UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Pavle Gartner

Primerjava različnih rešitev za izvedbo spletne trgovine

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00097/2011
Datum: 04.04.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **PAVLE GARTNER**

Naslov: **PRIMERJAVA RAZLIČNIH REŠITEV ZA IZVEDBO SPLETNE TRGOVINE**

THE COMPARISON OF DIFFERENT WEB SHOP IMPLEMENTATIONS

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

V diplomski nalogi predstavite sistematično primerjavo treh različnih izvedb spletne trgovine: z uporabo knjižnice Zend Framework, sistema za upravljanje vsebin Drupal ter sistema za elektronsko trgovanje Magenta. Po predstavitvi razvoja vseh treh izvedb najprej celovito zastavite kriterije, nato pa po njih izvedite primerjavo. Na koncu predstavite še odločitveno drevo za izbiro najustreznejšega načina glede na zahteve uporabnika.

Mentor:

viš. pred. dr. Igor Rožanc

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Pavle Gartner,

z vpisno številko 63060051,

sem avtor diplomskega dela z naslovom:

Primerjava različnih rešitev za izvedbo spletne trgovine

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca
- 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 Podpis avtorja:

Zahvala

Zahvaljujem se mentorju viš. pred. dr. Igorju Rožancu za strokovno pomoč.

Zahvaljujem se tudi staršem in punci, ki so me podpirali in bodrili pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic in simbolov

API - Application Programming Interface: skupek pravil in specifikacij programske opreme
ASP - Active Server Pages: Microsoftov skriptni jezik za pisanje dinamičnih spletnih strani
CGI - Common Gateway Interface: skupek pravil, ki določajo kako spletni strežnik komunicira z ostalo programsko opremo in obratno
CMS - Content Management System: sistem za upravljanje vsebin
CSS - Cascading Style Sheets: predloge za določanje izgleda spletnih strani
HTTP - Hypertext Transfer Protocol: omrežni protokol za porazdeljene informacijske sisteme
HTTPS - Hypertext Transfer Protocol Secure: varen omrežni protokol za porazdeljene informacijske sisteme
IMAP - Internet Message Access Protocol: spletni standardni protokol za prenos elektronske pošte
JDBC - Java Database Connectivity: standard za neodvisno povezovanje med podatkovnimi bazami in aplikacijami, ki ga podpira programski jezik Java
JMS - Java Message Service: API za pošiljanje sporočil med dvema klientoma, del platforme Java EE
JSON - JavaScript Object Notation: preprost format za prenos podatkov, ki temelji na podmnožici programskega jezika JavaScript
LDAP - Lightweight Directory Access Protocol: aplikacijski protokol za branje in urejanje imenikov preko IP omrežja
MVC - Model View Controller: tip arhitekture programske opreme
PDF - Portable Document Format: standardiziran format za izmenjavo elektronskih dokumentov
PHP - PHP Hypertext Processor: razširjen odprtokodni programski jezik
POP3 - Post Office Protocol 3: protokol za prejemanje elektronske pošte
Perl - Practical Extraction and Report Language: tolmačeni programski jezik
SEO - Search Engine Optimization: optimizacija za spletne iskalnike
SQL - Structured Query Language: strukturirani povpraševalni jezik za delo s podatkovnimi bazami
SVN - Subversion: protokol, ki označuje tudi sistem za sledenje spremembam izvorne kode, kateri omogoča sočasno delo na skupnem projektu
VPS - Virtual Private Server: virtualni privatni strežnik

Povzetek

Namen diplomske naloge je primerjava različnih izvedb spletne trgovine v programskem jeziku PHP. Na ta način se bo razvijalec, ki še nima veliko izkušenj na tem področju, lažje odločil, na kakšen način naj se loti razvoja poljubne aplikacije glede na zahteve naročnika, da jih bo pri razvoju lahko v čim večji meri zadovoljil. Primerjali smo izvedbe spletne trgovine Prodaja sestavljenih računalnikov. Izbrali smo tri načine izvedbe: na podlagi knjižnice Zend Framework, sistema za upravljanje vsebine Drupal ter sistema Magento Commerce, namenjenega izključno elektronskemu trgovanju. Zend Framework je odprtokodna PHP knjižnica za razvoj spletnih aplikacij in vsebuje bogat nabor komponent, ki omogočajo urejeno strukturo projekta, ter s tem hitrejši razvoj spletne strani. Drupal je odprtokodna spletna aplikacija za urejanje vsebine in gradnjo spletnih aplikacij, ki ima v jedru vkjučene funkcionalnosti za kreiranje spletnih strani z uporabniki, članki, blogi, komentarji in forumom. Magento Commerce je odprtokodni sistem za elektronsko trgovanje, ki uporabnikom omogoča prilagodljivost ter nadzor nad izgledom, vsebino in funkcionalnostim spletne trgovine.

Za primerjavo smo izbrali 10 kriterijev: čas realizacije, učno krivuljo, nadgradnje, prilagodljivost, vzdrževanje, podporo, hitrost, optimizacijo, obremenitev linije in plačljive opcije. Glede na primerjavo smo jim dodelil od 1 do 5 točk pri posameznih izvedbah ter jih na ta način rangirali.

Pri primerjavi smo ugotovili, da se načini izvedb močno razlikujejo med seboj, iz česar sledi da so primerni za različne situacije. Zato smo upoštevali ocene posameznih kriterijev primerjave glede na različne naročniške zahteve. S tem smo lahko priporočili določen način izvedbe. Na koncu smo naredili še odločitveno drevo, s katerim si lahko potencialni razvijalec pomaga pri izbiri načina izvedbe.

Ključne besede:

spletne trgovine, primerjalni kriterij, Zend Framework, Drupal, Magento Commerce, PHP, Eclipse PDT, XDebug, JMeter, FireBug, PageSpeed, webgrind

Abstract

The purpose of this thesis is comparison of different online shop implementations in PHP programming language. It will help developer, who lacks experiance in this area, to make a decision about the way of custom application development by taking customers' requirements into account. We will be comparing implementations of online shop Prodaja sestavljenih računalnikov. For comparison, we chose three ways of implementation: with Zend Framework library, Drupal CMS and Magento Commerce ecommerce system. Zend Framework is an opensource PHP library for web applications development and has rich features set, by which it supports ordered project structure and faster development. Drupal is an opensource CMS for building websites and has rich core features set like creation of dynamic websites, user accounts support, blogging, comments and forums. Magento Commerce is an opensource ecommerce system, which supports flexibility, control over apperance, content and functionalities of online shop.

For comparison, we chose 10 criteria: implementation time, learning curve, upgrades, flexibility, maintenance, online support, speed, optimization, bandwidth usage and commercial solutions. Implementations were rated from 1 to 5 points per criterion.

During the comparison it became obvious that implementations differ. That's why we used criteria rates according to customer requirements for recommending appropriate implementation. At the end we created a decision tree, which can help the potential developer with his decision for the way of implementation.

Key words:

online shops, comparison criterion, Zend Framework, Drupal, Magento Commerce, PHP, Eclipse PDT, XDebug, JMeter, FireBug, PageSpeed, webgrind

1. Uvod

Dandanes je na voljo veliko možnosti za izvedbo spletnih aplikacij, kar pa še ne pomeni, da je vsaka ustrezna, saj imajo naročniki različne zahteve, okus, finančna sredstva in rok izvedbe. Zato je zelo pomembno, da razvijalec ob začetku projekta izbere pravo rešitev za izvedbo oziroma lahko naročniku primerno svetuje glede izbire načina izvedbe. Če je razvijalec že seznanjen z določeno knjižnico ali pa ima za izvedbo različnih spletnih strani na voljo CMS, ki ga obvlada, je izbira ponavadi precej očitna. Kljub temu je potrebno nekje začeti, zato bomo rešitve primerjali s predpostavko, da razvijalec še ne pozna nobenega od načinov.

