali v dveh pogledih. Prvi pogled uporabniku prikaže graf odvisnosti klikov na posamezen izdelek, s katerega uporabnik razbere informacijo o odzivnosti posameznih izdelkov. Uporabnik se nato lahko odloči, ali bo določen izdelek izločil iz vzorca ali pa ga bo po potrebi zamenjal z novim. Drugi pogled je namenjen prikazu odzivnosti posameznega oglasa. Na grafu imamo predstavljene tri najbolj pomembne parametre (prikazi, kliki in RPK) skozi čas. Iz tega uporabnik pridobi informacijo, na kateri dan je oglas zabeležil največ prikazov ter klikov oziroma prikaže razmerje RPK.

3.2 Izdelava aplikacije

3.2.1 Izdelava vtičnika za spletno trgovino

Eden izmed pomembnih delov sistema je črpanje izdelkov iz spletne trgovine preko namenskega spletnega vtičnika. Za potrebe aplikacije smo postavili eno izmed najbolj pogostih odprto-kodnih spletnih trgovin.

Spletni vtičnik smo načrtovali tako, da imamo v danem trenutku na razpolago vse izdelke spletne trgovine. Predvsem smo hoteli doseči to, da bo vtičnik upravitelju spletne trgovine prijazna programska oprema. Vse, kar mora upravljavec trgovine narediti, da zadovolji potrebe po implementaciji vtičnika, je prosta naložitev datoteke na koren spletne trgovine, za vse ostalo poskrbi vtičnik sam.

Vtičnik ob zagonu sam pridobi nastavitve podatkovne baze iz spletne trgovine in se z njo avtomatsko poveže. Iz podatkovne baze začne črpati podatke in jih shranjevati v začasne obdelovalne tabele. V tabelo shranimo vse pomembne parametre izdelkov, vključno s potmi slik in dodatno ustvarjenimi absolutnimi potmi izdelkov v spletni trgovini. Absolutne poti izdelka bomo naknadno uporabili kot konverzijsko tarčo izdelka v ciljni spletni trgovini. Zapise tabel kodiramo v strukturo JSON in jih izpišemo kot rezultat vtičnika.

Rezultat vtičnika je urejena struktura tekočih izdelkov spletne trgovine, ki so zapisani v pravilni obliki za prenos v nadzorni sistem aplikacije.

3.2.2 Izdelava nadzornega sistema

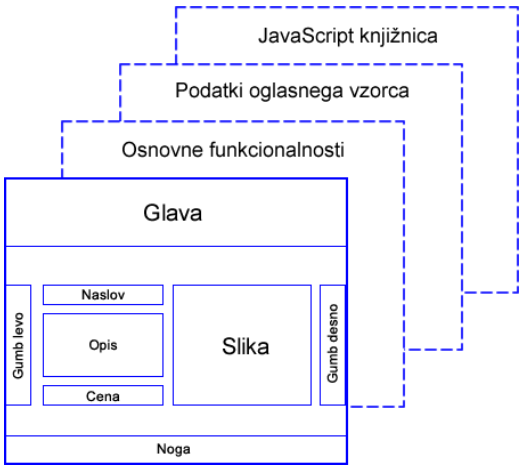
Izdelava nazornega sistema je potekala v sklopu zgornjega načrta. V praksi je bilo treba prvotni načrt grafičnega vmesnika razdeliti na sklope in te uvrstiti v zgodbe, ki smo jih kasneje obdelovali v iteracijah po metodi Scrum. Izde-

lava grafičnega vmesnika je samo en del kode, ki smo jo napisali za realizacijo nadzornega sistema. Naprej smo ustvarili podatkovni model iz poglavja 3.1.1 in ga vstavili v podatkovno bazo MySql. Naslednji korak je bil postavitev ogrodja CodeIgniter in realizacija vseh Scrum zgodb, ki so bile ključnega pomena za realizacijo nadzornega sistema. Na omenjeno strukturo smo tako lahko postavili grafični vmesnik.

Grafični vmesnik nazornega sistema je v celoti zgrajen z javaScript knjižnico ExtJS. Vsak del posebej smo objektno določili in ga povezovali z ogrodjem na strežnik spletne strani preko zahtevkov in odgovorov. S takšnim pristopom smo grafični vmesnik realizirali na način, v katerem se ta grafični vmesnik v celoti prenese samo enkrat, naknadno pa se povezuje oziroma osvežuje z zahtevki na ogrodje aplikacije, ki je definirana na strežniški strani. Videz form, oken in grafov smo prevzeli iz standardnih objektov knjižnice Extjs.

3.2.3 Generiranje servirnih kod in oglasov

Tretji ključni del sistema je generiranje servirnih kod. Servirna koda, ki vsebuje oglasni vzorec, obliko oglasa in vse potrebne funkcionalnosti, je sestavljena iz štirih plasti.



Slika 3.4: plasti servirne kode

Zgradba servirne kode:

1. **JavaScript knjižnica:** osnovno delovanje oglasa bazira na jQuery knjižnici. Knjižnico bi lahko zamenjali s kakšno svojo skripto, ki bi opravljala iste naloge. Vendar smo mnenja, da je potrebno uporabiti znanja več ljudi in uporabiti knjižnice, v k katere so strokovnjaki vložili veliko truda. Poleg tega s tem pridobimo čas in preprečimo morebitne napake programske kode. Uporaba knjižnice je namenjena predvsem animiranim delom in delom z zahtevki Ajax. Kodo smo pridobili v stisnjeni obliki in jo pripeli na prvo plast servirne kode.