1.1 Razčlenitev tipov izvedb

Za tip spletne aplikacije smo si izbrali spletno trgovino, saj imamo na tem področju že nekaj izkušenj, poleg tega pa se nam zdi pomembno, da za raziskavo ne izberemo preveč preproste spletne strani ali bloga, temveč aplikacijo, ki ima dejansko tudi kompleksno funkcionalnost. Spletne trgovine smo realizirali v skriptnem jeziku PHP. Ker je za realizacijo na voljo več pristopov (od izvedbe iz nič do že obstoječih rešitev, namenjenih izključno spletnim trgovinam) smo le-te razdelili v tri skupine:

- rešitve po meri na podlagi knjižnice (Zend Framework [48], Cake PHP [3], Symfony [54], CodeIgniter [4],...),
- rešitve na podlagi obstoječega CMS-ja (Drupal [17], Joomla [19], WordPress [62], Typo3 [56],...),
- rešitve namenjene izključno elektronskemu trgovanju (Magento Commerce [26], Zen Cart [66], osCommerce [39],...).

Ker že primerjava najbolj priljubljenih rešitev presega obseg te diplomske naloge, smo izbrali po enega predstavnika iz vsake skupine:

- Zend Framework knjižnica in nekaj dodatnih knjižnic (Html Purifier [15], ExtJS [13], TinyMCE [55]) kot rešitev po meri,
- Drupal 6 platforma v kombinaciji z Ubercart modulom [21] kot rešitev na podlagi sistema za upravljanje vsebin in
- Magento Commerce kot rešitev, namenjena izključno elektronskemu trgovanju (brezplačna različica, brez plačljivih razširitev).

1.2 Funkcionalnost spletne trgovine

Kot primer spletne trgovine smo izbrali seminarsko nalogo Prodaja Sestavljenih Računalnikov, ki je bila sicer ena izmed nalog pri predmetu Razvoj Programskih Sistemov 2. Naročnik projekta je podjetje, ki prodaja in sestavlja računalnike. Dele za računalnike nakupujejo na veliko in jih potem uporabijo za sestavo posameznega računalnika. Do sedaj so za beleženje delov, ki so na voljo, uporabljali evidenco na papirju, sedaj pa želijo svoje poslovanje izboljšati z uporabo spletnega portala, ki vsebuje 4 različne tipe uporabnikov: trije administrativni (skrbnik, skladiščnik, sestavljalec) ter stranka.

Posamezen računalnik je sestavljen iz več delov. Stranka lahko sestavi računalnik samo, če so na voljo vsi potrebni deli. Nakupe bodo stranke naročnika še vedno opravljale osebno, vendar morajo biti obveščene, ali je v danem trenutku določen računalnik možno sestaviti. Zagotoviti je potrebno tudi funkcionalnost, ki podpira 10% maržo na ceno sestavnih delov. Poleg tega želi naročnik še naslednje funkcionalnosti, razdeljene po tipih uporabnikov:

skrbnik:

- urejanje administrativnih uporabnikov (dodajanje, spreminjanje, brisanje),
- urejanje skladišča, ki vsebuje urejanje tipov sestavnih delov in urejanje sestavnih delov (dodajanje, brisanje, sprememba količine),
- sestavljanje računalnikov glede na komponente, ki so na voljo v skladišču,
- pregled naročil strank (za vsak računalnik posebej cena, sestavni deli, ime, priimek in e-mail stranke),

skladiščnik:

- urejanje sestavnih delov, ki so že na voljo v skladišču,

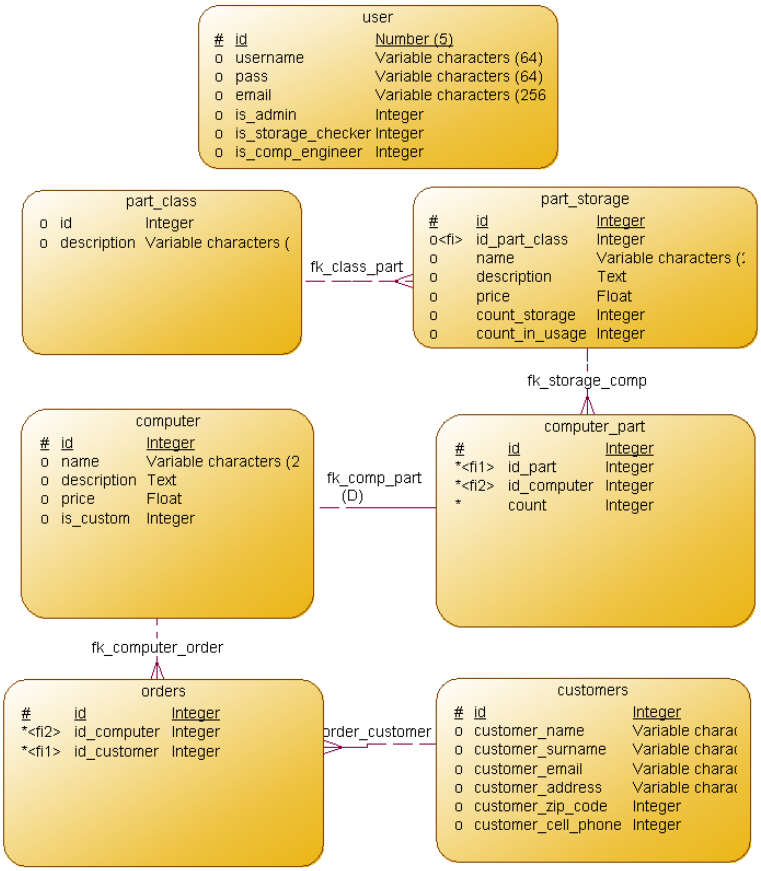
sestavljalec:

- sestavljanje računalnikov glede na komponente, ki so na voljo v skladišču (ista funkcionalnost kot pri skrbniku strani),

stranka:

- pregled in dodajanje že sestavljenih računalnikov v košarico,
- sestavljanje poljubnih računalnikov iz komponent, ki so na voljo v skladišču,
- pregled in urejanje košarice ter potrditev naročila s predhodnim vnosom osebnih podatkov stranke.

1.3 Logični podatkovni model



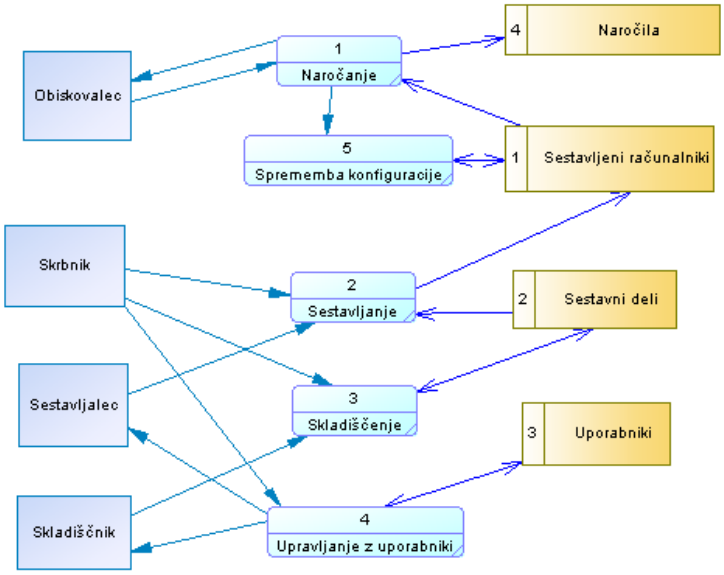
Slika 1.1: Logični podatkovni model.

Za projekt smo izdelali logični podatkovni model [25], prikazan na sliki 1.1. Uporabili smo ga pri izvedbi na podlagi knjižnice Zend Framework, saj smo podatkovno bazo načrtovali sami, medtem ko je pri ostalih dveh implementacijah podatkovna baza že zraven, tako da nam je bil samo v dodatno pomoč pri načrtovanju izvedbe. Vsebuje naslednje entitete:

- `user`: tabela uporabnikov,
- `part_class`: tabela tipov sestavnih delov (matična plošča, grafična kartica, napajalnik ...),
- `part_storage`: tabela sestavnih delov v skladišču (vsebuje ime, opis, ceno, št. kosov v skladišču, št. kosov v računalnikih, ki so naprodaj),
- `computer_part`: tabela predstavlja relacijo med tabelo `part_storage` ter `computer` (v njej so podatki o tem, katere sestavne dele vsebuje določen računalnik),

- `computer`: tabela vsebuje računalnike, ki so na voljo strankam spletne strani, poleg tega pa tudi računalnike, ki jih bodo stranke same skonfigurirale,
- `orders`: tabela vsebuje naročila strank,
- `customers`: tabela vsebuje podatke o strankah.

1.4 Diagram podatkovnih tokov



Slika 1.2: Diagram podatkovnih tokov.

Izdelali smo tudi globalni diagram podatkovnih tokov [6], ki je prikazan na sliki 1.2 in vsebuje:

- 5 procesov:
  - `Naročanje`: stranka tu dodaja računalnike v košarico, zaključi naročilo, prav tako pa lahko dostopa do procesa spremembe konfiguracije,
  - `Sprememba konfiguracije`: stranka lahko tu spreminja sestavne dele računalnikov, ki so na voljo,
  - `Sestavljanje`: sestavljalci in skrbniki lahko tu dodajajo, urejajo in brišejo posamezne sestavljene računalnike,
  - `Skladiščenje`: skladiščnik in skrbnik lahko tu upravljata s sestavnimi deli,

  - `Upravljanje z uporabniki`: skrbnik lahko tu dodaja, ureja in briše uporabnike. Poleg tega ta proces skrbi za prijavo in odjavo uporabnikov,

- 4 podatkovne strukture:
  - `Sestavljeni računalniki`,
  - `Sestavni deli`,
  - `Uporabniki`,
  - `Naročila`,
- 4 entitete:
  - `Stranka`,
  - `Skrbnik`,
  - `Sestavljalec` in
  - `Skladiščnik`.

1.5 Kriteriji primerjave

Pri primerjavi načinov izvedbe je za dobro oceno posameznih rešitev in pridobitev uporabnih rezultatov zelo pomembna izbira primernih kriterijev. Odločili smo se za točkovanje vsakega izmed kriterijev z oceno med 1 in 5 točk za vsako izmed izvedb glede na medsebojno primerjavo. Po raziskavi na spletu [47, 45] in tehtnem razmisleku med izvedbo rešitev smo sestavili seznam 10 kriterijev:

- čas realizacije,
- učna krivulja,
- nadgradnje,
- prilagodljivost,
- vzdrževanje,
- spletna podpora,
- hitrost,
- obremenitev linije,
- optimizacija in
- plačljive opcije.

Na seznamu kriterijev sta se znašla tudi optimizacija spletnih strani za iskalnike (SEO) ter analiza varnosti. Vendar sta to preveč zahtevna kriterija, za katera bi lahko napisali samostojni diplomski nalogi, še posebno glede na to, da bi morali izvajati teste varnosti ter optimizacijo za iskalnike na treh izvedbah.

1.6 Namen in cilj diplomske naloge

Namen diplomske naloge je:

- opisati vsakega izmed načinov izvedbe,
- predstaviti kriterije za medsebojno primerjavo posameznih izvedb,
- primerjati posamezne načine izvedb po kriterijih ter
- analizirati dobljene rezultate.

Cilj diplomske naloge je:

- bralcu predstaviti različne možnosti za izvedbo spletne trgovine,
- ugotoviti prednosti in slabosti posameznih izvedb,
- uporabiti dobljene rezultate analize za izdelavo napotkov, ki bodo služili kot pomoč oziroma referenca pri izbiri načina izvedbe glede na različne zahteve naročnikov.

2. Predstavitev posameznih načinov izvedbe

2.1 Definicije pogostih pojmov

2.1.1 Izvedba

Izvedba je ena od faz razvoja informacijskega sistema, ki zajema uresničitev podrobnega načrta računalniškega programa ali informacijskega sistema, namestitev uporabniškega programa, prenos podatkov ter usposobitev uporabnikov [16].

2.1.2 Skriptni jezik

Skriptni jezik je programski jezik, ki se uporablja za upravljanje z aplikacijami. Skripte se razlikujejo od jedra aplikacije in so ponavadi napisane v drugem jeziku, občasno tudi s strani končnega uporabnika. Skripte so pogosto interpretirane na podlagi izvorne kode oziroma binarne predstavitve izvršnega programa, pri katerem so aplikacije tipično najprej prevedene v strojno kodo [50].

2.1.3 Knjižnica

Knjižnica je v računalništvu zbirka podprogramov (oziroma funkcij) za pomoč pri izdelavi oziroma razvoju programske opreme. Knjižnice vsebujejo kodo in podatke, ki se jih da uporabiti v neodvisnih programih. Zaradi tega se programi izmenjujejo in spreminjajo modularno. Neketere izvršne datoteke so lahko oboje, samostojni programi ali knjižnice, vendar večina knjižnic ni izvršljivih [22].

2.1.4 PHP

PHP je razširjen odprtokodni programski jezik, ki se uporablja za strežniške uporabe oziroma za razvoj dinamičnih spletnih vsebin. Lahko ga primerjamo z Microsoftovim sistemom ASP, VBScript in JScript, Sun Microsystemovim sistemom JSP in Java ter sistemom CGI in Perl. Podoben je običajno strukturiranim programskim jezikom (najbolj jezikoma C in Perl) in izkušenim programerjem dovoljuje razvijanje zapletenih uporab brez dolgega učenja. Trenutno sta v uporabi dve različici: 5.3.x in 5.2.x.

2.1.5 MySQL

MySQL je sistem za upravljanje s podatkovnimi bazami. MySQL je odprtokodna izvedba relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL. MySQL deluje na principu odjemalec - strežnik, pri čemer lahko strežnik namestimo kot sistem, porazdeljen na več strežnikov. Obstaja veliko število odjemalcev, zbirk ukazov in programskih vmesnikov za dostop do podatkovne baze MySQL [28].

2.1.6 JavaScript

JavaScript je objektni skriptni programski jezik, ki ga je razvil Netscape, da bi spletnim programerjem pomagal pri ustvarjanju interaktivnih spletnih strani. Jezik je bil razvit neodvisno od Jave, vendar si z njo deli številne lastnosti in strukture. JavaScript lahko sodeluje s HTML-kodo in s tem poživi stran z dinamičnim izvajanjem. JavaScript podpirajo velika programska podjetja in ga kot odprt jezik lahko uporablja vsakdo, ne da bi pri tem potreboval licenco. Podpirajo ga vsi novejši spletni brskalniki. Trenutna različica je JavaScript 1.6, ki ustreza specifikaciji ECMA-262 Edition 3 [18].

2.1.7 Sistem za upravljanje vsebin

Sistem za upravljanje vsebin je sistem, ki omogoča urejanje in vzdrževanje vsebine spletnih strani brez znanja označevalnega jezika HTML. Urednik spletne strani lahko tako samostojno spreminja besedila, slike in druge elemente spletne strani brez pomoči podjetja ali osebe, ki je stran izdelala [49].

2.1.8 Elektronsko trgovanje

Elektronsko trgovanje je nakupovanje in prodajanje izdelkov oz. storitev preko elektronskega sistema, kot je internet ali pa računalniško omrežje. Vendar pa ima izraz tudi širši pomen, saj služi razvoju, marketingu, dostavi ipd. Od začetka množične uporabe interneta dalje je vedno bolj priljubljen način trgovanja [12].