2. **Podatki oglasnega vzorca:** vsaka servirna koda poleg ostalih funkcionalnosti vsebuje še vse podatke oglasnega vzorca. Če želimo te podatke, je potrebno pridobiti informacije o izdelkih, ki jih želimo prikazati v oglasu. Ker vzorec lahko vsebuje več izdelkov, je treba pridobiti podatke o vsakem izdelku, ki se nahaja znotraj vzorca, posebej. Pridobljene podatke se shrani, obdela in zapiše v polje objektov. Z drugimi besedami zgradimo mrežo izdelkov. Urejene podatke se kodira v strukturo JSON in pripne na drugo plast servirne kode. Naslednji korak je pripenjanje slik iz spletne trgovine. Vsak izdelek ima svojo sliko, zato je potrebno pridobiti vse slike izdelkov znotraj vzorca in jih pripeti v podatkovno strukturo. Podali vam bomo še namig o optimizaciji. Če imamo slike, shranjene na strežniku, je za prenos vsake slike potreben zahtevek, kar pomeni, da je število povezav odvisno od števila izdelkov. Posledica tega je, da se število povezav povečuje z velikostjo oglasnega vzorca, kar lahko postane velik problem pri velikem številu izdelkov v oglasnem vzorcu. Na primer, če imamo 30 izdelkov v oglasnem vzorcu, se ob prikazu oglasa vzpostavi 30 povezav do podatkovnega strežnika.

Izrek 3.1 *Za vsak prenos oglasa velja:*

$$stevilo\ slik + prenos\ oglasa = stevilo\ povezav \tag{3.1}$$

Optimizacijski rešitvi:

- Vse slike se prenese z enim samim zahtevkom. To dosežemo tako, da fotografije zlepimo skupaj v eno dolgo sliko. Nato pa s CSS ukazi pomikamo ozadje slike z odmikom dolžine fotografije. Slabost tega algoritma je, da slik ne moremo več proporcionalno zmanjševati.

  Izrek 3.2 *Vsakemu prenosu oglasa prištejemo dodaten prenos slike:*

  $$prenos\ oglasa + 1 = stevilo\ povezav \tag{3.2}$$

- Vse slike se kodira v Base64 in kodne zapise shranjuje v tabelo. Ključ tabele je ime slike, vrednost pa Base64 zapis slike. Nato se celotna tabela zapiše v obliko JSON in se shrani v servirno kodo oglasa.

  Izrek 3.3 *Za vsak prenos oglasa velja:*

  $$prenos\ oglasa = stevilo\ povezav \tag{3.3}$$

Za končanje druge plasti servirne kode se pripne JavaScript polje objektov, katerih vrednost so Base64 zapisi slik.

3. **Osnovne funkcionalnosti:** Vsak oglas vsebuje osnovne funkcionalnosti: premikanje izdelkov levo in desno ter merjenje prikazov in klikov na izdelek. Premikanje smo rešili z navigacijskimi gumbi, ki krmilijo izdelke levo in desno ter tako menjavajo vsebino oglasnega prostora znotraj ene dimenzije. S tem lahko znotraj ene dimenzije prikažemo

več izdelkov in tako optimiziramo prikaze izdelkov oziroma prostornino oglasa. Za posamezen premik se uporabi funkcija, ki animira odmik vsebine od centralnega prostora oglasa.

Beleženje prikaza se izvede v trenutku, ko je oglas prikazan na spletnem mestu, klik pa ob uporabnikovi interakciji z izdelkom. Uporabnika se potem preusmeri v spletno trgovino.

4. **Oblika oglasa:** Vsak oglas ima svojo obliko, ki jo definira uporabnik. Ta lahko spremeni obliko pod nastavitvami spletne trgovine v nadzornem sistemu.
   Pri prilagoditvi videza bi izpostavili še dva načina, kako spremeniti obliko:

   - Prvi način je ta, s katerim spreminjamo tri osnovne slike oglasa. Kazalec za premikanje izdelkov v smeri levo in desno ter slika, ki je v ozadju. S temi tremi elementi lahko popolnoma zamenjamo videz oglasa.
   - Drugi način je ta, s katerim lahko z nekoliko programerskega znanja spremenimo vse položaje elementov in ostalih parametrov. Vsak del lahko tudi drugače obarvamo ali uresničimo druge specifične potrebne oziroma želje naročnika. V tem primeru je mogoče pripeti tudi dodatne datoteke JavaScript, ki opravljajo posebne animacije.

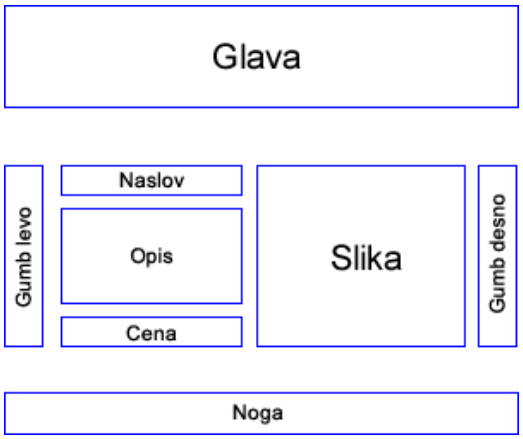
   Oblika oglasa se na četrto plast pripenja v dveh delih. Najprej se pripne koda za videz in položaje posameznih elementov, nato pa se pripne še tabela kodnih zapisov slik v formatu base64. Uporabimo optimizacijski algoritem iz zgoraj navedene rešitve(3.3).