2.2 Izvedba na podlagi knjižnice Zend Framework

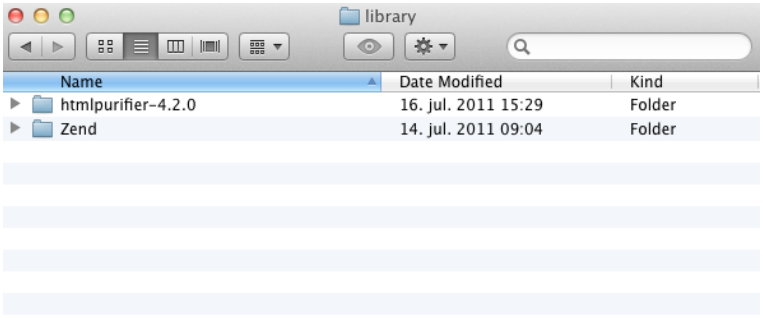
2.2.1 O knjižnici

PHP je v uporabi za izvedbo dinamičnih spletnih strani že skoraj 15 let. Najprej se je uporabljal za precej preproste spletne strani, saj je bila skripta vsebovana kar znotraj HTML strani. Vendar pa je z novimi nadgradnjami različic neizogibno pisanje večjih aplikacij z uporabo tega skriptnega jezika. Prav tako je postalo očitno, da mešanje HTML-ja in PHP kode ne mora biti dolgoročna rešitev za izvedbo večjih spletnih strani. Problem nastane pri vzdrževanju in razširljivosti, saj se kombinacija PHP-ja in HTML-ja hitro odziva, vendar po drugi strani otežuje osveževanje in razširitev spletnih strani. Ravno zaradi tega je prišlo do razvoja Zend Framework knjižnice, ki zagotavlja dolgoročen razvoj PHP spletnih strani s poenostavitvijo vzdrževanja in razširitev [48].

Zend Framework je odprtokodna PHP knjižnica za razvoj PHP spletnih aplikacij. Vsebuje bogat nabor komponent, ki podpirajo vse od MVC modela do generiranja PDF dokumentov, preko katerih omogoča urejeno strukturo projekta ter s tem hitrejši razvoj spletne strani [48].

2.2.2 Zahteve in namestitev

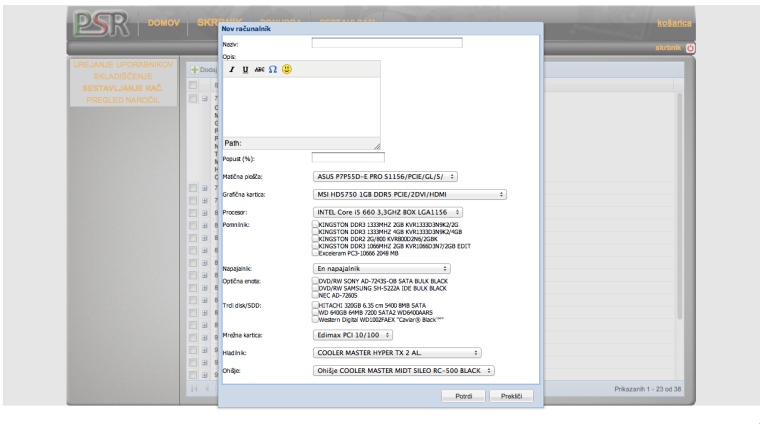
Zend Framework za delovanje potrebuje strežnik s podporo PHP 5 tolmačenja, poleg tega pa je priporočena uporaba PHP različice 5.2.4 ali višje. Nekatere funkcije zahtevajo dodatne razširitve na strežniku, npr. `mod_rewrite` omogoča podporo olepšanih URL-jev, s katerimi lahko dosežemo boljšo optimizacijo za spletne brskalnike [57]. Knjižnica preko vmesnika `Zend_Db` podpira uporabo podatkovnih baz IBM DB2, MariaDB, MySQL, Microsoft SQL Server, Oracle, PostgreSQL in SQLite [59]. Za namestitev je privzeto potrebno prepisati direktorij Zend znotraj direktorija `library`, ki se nahaja v korenskem direktoriju spletne strani, definiranem v konfiguraciji strežnika. Prikaz namestitve je na sliki 2.1.



Slika 2.1: Namestitev knjižnice Zend Framework.

2.2.3 Izvedba

Za izvedbo smo uporabili različico 1.11.4., ki smo jo med primerjavo nadgradili, poleg tega pa še tudi PHP knjižnico Html Purifier 4.2.0 [15], ki služi filtriranju poizvedb spletnih obrazcev za zaščito pred vrinjeni zlonamerni kodi, napisani v skriptnem jeziku. Za administrativne funkcionalnosti (slika 2.2) smo uporabili Javascript knjižnici ExtJS 3.3.1 [13] in TinyMCE 3.2.7 [55]. Omeniti velja tudi to, da je ExtJS za komercialno uporabo plačljiva knjižnica.

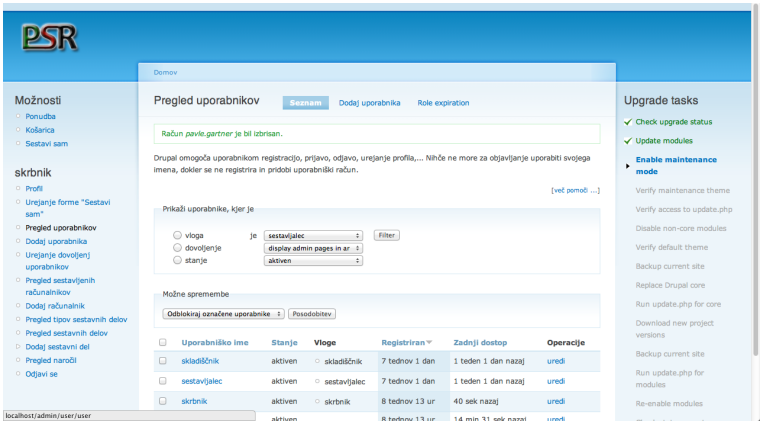


Slika 2.2: ExtJs in TinyMCE v akciji.

2.3 Izvedba na podlagi sistema za upravljanje vsebine Drupal 6

2.3.1 O sistemu

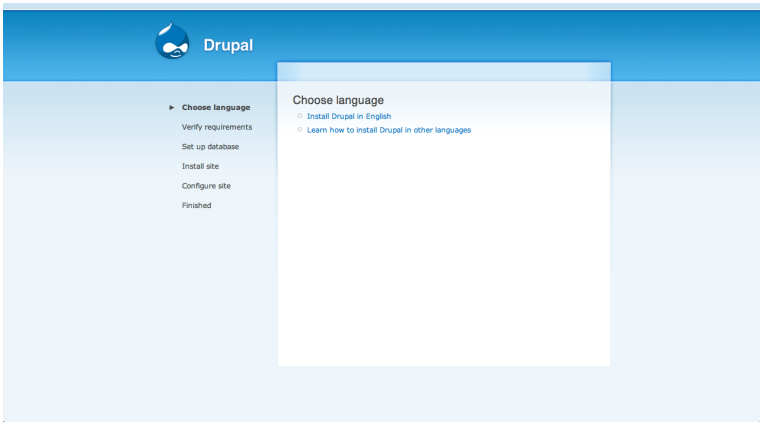
Drupal je odprtokodna spletna aplikacija za urejanje vsebine in gradnjo spletnih aplikacij. V jedru ima že vkjučene funkcionalnosti za kreiranje spletnih strani z uporabniki, članki, blogi, komentarji in forumom. Z dodajanjem modulov se lahko spletna stran prelevi v spletno trgovino, galerijo fotografij in še marsikaj. Ker ima modularno zasnovo, omogoča hitro in preprosto razširljivost ter pisanje poljubnih funkcionalnosti [17]. Vsebuje čelni del sistema, ki ga vidijo potencialni kupci, ter zaledje za administrativna opravila, ki je prikazano na sliki 2.3.



Slika 2.3: Zaledje sistema za upravljanje vsebin Drupal 6.

2.3.2 Zahteve in namestitev

Drupal za delovanje potrebuje spletni strežnik s podporo PHP tolmačenju. Priporočena različica PHP-ja za Drupal 5 in 6 je vsaj 5.2, za Drupal 7 pa vsaj 5.3, pri čemer Drupal 5 ne podpira PHP različice 5.3. Za čim večjo kompatibilnost je priporočena uporaba podatkovne baze MySQL, podporo pa ima z določenimi omejitvami glede kompatibilnosti tudi za PostgreSQL, SQLite ter preko uporabe posebnega modula za Oracle in Microsoft SQL Server. Za namestitev je potrebno kopirati direktorij s sistemom Drupal v korenski direktorij spletnega strežnika. Preostanek namestitve sistema poteka v spletnem brskalniku po korakih [65], kar je vidno na sliki 2.4.



Slika 2.4: Namestitev sistema za upravljanje vsebin Drupal 6.

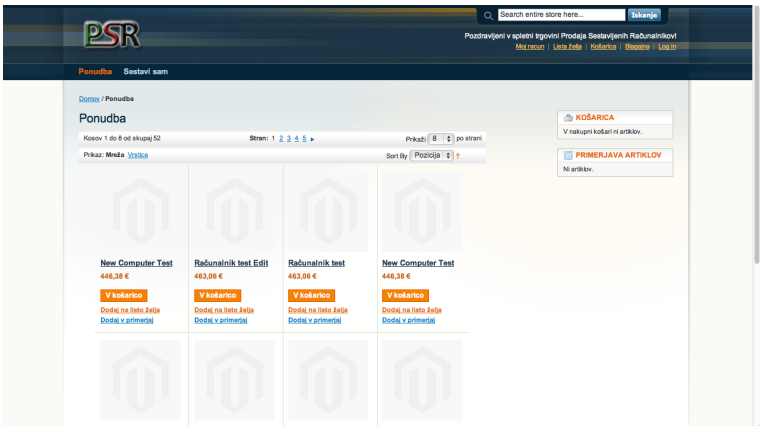
2.3.3 Izvedba

Za izvedbo smo uporabili različico 6.22 in ne 7, saj je Ubercart 3.0 modul za Drupal 7 še v testni različici. Ubercart je odprtokodni modul in podpira konfiguracijo ponudbe izdelkov in njihove zaloge, urejanje atributov, konfiguracijo postopka naročila, različne plačilne sisteme, itd. [21]. Uporabili smo različico 2.4 za Drupal 6 ter množico dodatnih modulov za podporo potrebnih funkcionalnosti, in sicer modul `smtp` namenjen konfiguraciji strežnika za pošiljanje elektronske pošte, `stringoverrides` za lokalizacijo itd.

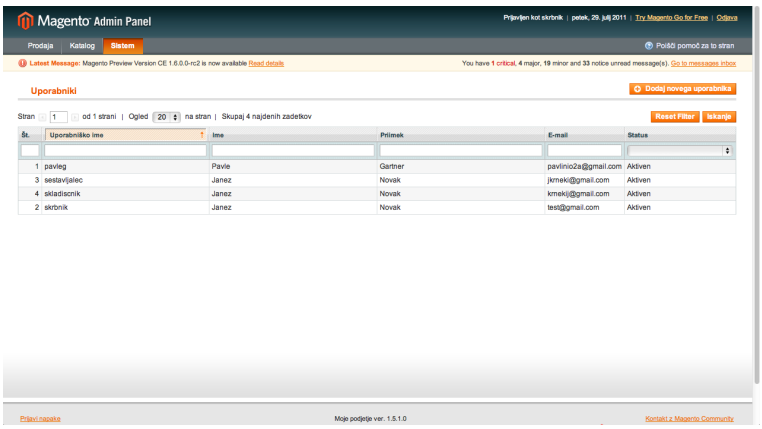
2.4 Izvedba na podlagi sistema za elektronsko trgovanje Magento Commerce

2.4.1 O sistemu

Magento Commerce je odprtokodni sistem za elektronsko trgovanje, ki uporabnikom omogoča prilagodljivost ter nadzor nad izgledom, vsebino in funkcionalnostjo spletne trgovine. Preko intuitivnega uporabniškega vmesnika lahko razvijalec upravlja z marketingom, optimizacijo za spletne brskalnike in urejanjem kataloga ponudbe ter s tem omogoča izdelavo spletne trgovine, ki ustreza potrebam naročnika. Vsebuje čelni del sistema (slika 2.5), ki ga vidijo potencialni kupci, ter zaledje za administrativna opravila (slika 2.6), ki je potencialnim kupcem skrito [26].



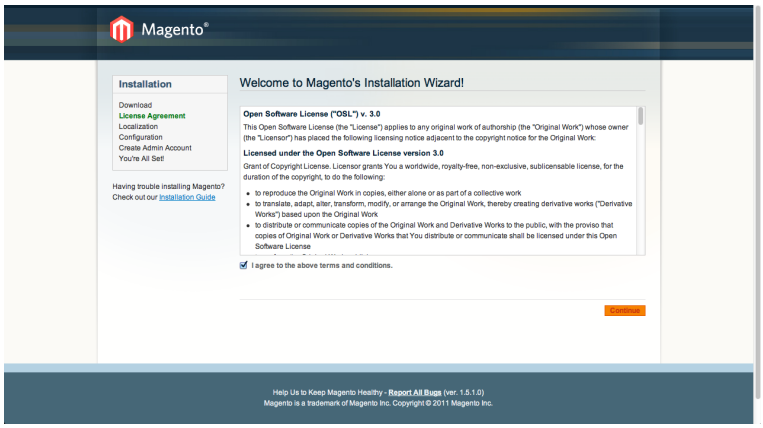
Slika 2.5: Čelni del sistema za elektronsko trgovanje Magento Commerce.



Slika 2.6: Zaledje sistema za elektronsko trgovanje Magento Commerce.

2.4.2 Zahteve in namestitev

Magento za delovanje potrebuje spletni strežnik s podporo PHP tolmačenju v različici 5.2.13 ali višji različici. Edina podprta podatkovna baza je MySQL 4.1.20 oziroma novejši, poleg tega pa zahteva še pogon hrambe InnoDB. Za namestitev je potrebno kopirati direktorij s sistemom Magento v korenski direktorij spletnega strežnika. Preostanek namestitve sistema poteka v spletnem brskalniku po korakih, [64] kar vidimo na sliki 2.7.



Slika 2.7: Namestitev sistema za elektronsko trgovanje Magento Commerce.

2.4.3 Izvedba

Za izvedbo smo uporabili različico 1.5.1.0. Dodatne razširitve niso prišle v poštev, saj so bile vse, ki bi bile primerne za izvedbo funkcionalnosti izven jedra, plačljive.

3. Uporabljena strojna in programska oprema

3.1 Platforma

Za izvedbo in primerjavo vseh treh različic spletne trgovine smo uporabili prenosni računalnik MacBook Pro 13", Mid 2010 na sliki 3.1, ki ima naslednje specifikacije:

- procesor 2,4 GHz Intel Core 2 Duo,
- 4 GB 1067 Mhz DDR3 pomnilnika,
- grafično kartico NVIDIA GeForce 320M 256 MB in
- SATA trdi disk TOSHIBA MK2555GSXF.

Operacijski sistem, na katerem smo izvedli primerjalne teste, je bil Mac OS X Snow Leopard 10.6.7.