Oglas je sestavljen, ko smo uspešno sestavili vse štiri plasti v eno da-

toteko JavaScript. Dobljeno datoteko posnamemo na spletni strežnik in ji določimo unikatno ime. Ime bo kasneje natančno določalo spletni prostor servirne kode.

3.3 Zgradba oglasa

Današnji trendi v oglaševanju podpirajo organizacijo IAB, ki je formalizirala osnovne nastavitve in dimenzije oglasov. Ker smo v diplomskem delu sledil temu, smo se odločali med osnovnimi oglasnimi formati po standardu IAB, ki bi bili najbolj primerni za oglaševanje spletnih izdelkov. Po tehtnem premisleku smo se odloči za pravokotno obliko 300x250, saj takšna površina zadostuje potrebam oglasnih vzorcev. Na podlagi tega smo naredili žični izris oziroma načrt položajev elementov znotraj oglasa, dimenzije 300x250.



Slika 3.5: žični izris posameznih elementov

Na sliki 3.5 je prikazan žični izris posameznih elementov, ki smo jih uporabili za osnovne komponente oglasa. Razporejeni so v 3 skupine:

- glava in noga sta narejeni tako, da odražata celostno grafično podobo (CGP) podjetja,

- naslov, opis, cena in slika so osrednja vsebina oglasa in so za razliko od ostalih elementov spremenljive. Posebnost osrednje vsebine je, da mora biti to območje klikabilno, ciljna spletna stran pa je spletna trgovina,

- »gumb levo« in »gumb desno« skrbita, da se osrednja vsebina pomika v smeri levo ali desno. S to funkcijo premikanja dobimo dodaten prostor znotraj oglasnega prostora. Tako na preprost način prikažemo N-število izdelkov, kar daje oglaševalcu konkurenčno prednost pred klasičnim oglasom dimenzije 300x250.

Poglavje 4

Delovanje in implementacija aplikacije

Celoten sistem je sestavljen iz treh delov, zato mora administrator sistema poskrbeti za implementacijo vtičnika v spletno trgovino ter zagotoviti postavitev nadzornega sistema. Tretji del oziroma del, s katerim namestimo generirane kode na spletno mesto, pa lahko opravi kar upravitelj spletnega mesta v sodelovanju z administratorjem sistema, ki generirane kode posreduje.

Nadzorni sistem je zasnovan tako, da se mora vsak uporabnik naprej prijaviti v sistem s svojim uporabniškim imenom in geslom. Po uspešni prijavi je preusmerjen v nadzorni sistem, kjer lahko začne z uporabo sistema. Nadzorni sistem uporabniku omogoča postavitev osnovnega delovnega okolja. Uporabnik mora najprej dodati spletno trgovino, iz katere z izbiro postane koren naslednjih dogodkov nadzornega sistema. Sledi ustvarjanje oglasnih vzorcev in avtomatski prenos izdelkov v sistem. Uporabnik mora nato le še izbrati želene izdelke. Rezultat je generirana servirna koda, ki jo vstavi na želeno spletno stran.

4.1 Implementacija vtičnika in nadzornega sistema

4.1.1 Nameščanje vtičnika v spletno trgovino

Nadzorni sistem ne more delovati brez pravilno nameščenega vtičnika. Pri dodajanju spletne trgovine se namreč vzpostavi povezava do vtičnika, iz katerega se črpajo izdelki spletne trgovine.

Trgovec mora imeti implementirano spletno trgovino OpenChart. Za pravilno delovanje spletnega vtičnika je potrebno namestiti posebej prirejeno kodo na koren spletne trgovine OpenCart. Vse, kar mora administrator spletne trgovine storiti, je prenesti datoteko »products.php«. Datoteka vsebuje kodo vtičnika, ki poskrbi, da se vtičnik avtomatsko poveže s podatkovno bazo spletne trgovine in pridobi vse njene izdelke. Rezultat uspešno nameščenega vtičnika je JSON izpis izdelkov na spletnem naslovu »url_spletne_trgovine/products.php«.

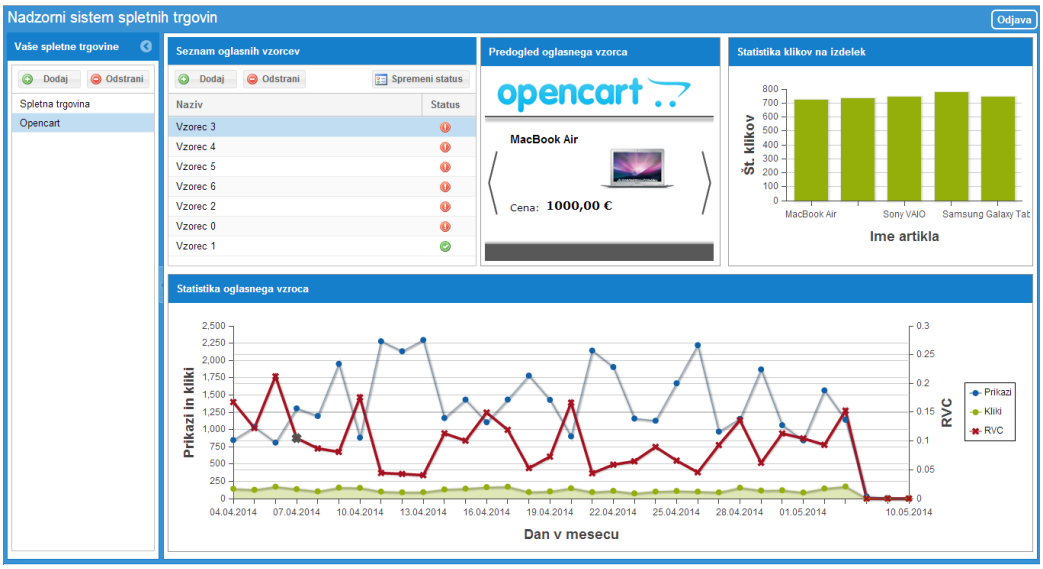
4.1.2 Postavitev nadzornega sistema

Za postavitev nadzornega sistema je potrebno imeti spletni strežnik (priporočen je spletni strežnik Apache) in podatkovno bazo MySql. Prvi korak namestitve je nameščanje izvorne kode nadzornega sistema na želeno spletno mesto. Sledi ustvarjanje podatkovne baze po modelu, omenjenem v poglavju 3.1.1 in konfiguracija podatkovnih nastavitev v ogrodju CodeIgniter. Tu je potrebno le še spremeniti podatke o povezovanju na podatkovno bazo, ki jih najdemo v datoteki »database.php« v konfiguracijskem delu. Po uspešno ustvarjeni in definirani bazi podatkov, lahko administrator začne z vnašanjem uporabnikov v nadzorni sistem.

4.2 Izgled nadzornega sistema

Videz nadzornega sistema je načrtovan tako, da tok dogodkov teče z levega dela aplikacije, kjer uporabnik dodaja spletne trgovine in ustvarja oglasne vzorce, proti desnemu delu, ki je namenjen predogledu oglasnega vzorca in statističnemu poročilu izbranega vzorca.

Na levi strani imamo seznam spletnih trgovin. Uporabniki lahko omenjeni seznam urejajo preko gumbov na vrhu seznama. Centralni del nadzornega sistema je namenjen dodajanju, urejanju, odstranjevanju in spreminjanju statusa aktualnih spletnih vzorcev. Slednja dejanja pa se aktivno odražajo na predogledu oglasnega vzorca, kjer lahko uporabnik tudi preveri dejanski videz oglasa. Desni in spodnji del je v celoti namenjen statistiki trenutno izbranega vzorca. Statistika je razdeljena na dva dela. Prvi del oziroma desni kot nadzornega sistema je namenjen statistiki klikov posameznega izdelka v oglasnem vzorcu. Iz njega lahko uporabnik razbere, kateri izdelek je v vzorcu najbolj zaželen, oziroma kateri ima najslabšo odzivnost. Spodnji del je v celoti namenjen časovnemu poteku RPK-odzivnosti, iz katere lahko razberemo učinkovitost delovanja oglasa na posamezen dan.



Slika 4.1: izgled nadzornega sistema

Slika 4.1 prikazuje nadzorni sistem. Iz njega lahko razberemo, da uporabnik preverja statistični vzorec pod imenom »Vzorec 3«, ki ga je ustvaril za spletno trgovino z imenom »Opencart«. Na sredini lahko opazimo ustvarjen oglas, desno in spodaj pa statistične podatke oglasov iz preteklega meseca oz. aktivnih oglasov.

4.3 Funkcionalnosti aplikacije

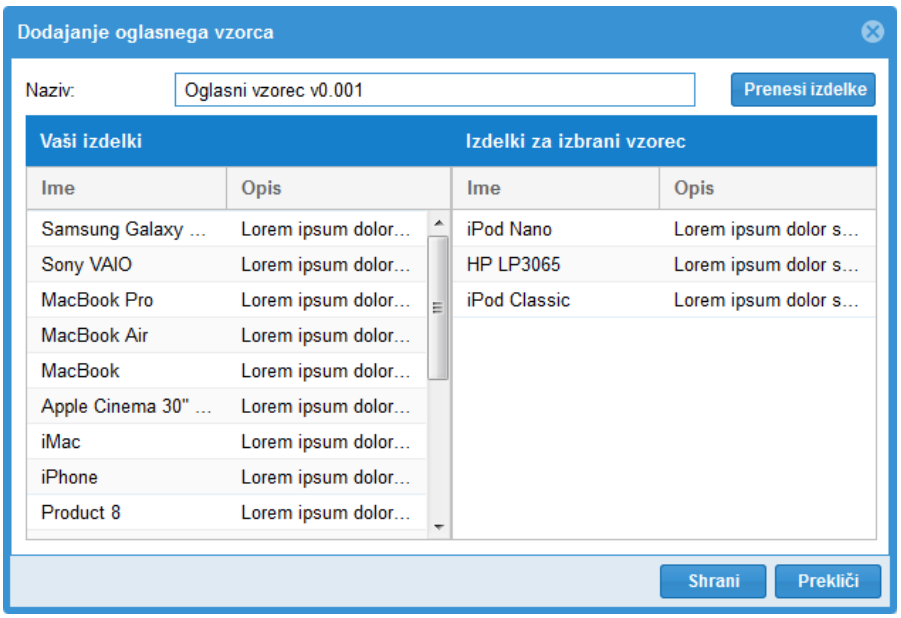
4.3.1 Dodajanje spletne trgovine

Uporabnik nadzornega sistema lahko doda eno ali več spletnih trgovin. Za uspešno dodajanje spletne trgovine mora uporabnik imet žei v naprej nameščen vtičnik za pridobivanje izdelkov. Trgovino lahko vnese pod rubriko »Vaše spletne trgovine«. Za uspešno dodajane mora vnesti URL izdelkov oziroma pot do spletnega vtičnika, kjer so pravilno kodirani izdelki. Naslednji korak je definiranje poti slik, ki določajo obliko oglasa. Za vsako trgovino je predpostavljena ena grafična predloga, ki je sestavljena iz treh slik. Potrebni sta

dve sliki za navigacijske gumbe levo in desno ter tretja slika za celotno ozadje oglasa.