Slika 3.1: MacBook Pro 13", Mid 2010.

3.2 Programska oprema

3.2.1 XAMPP

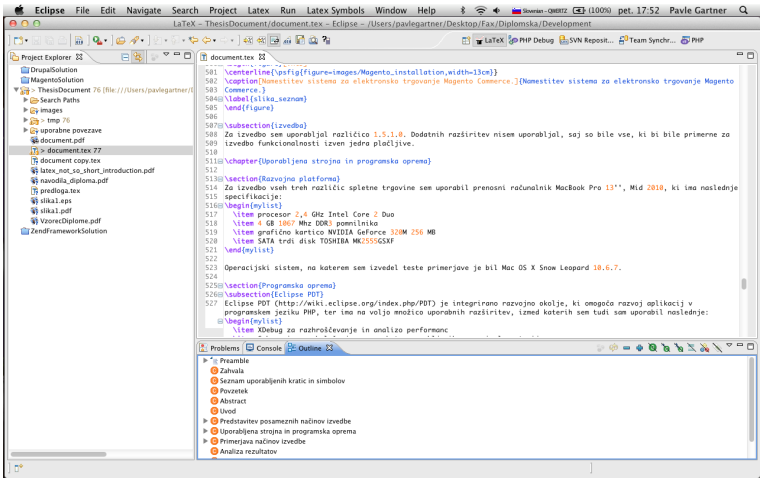
Za gostovanje smo uporabili rešitev XAMPP 1.7.3 za Mac OS X, ki vsebuje naslednje komponente: Apache 2.2.14, MySQL 5.1.44, PHP 5.3.1, Perl 5.10.1, ProFTPD 1.3.3, phpMyAdmin 3.2.4, OpenSSL 0.9.8k, GD 2.0.35, Freetype 2.3.5, libjpeg 6b, libpng 1.2.32, libungif-4.1.4, zlib 1.2.3, expat 2.0.1, Ming 0.4.2, Webalizer 2.01-10, pdf class 009e, mod_perl 2.0.4, SQLite 3.6.3, gdbm-1.8.3, libxml-2.7.2, libxslt-1.1.24, openldap-2.3.43, imap-2004g, gettext-0.16.1, libmcrypt-2.5.8, mhash-0.9.9, zziplib-0.13.48, bzip2-1.0.5, freetds-0.64 [63].

3.2.2 Eclipse PDT

Eclipse PDT [42] je integrirano razvojno okolje (slika 3.2), ki omogoča razvoj aplikacij v programskem jeziku PHP in je na voljo za operacijske sisteme Windows, Linux ter OS X. Na voljo ima množico uporabnih razširitev in funkcionalnosti, izmed katerih smo uporabili naslednje:

- XDebug za razhroščevanje in analizo performans,
- Subversion klient za beleženje sprememb ter porabljenih ur za izvedbo na lokalni SVN strežnik,
- Texlipse za pisanje končnega izdelka.

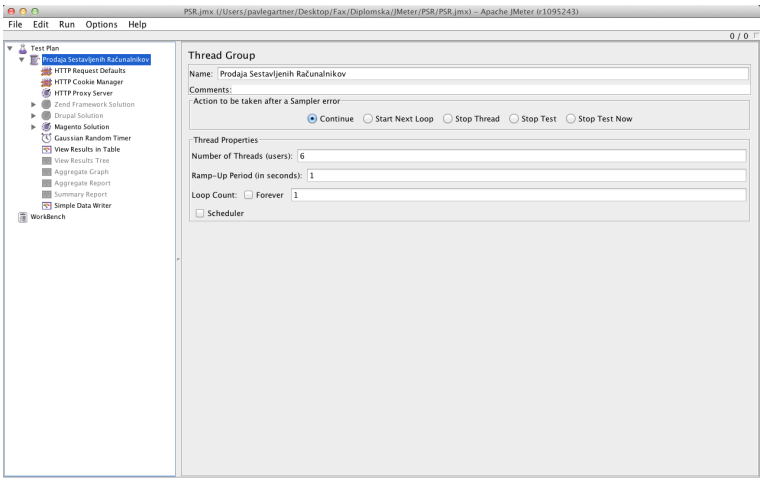
Na ta način smo imeli v enem orodju združenih več uporabnih funkcionalnosti.



Slika 3.2: Integrirano razvojno okolje Eclipse PDT.

3.2.3 Apache JMeter

Apache JMeter (slika 3.3) je brezplačna odprtokodna namizna javanska aplikacija za izvajanje avtomatiziranih performančnih in obremenitvenih testov. Tipično merimo in analiziramo dobljene podatke aplikacij tipa odjemalec-strežnik. Testno orodje JMeter ima možnosti za testiranje različnih tipov strežnikov, in sicer LDAP, POP3, IMAP, SOAP, JMS, JDBC, HTTP in HTTPS. Uporabili smo jo za primerjavo hitrosti in obremenitve linije s pomočjo avtomatiziranih poizvedb za vsako izmed spletnih trgovin [1].



Slika 3.3: Apache JMeter.

3.2.4 Page Speed

Page Speed je odprtokodni projekt, s katerim je začel Google, in je namenjen razvijalcem za optimizacijo spletnih strani. Na voljo je kot razširitev za brskalnika Google Chrome in Firefox ter omogoča oceno hitrosti spletne strani, hkrati pa razvijalcu predlaga možne izboljšave, s katerimi bi izboljšal hitrost nalaganja spletne strani [41].

3.2.5 Webgrind

Webgrind je čelni del za prikaz rezultatov, pridobljenih z orodjem za analizo performans XDebug, in je napisan v programskem jeziku PHP 5. Vsebuje podmnožico funkcionalnosti orodja kcachegrind [20], uporabljamo ga lahko znotraj spletnega brskalnika neodvisno od platforme [61].

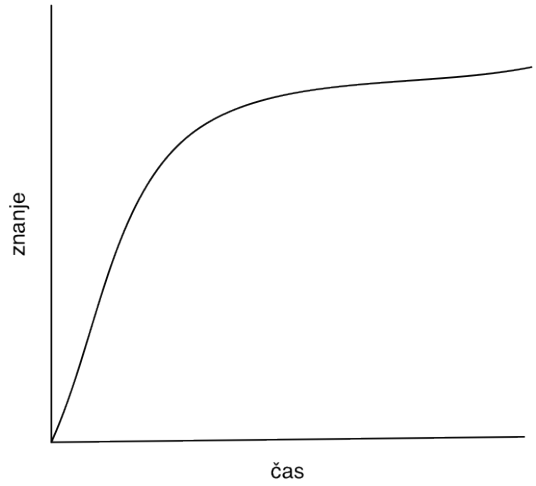
4. Primerjava načinov izvedbe po kriterijih

4.1 Krivulja učenja

Za vsako izvedbo smo določili krivuljo učenja, ki predstavlja razmerje med porabljenim časom in pridobljenim znanjem o posameznem načinu izvedbe [58].

4.1.1 Zend Framework

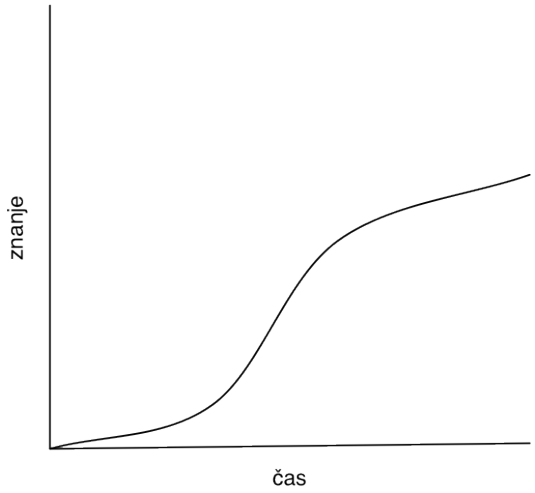
Znanje za izvedbo želenih funkcionalnosti smo pridobili kar iz spletne dokumentacije knjižnice ZendFramework, vodičev in raznih forumov [37, 60, 53]. Zend Framework je dokaj kompleksna knjižnica, saj smo porabili skoraj teden dni, smo razumeli princip delovanja uporabljenih komponent [23]. Vendar nato izvedba z vidika uporabe knjižnice ni bila več problematična. Edine stvari, ki se jih je potrebno učiti sproti, so podrobnosti uporabe posameznih komponent. Na ta način je dobljena krivulja na sliki 4.1 najprej strma, nato pa s časom postane bolj položna. **Število točk: 4,5**.