4.3.2 Ustvarjanje oglasnih vzorcev

Ustvarjanje oglasnih vzorcev je mogoče pod rubriko »vaši oglasni vzorci«. Uporabnik mora najprej pridobiti izdelke spletne trgovine s klikom na gumb »Prenesi izdelke«. Iz spletne trgovine se mu avtomatsko prenesejo vsi izdelki, ki jih kasneje vidi pod izbirnim pregledom »Vaši izdelki«. Iz izbirnega pregleda lahko uporabnik s klikom izbere želene izdelke za oglasni vzorec. Novo ustvarjeni oglasni vzorec bo sestavljen iz izdelkov pod izbirnim pregledom »Izdelki za izbrani vzorec«. Uporabnik lahko oglasni vzorec še poimenuje in shrani.



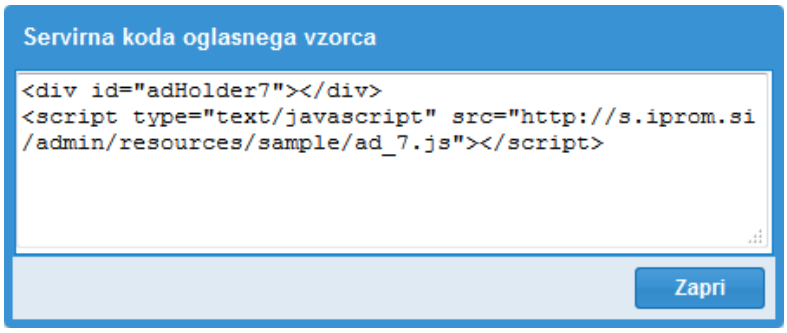
Slika 4.2: ustvarjanje oglasnih vzorcev

Slika 4.2 prikazuje formo za dodajanje oglasnih vzorcev. Opazimo lahko, da je uporabnik sistema ustvaril oglasni vzorec z imenom »Oglasni vzorec

v0.001« in za slednjega izbral izdelke: iPod Nano, HP LP3065 in iPod CLassic.

4.3.3 Pridobivanje in vstavljanje servirnih kod

Po ustvarjanju vzorca le-tega shranimo. Nato se pojavi okno »Servirna koda«. Rezultat uspešno ustvarjenega vzorca je servirna koda v pojavnem oknu. Pridobljeno kodo je potrebno vstaviti na spletno stran za želeno oglasno mesto v velikosti 300x250(pix). Kodo lahko na oglasno mesto vstavimo tudi preko sistema za serviranje oglasov (ad server). Velika večina oglasnih strežnikov takšne kode obravnava pod naslovom »Third-Party Code« ali kode tretjih ponudnikov.



Slika 4.3: ustvarjanje oglasnih vzorcev

Slika 4.3 prikazuje rezultat uspešnega ustvarjanja servirne kode. Servirna koda je sestavljena iz dveh delov oziroma iz enega »div-a«, ki je namenjen poziciji oglasa na spletni strani, in generirane datoteke JavaScript, ki naloži vse potrebne funkcije in obliko oglasa. Takšna koda ni sestavljena na tak način samo zaradi lažjega spreminjanja pozicije oglasa na spletni strani, pač pa tako sestavljena koda pripomore tudi k izboljšanju odzivnosti spletne strani. Morajo pa biti kode pravilno implementirane. Dobra praksa pravilno implementirane kode je, da se kodo naloži v nogo spletne strani, kar povzroči, da se bo spletna stran prikazala pred oglasom.

4.4 Razbremenjevanje in optimizacija

Razbremenjevanje in optimizacija sta dva ključna parametra pri izvedbi spletne aplikacije, namenjene komercialni rabi. Poudariti bi želeli predvsem optimizacijo na strani prenosov iz spletnega strežnika in minimizacijo ter združevanje datotek. Če bi želeli našo programsko opremo uporabljati v večji meri, bi morali poudarek dati na dostopnosti servirnih kod. Servirne kode so sestavljene tako, da vsebujejo vse podatke o oglasu in njegovem videzu. Temu bi lahko rekli kar oglas v paketu oz. JavaScript datoteka. Takšna oblika je bila načrtovana že od samega začetka prav zaradi omenjenih problemov prenosa in razbremenjevanja strežnikov.

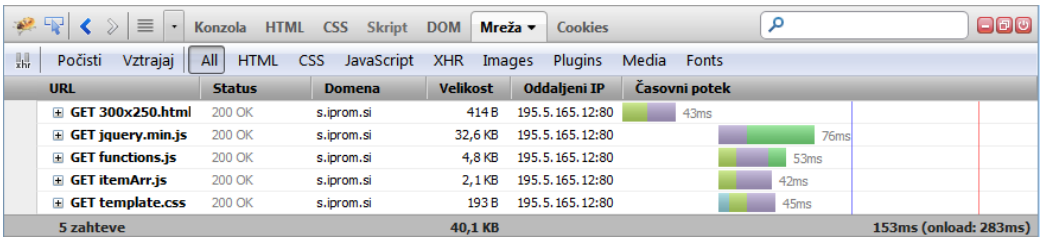
4.4.1 Razbremenjevanje

Paketni dostop do oglasa nam omogoča, da do njega dostopamo z enim samim zahtevkom. Slednje je bistvenega pomena, kajti več kot imamo zahtevkov do spletnega strežnika, več procesorske moči potrebujemo na strani strežnika. Z vsakim dodatnim zahtevkom odzivnost strežnika pada zaradi povečevanja latence. Zelo dobra razbremenjevalna metoda je metoda z uporabo CDN.

Na spletno stran namestimo servirne kode iz več različnih CND-jev, kar nam omogoča, da se servirne kode, ki prikazujejo oglase, naložijo paralelno. S paralelnostjo razbremenimo strežnike in zmanjšujemo latence ter pospešimo nalaganje spletne strani oziroma oglasov na spletni strani. Zelo dobro je tudi, da poskrbimo za optimizacijo in minimizacijo kode. Predlagamo še, da datoteke stisnemo in kodiramo z orodji za delo z datotekami JavaScript. Aktualnih orodij za stiskanje omenjenih datotek je na spletu kar nekaj in se neprestano spreminjajo ter izboljšujejo. Predlagamo, da si pogledate orodje Javascript Obfuscator.

4.4.2 Optimizacija

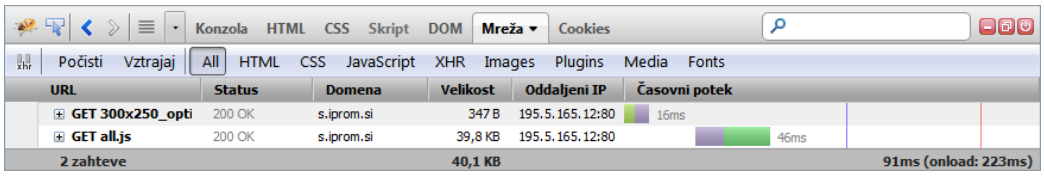
Prikazovanje oglasov, ki vsebujejo več izdelkov, je lahko zelo zahtevno z razvojnega vidika. Predpostavimo, da imamo oglas, zgrajen brez optimizacije po načrtu »Generiranje servirnih kod« iz poglavja 3.2.3. Za tak oglas bi bilo potrebno narediti vsaj 5 zahtevkov.



Slika 4.4: brez optimizacije zahtevkov

Slika 4.4 prikazuje časovni potek prenosa enega oglasa, ki je sestavljen iz več datotek. Takšen prenos traja v povprečju 153ms. Ideja, ki se tukaj pojavi, je, da bi lahko datoteke naložili na več različnih strežnikov in tako vzporedno naložili vse potrebne datoteke oglasa. Slednje bi razbremenilo strežnik in oglas bi se hitreje prikazal. Če nimamo na razpolago toliko strežnikov oz. CND, pa predlagamo optimizacijo združevanja. Optimizacija združevanja temelji na tem, da se zaradi latence posamezna datoteka velikosti X naloži hitreje kot več datotek velikosti Y, katerih skupna velikost je enaka skupni velikosti posamezne datoteke X. Optimizacijo združevanja smo preizkusili na oglasu, ki je enake velikosti in je prikazan na sliki 4.5. Dobili smo boljše rezultate.

Z optimizacijo združevanja smo izboljšali časovni dostop na91 ms, kar je 62ms hitrejše od prenosa vsake datoteke posebej. Omenjeni test smo opravili na istem strežniku z enako velikim oglasom. Opozorili bi, da smo tukaj sešteli vse zahtevke. Če bi hoteli videti samo zahtevke oglasov, bi morali pri sliki brez optimizacije(slika 4.4) in sliki z optimizacijo združevanja(slika 4.5)

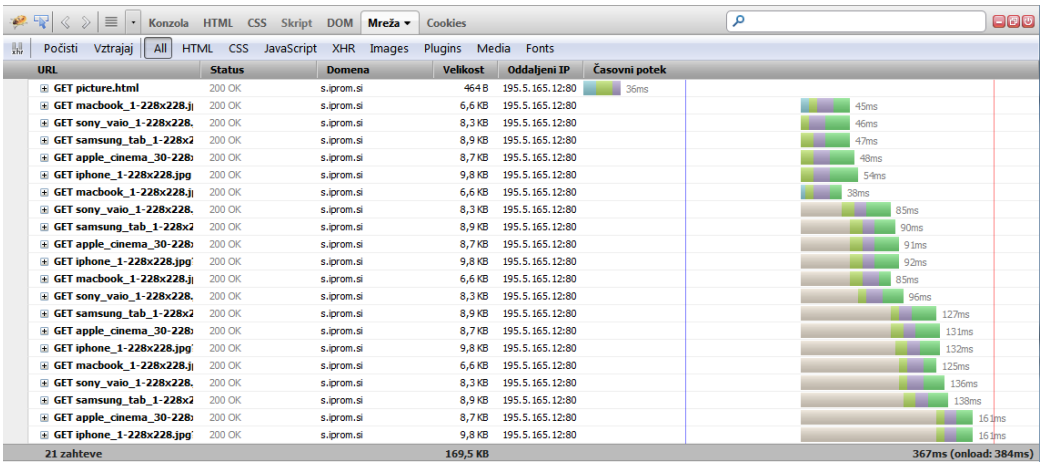


Slika 4.5: optimizacija zahtevkov

odšteti dostop do spletne strani.