Slika 4.1: Krivulja učenja za knjižnico Zend Framework.

4.1.2 Drupal

Tudi pri izvedbi v sistemu za urejanje vsebin Drupal smo potrebno znanje pridobili kar iz spletnih virov: iz uradne dokumentacije [7] ter za kompleksnejše posege iz raznih forumov in vodičev [8]. Za učenje osnov smo porabili približno en dan. Začetek izvedbe je bil dokaj enostaven, saj je mnogo funkcionalnosti podprtih z raznimi moduli. Ob izvedbi bolj kompleksnih funkcionalnosti, ki so zahtevale pisanje kode, pa smo se morali poglobiti v izvedbo določenih modulov. Ko smo enkrat razumeli princip delovanja, izvedba z vidika CMS-ja ni bila več problematična. Kljub temu je potrebno med izvedbo zaradi potrebnih dodelav proučiti določene funkcionalnosti jedra zaradi potrebnih dodelav. Na ta način je dobljena krivulja na sliki 4.2 v začetku položna, ko pride do učenja izvedbe poljubnih modulov postane malo bolj strma in na koncu spet položna. **Število točk: 4**.

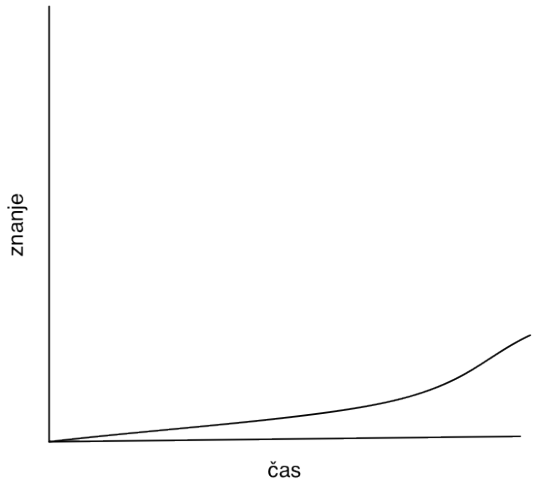


Slika 4.2: Krivulja učenja sistema za upravljanje vsebin Drupal.

4.1.3 Magento

Magento ima za osnove in kompleksnejše probleme na voljo Magento Wiki [27], iz katerega smo dobili vse potrebne podatke za osnovno izvedbo. Tudi za izvedbo poljubnih funkcionalosti je na voljo pomoč, ki pride zelo prav, saj je velika večina uporabnih modulov plačljiva, kar pomeni, da je potrebno marsikaj, kar ni na voljo v jedru implementirati po svoje. To pa ni lahko, saj je Magento zelo kompleksen sistem. Za učenje osnov smo porabili približno tri dni. Podobno kot pri Drupalu je bil začetek relativno preprost, vendar pa kasneje težavnost naraste in zato učenje počasneje napreduje. Že sama izvedba preprostih stvari, kot je na primer odstranjevanje prednastavljenega oglasnega bloka, je zahtevna in časovno potratna, kar je razvidno tudi iz tabele

za porabljen čas realizacije `Ponudbe`. Temu primerna je krivulja na sliki 4.3, ki je najprej položna nato pa začne zelo počasi naraščati, kar kaže na veliko porabo časa pri izvedbi preprostih funkcionalnosti in s tem počasno učenje. **Število točk: 3**.



Slika 4.3: Krivulja učenja sistema za elektronsko trgovanje Magento.

4.2 Čas realizacije

Čas realizacije smo dobili tako, da smo vsako spremembo v kodi poslali na lokalni SVN strežnik ter zapisali v komentar porabljen čas. V seštevku nismo upoštevali načrtovanja spletne strani na podlagi knjižnice Zend Framework, saj je to delo grafičnega oblikovalca in ne razvijalca. Za Drupal in Magento smo na tem delu uporabili prednastavljene teme.

4.2.1 Zend Framework

Uporabnik	Funkcionalnost	Čas	Opomba
Skrbnik	Urejanje uporabnikov	5h	
	Urejanje skladišča	25h	
	Sestavljanje računalnikov	30h	
	Pregled naročil	4h	
Skladiščnik	Urejanje skladišča	2h	Uporabljena logika skrbnika
Sestavljalec	Sestavljanje računalnikov	2h	Uporabljena logika skrbnika
Stranka	Ponudba	2,5h	
	Košarica	4,5h	
	Naročilo	6h	
	Sestavljanje poljubnih računalnikov	2,5h	Uporabljena logika skrbnika

Tabela 4.1: Poraba časa po funkcionalnostih uporabnikov za realizacijo s knjižnico Zend Framework.

Glede na to, da je potrebno praktično vse napisati iz nič (tukaj je poudarek predvsem na skrbniških funkcionalnostih), je izvedba z Zend Framework knjižnico časovno dokaj potratna. Skupno število porabljenih ur iz SVN komentarjev je kar 83,5, kar je velik minus, saj je običajno potrebna čim hitrejša izvedba. Poleg tega je pri izvedbi na podlagi knjižnice večja možnost napak, zato je načeloma potrebnega več testiranja. Ker je poraba časa občutno višja kot pri ostalih dveh izvedbah, smo knjižnici Zend Framework za čas realizacije dodelili le **2,5 točki**.

4.2.2 Drupal

Uporabnik	Funkcionalnost	Čas	Opomba
Skrbnik	Urejanje uporabnikov	0h	Funkcionalnost je že na voljo v jedru
	Urejanje skladišča	3,5h	Dodelava za 10% maržo na ceno
	Sestavljanje računalnikov	10h	Dodelave: sestavni deli, ki niso na zalogi niso na voljo, odstranitev nepotrebnih polj
	Pregled naročil	0h	Funkcionalnost je že na voljo v modulu Ubercart
Skladiščnik	Urejanje skladišča	1h	Uporabljena prilagojena funkcionalnost skrbnika
Sestavljalec	Sestavljanje računalnikov	1h	Uporabljena funkcionalnost skrbnika
Stranka	Ponudba	4h	Dodelava za filtriranje posameznih sestavnih delov iz ponudbe
	Košarica	4,5h	Dodelava za preprečevanje spreminjanja količine posameznih računalnikov v košarici
	Naročilo	8h	Dodelave: računalnik na voljo za prodajo samo enkrat, validacija
	Sestavljanje poljubnih računalnikov	2,5h	Uporabljena logika skrbnika, vendar pa so izbrani sestavni deli dodani v košarico namesto kreiranja novega računalnika v podatkovni bazi

Tabela 4.2: Poraba časa po funkcionalnostih uporabnikov za realizacijo s sistemom za urejanje vsebine Drupal.

Skupno število porabljenih ur iz SVN komentarjev je 34,5, kar je občutno manj kot pri izvedbi v Zend Framework knjižnici. Razlog za to so obstoječe funkcionalnosti v jedru ter množica obstoječih brezplačnih modulov. Dodatno je dodelave funkcionalnosti in razširitve obstoječih preprosto izvesti z uporabo dodatnih modulov. **Število točk: 4,5**.