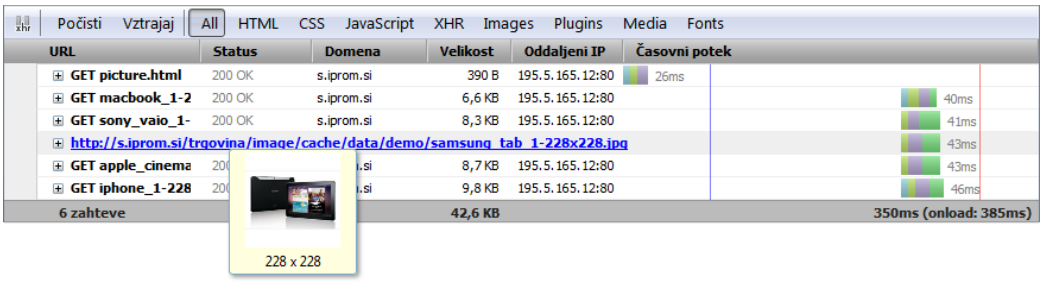
Tukaj se pojavi vprašanje, zakaj ne bi naredili preprosti vrstični izpis datoteke JavaScript na spletni strani in tako oglas prenesli s prvotnim zahtevkom. Odgovor na to je, da je zaradi zahtev po nadzorovanju oglasnih vzorcev, potrebno gostiti generirane datoteke na drugem strežniku. Omenjene generirane datoteke se dinamično spreminjajo na podlagi uporabnikovih zahtev v nadzornem sistemu in tako nenehno spreminjajo svojo vsebino, to pa povzroči spreminjanje oglasnega vzorca na spletni strani.

Naslednja zahteva po optimizaciji se pojavi s prenosom slik izdelkov v oglasni prostor. Če imamo v vzorcu 20 izdelkov, imamo tako posledično 20 zahtevkov za slike. Z zahtevki bremenimo strežnik in posledično se zvišuje latenca. Problem se stopnjuje, če hočemo istočasno prikazati tisoč instanc oglasa, kar posledično povzroči prenos 20 000 slik.



Slika 4.6: prenos slik iz strežnika

Problem je rešljiv z združevanjem ali s kodiranjem slik v zapis base64. Z združevanjem večih manjših slik sestavimo eno večjo sliko, ki pa ji spreminjamo odmik ozadja s CSS. Pravilen odmik od začetka združene slike, omejen na velikost slike, prikaže želeno sliko. S takšno optimizacijo pridobimo in zmanjšujemo skupno latenco.



Slika 4.7: prenos združene slike

Optimizacijo smo preizkusili še na ekstremnih primerih in preverili odzivnost strežnika(slika 4.8). V teh primerih se moremo zavedati, da že samo testiranje postane problematično in odzivnost brskalnika vprašljiva. Poudarili bi tudi, da je treba določiti pravilno zgornjo in spodnjo mejo slik v

oglasu. Visoka zgornja meja je smiselna samo v primeru, da oglas programiramo tako, da ima vgrajeno avtomatsko pomikanje slik in je uporabniku izpostavljen dlje časa. Če pa je oglas na mestu, kjer je osveževanje spletne strani pogosto, pa je smiselno imeti do pet izdelkov v oglasnem vzorcu, seveda v kombinaciji s prej omenjenim avtomatskim pomikanjem.

Slika 4.8: graf časovnih zahtevkov

Iz slike 4.8 razberemo, da se razlika začne kazati pri približno 25 izdelkih. Z večanjem števila izdelkov se v primeru »posameznih slik« začne časovna zahtevnost eksponentno povečevati. Združena slika oz. na grafu »skupinska slika« pa se povečuje linearno.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo uspešno predstavili in rešili problematiko serviranja in prenosa izdelkov iz spletne trgovine v oglasni prostor. Z nadzornim sistemom, ki črpa izdelke spletne trgovine in s katerim ustvarjamo oglasne vzorce, smo dosegli glavni cilj diplomskega dela. Z uporabo nadzornega sistema lahko namreč uporabnik sam prenaša izdelke in ustvarja nove oglasne vzorce, kar rešuje problematiko dodatnega strokovnega kadra in zmanjšuje stroške oglaševanja izdelkov. Dodana vrednost avtomatiziranega prenosa in ustvarjanja oglasnih vzorcev je statistični pregled prikazov in klikov, ki ga lahko uporabnik sam administrira in optimizira za prikazovanje posameznih izdelkov ter s tem pripomore k izboljšanju RPK-ja.

Uspešno smo načrtovali in realizirali vse tri dele avtomatiziranega sistema. Prvi del smo rešili z integracijo vtičnika v spletno trgovino. S črpanjem izdelkov in ustvarjanjem vzorcev smo zadovoljili potrebe drugega dela avtomatiziranega sistema. Z generiranjem servirnih kod in uspešno integracijo na spletno stran pa smo realizirali še tretji del sistema. Vsi trije deli so po uspešni povezavi uresničili idejo po avtomatskem prenosu in nadzoru, kot smo si jo zastavili v samem začetku diplomskega dela. Skozi izvedbo vseh komponent smo spoznali, da je za uspeh delovanja avtomatiziranega sistema ključnega pomena uskladitev vseh komponent in ne samo dobra realizacija

posameznega dela. Rezultat diplomskega dela je tako popolnoma delujoča spletna aplikacija, ki jo lahko preprosto namestimo na spletni strežnik in začnemo z njeno uporabo.

Avtomatizirani sistem je predvsem namenjen administratorjem spletnih trgovin in upraviteljem spletnih rešitev oz. medijskem načrtovalcem v različnih spletnih agencijah, ki imajo nadzor nad spletnimi trgovinami in prodajo izdelkov. Uporabniki nadzornega sistema lahko z njim prenašajo in izdelujejo oglasne vzorce po svoji meri. S preprostimi kliki lahko zamenjajo oglasni vzorec in ga shranijo. Slednje povzroči, da se na spletni strani znotraj oglasnega prostora zamenjajo izdelki. Takšna spletna aplikacija je orodje, s katerim lahko preprosto zamenjujejo izdelke iz več spletni trgovin na več spletnih straneh hkrati, kar med potekom oglaševalske kampanje.

Naš sistem oglaševanja izdelkov ni edini. Na spletu najdemo podobne sisteme drugih podjetij. Oglaševalski trg je trenutno zasičen z idejo o t. i. vnovičnem ciljanju oglasov. Aktualne aplikacije oz. podjetja, ki ponujajo tovrstne storitve, so Google, Criteo, Ad roll, Facebook itd. vendar pa omenjene platforme izvirajo iz drugačnih oglaševalskih trgov, kot je Slovenija, in imajo drugačne ideje o serviranju izdelkov. Ti namreč beležijo podatke o uporabnikih in na podlagi predhodno obiskanih strani oz. kriterijev predlagajo izdelke v oglasnem prostoru. Problemi, ki se pojavijo tukaj, so na primer povezani z vnašanjem izdelkov iz spletne trgovine v oglaševalske platforme. Uporabniki morajo v platforme vnesti oglase ali pa preko API vmesnika programirati vnose v sisteme, slednje pa zahteva veliko programerskega znanja in strokovno podkovanega osebja.

Možnosti nadaljevanja diplomskega dela vidimo predvsem v dveh smereh. Prva smer bi bila integracija algoritmov za izbiro izdelkov v oglasnem prostoru. V kolikor bi imeli veliko število izdelkov oz. veliko število spletnih strani, podprtih našo servirno kodo, bi na podlagi neke pametne izbire lahko

dosegali boljše rezultate. Trenutno je v naš avtomatiziran sistem mogoče vgraditi zgoraj omenjeni algoritem »vnovičnega ciljanja« ali pa kakšen drug primer algoritma iz vedenjskega vzorca. Zaradi omejevanja beleženja podatkov o uporabnikih pa vse bolj aktualni algoritmi prihajajo iz semantičnega ciljanja.

Druga smer je vgradnja finančnega sistema, s katerim bi določili vrednosti izdelkov v administrativnem delu in merili konverzije opravljenih nakupov v spletni trgovini. To bi lahko realizirali z dodatnim vgrajevanjem dogodkov v spletno košarico oziroma bi si lahko pot olajšali z vgradnjo analitičnega sistema Google Analytics. Podatke, pridobljene iz Google Analytics, bi kasneje črpali nazaj v administrativni del analitičnega sistema preko spletnega vmesnika.

Literatura

[1] M. Chon, "Agile Estimating and Planing", Prentice Hall Professional Technical Reference, 2006

[2] L. Černuta, "Uporaba informacijsko - komunikacijske tehnologije v gospodinjstvih in pri posameznikih", Statistični urad RS, 2013, dostopno na: https://www.stat.si/novica_prikazi.aspx?id=4240

[3] J. Lane, T. Barker, J. Lewis, M. Moscovitz, "Foundation Website Creation with HTML5, CSS3, and JavaScript", Springer Science+Business Media New York, 2012

[4] M. Zandstra, "PHP 5 Objects, Patterns, and Practice", Springer-Verlag New York, 2004

[5] P. Zdešar, "Uporaba interneta v gospodinjstvih in pri posameznikih", Statistični urad RS, 2011, dostopno na: https://www.stat.si/novica_prikazi.aspx?id=5795

[6] (2014) "IAB Display Advertising Guidelines", IAB, dostopno na: http://www.iab.net/guidelines/508676/508767/displayguidelines

[7] (2014) "Spletna trgovina", OpenCart, dostopno na: https://www.opencart.com

[8] (2014) "Universal Ad Package", IAB, dostopno na: http://www.iab.net/guidelines/508676/508767/UAP

[9] (2014) “Kohana framework”, Kohana framework, dostopno na: http://kohanaframework.org

[10] (2014) “What is Symfony”, Symfony, dostopno na: http://symfony.com/what-is-symfony

[11] (2014) “Phalcon documentation”, Phalcon, dostopno na: http://docs.phalconphp.com/en/latest/index.html

[12] (2014) “CakePHP documentation”, CakePHP, dostopno na: http://cakephp.org/documentation

[13] (2014) “jQuery library”, jQuery, dostopno na: http://jquery.com/

[14] (2014) “CodeIgniter Framework ”, CodeIgniter, dostopno na: http://ellislab.com/codeigniter

[15] (2014) “Google Hosted Libraries - Developer’s Guide”, Google Inc., dostopno na: https://developers.google.com/speed/libraries/devguide

[16] (2014) “Why Sencha”, Sencha Inc., dostopno na: http://www.sencha.com/products/

[17] (2014) “Model-view-controller approach”, Microsoft Corporation, dostopno na: http://msdn.microsoft.com/en-us/library/ff649643.aspx

[18] (2014) “What can PHP do”, PHP, dostopno na: http://www.php.net/manual/en/intro-whatcando.php

[19] (2014) “HTML and CSS”, W3C, dostopno na: http://www.w3.org/standards/webdesign/htmlcss

[20] (2014) “World Wide Web Consortium”, W3C, dostopno na: http://www.w3.org/standards/

[21] (2014) “Marketing Magazin - denarni tokovi”, Warc, dostopno na: http://www.marketingmagazin.si/novice/mmarketing/10665/ameriski-oglasevalci-lani-spletu-prvic-namenili-vec-denarja-kot-televiziji