4.2.3 Magento

Uporabnik	Funkcionalnost	Čas	Opomba
Skrbnik	Urejanje uporabnikov	0h	Funkcionalnost je že na voljo v jedru
	Urejanje skladišča	8h	Dodelava za 10% maržo na ceno
	Sestavljanje računalnikov	0h	Funkcionanlosti si že na voljo v jedru
	Pregled naročil	0h	Funkcionalnost je že na voljo v jedru
Skladiščnik	Urejanje skladišča	4h	Potrebna dodelava dovoljenj uporabnikov
Sestavljalec	Sestavljanje računalnikov	4h	Potrebna dodelava dovoljenj uporabnikov
Stranka	Ponudba	3,5h	Odstranjevanje prednastavljenih reklamnih oglasov iz ponudbe
	Košarica	5h	Dodelava za preprečevanje spreminjanja količine posameznih računalnikov v košarici
	Naročilo	6h	Odstranjevanje nepotrebnih korakov pri naročilu (naslov dostave, način plačila)
	Sestavljanje poljubnih računalnikov	0h	Funkcionalnost je že na voljo v jedru

Tabela 4.3: Poraba časa po funkcionalnostih uporabnikov za realizacijo s sistemom za elektronsko trgovanje Magento.

Skupno število porabljenih ur v SVN komentarjih je 30,5h, kar je približno enako kot pri izvedbi za Drupal. Magento se je pri porabi časa dobro odrezal,

ker ima funkcionalnosti za spletne trgovine izvedene bolj celovito, zato razen manjših sprememb ter dodelav preko dodatnih modulov na srečo konkretnejši posegi v delovanje jedra niso bili potrebni. **Število točk: 4,5**.

4.3 Nadgradnje

Nadgradnja pomeni nadomestitev produkta z novejšo različico istega produkta. Najbolj pogosto se uporablja v računalništvu in največkrat pomeni nadomestitev strojne, programske ali strojne programske opreme z novejšo oz. boljšo različico [29]. Pri nadgradnjah smo primerjali obstoječa navodila, težavnost posodabljanja posameznih izvedb ob izidu novejših verzij ter dokumentacijo verzij.

4.3.1 Zend Framework

Zend Framework nadgradnje izhajajo v obliki manjših nadgradenj ter večjih nadgradenj. Vse verzije od 1.x naprej so obratno združljive z vsako verzijo od 1.0 naprej [73], poleg tega pa je za vsako izmed verzij na voljo dokumentacija [2]. Posodobitev verzije je zaenkrat popolnoma preprosta in smo jo izvedli kar med razvojem spletne trgovine (nadgradnja iz verzije 1.11.4 na 1.11.9); vse kar je potrebno storiti je to, da razvijalec na strežniku nadomesti obstoječ direktorij Zend z najnovejšo verzijo [30]. Ob večjih nadgradnjah je potrebno biti pazljiv, ter upoštevati opozorila za migracijo [31]. Do večjih sprememb bo prišlo ob izidu verzije 2.0, za katero je obratna združljivost s starejšimi verzijami vprašljiva. **Število točk: 4,5**.

4.3.2 Drupal

Postopek za nadgradnjo je zelo podrobno opisan na uradni spletni strani [36]. Obstajata 2 vrsti nadgradnje. Pri manjši posodobitvi jedra, npr. iz 6.1 na 6.x, ni potrebno uporabiti vseh obstoječih verzij med navedenima verzijama, ampak jo lahko preprosto nadgradimo na 6.x. Druga vrsta je nadgradnja jedra, npr. iz 6.x na 7.x, za katero je potrebno najprej posodobiti Drupal 6 na najnovejšo verzijo, šele nato pa je možna nadgradnja na Drupal 7.

Vendar pa je pri nadgradnjah jedra potrebno upoštevati še kompatibilnost z moduli, ki jih uporabljamo. Kot primer bom navedel našo izvedbo, ki temelji na Drupal različici 6 ter Ubercart modulu v različici 2.0. Uporabili smo modul Upgrade Status [9] za preverjanje obstoječih posodobitev modulov ter združljivosti le-teh z nadgrajenim jedrom. Za Drupal 7 obstaja novejši modul Ubercart, ki pa je v Beta verziji in zato ni povsem stabilen, poleg tega pa

mu manjkajo določeni dodatni moduli, ki so na voljo v verziji 6 ter smo jih uporabili pri izvedbi. Če pa bodo pomanjkljivosti v prihodnosti odpravljene, bi lahko po nadgradnji jedra na verzijo 7 modul ročno nadgradili po postopku, opisanem na uradni strani [33].

Poleg nadgradnje verzije jedra in modulov je na voljo tudi prenos podatkov iz starejše verzije v novejšo, če gradimo spletno stran znova na podlagi najnovejše verzije jedra [32]. Za test smo brez težav posodobili modul `Views`, saj smo preko uporabe orodja Upgrade Status ugotovili, da obstaja novejša verzija. S posodobitvami in nadgradnjami je torej več dela kot pri izvedbi v knjižnici Zend Framework, kar je logično zaradi združljivosti, saj ima na voljo mnogo funkcionalnosti in modulov, medtem ko je Zend Framework samo jedro, na katerem razvijalec gradi sam. **Število točk: 3,5**.

4.3.3 Magento

Za nadgradnje do verzije 1.4.2.0 so na voljo navodila na Magento Wiki [34]. Če nadgradnjo delamo ročno, je potrebno narediti varnostno kopijo podatkovne baze, ki jo uvozimo v podatkovno bazo za novo verzijo, shraniti razvite module, slike produktov in določene konfiguracijske datoteke. Sledi prenos novejše verzije (torej do 1.4.2.0) ter nato standarden postopek konfiguracije spletne trgovine. Podatkovna baza se posodobi avtomatsko. Na voljo je tudi nadgradnja preko SSH lupine ter avtomatska nadgradnja preko orodja MagentoConnect Manager. Prav tako so na uradni strani na voljo navodila za nadgradnje novejših verzij. Poizkušali smo izvesti nadgradnjo iz verzije 1.5.1.10 na 1.6.0.0 beta [35], vendar ni šlo brez težav. Tudi ko smo uporabili avtomatski sistem za preverjanje obstoječih nadgradenj, nismo dobili rezultatov. Zato smo ročno vnesli razširitveni ključ v MagentoConnect Manager ter na ta način dobil seznam možnih nadgradenj, vendar pa smo ob nadgradnji v konzoli dobivali množico napak, nato pa se zaradi pomanjkanja časa z nadgradnjo nismo več ukvarjali. Naš vtis je, da so nadgradnje za Magento dokaj težavne in časovno potratne. **Število točk: 3**.

4.4 Prilagodljivost

Prilagodljivost se nanaša na zmožnost prilagajanja sistema v primeru pojava zunanje spremembe [14]. Tako smo pri primerjavi upoštevali stopnjo zmožnosti prilagajanja željam strank za posamezne izvedbe glede na izkušnje, pridobljene med izvedbo.

4.4.1 Zend Framework

Knjižnica je zelo prilagodljiva, saj ima razvijalec popolno svobodo in sam izbere v kolikšni meri uporabi komponente platforme ter kako realizira določeno funkcionalnost, zato z upoštevanjem specifikacij za spletno trgovino nismo imeli težav. Tudi vključevanje raznih dodatnih knjižnic ni problematično. Za spletno trgovino smo brez večjih težav implementirali administratorske funkcionalnosti z integracijo ExtJs Javascript knjižnice (JSON komunikacija s strežnikom). Omeniti velja tudi to, da je cena zelo velike prilagodljivosti večja poraba časa ter potencialna nevarnost nepregledne kode, če razvijalec ni pazljiv pri izvedbi [70]. **Število točk: 5**.

4.4.2 Drupal

Drupal je zelo prilagodljiv sistem, če razvijalec razume princip delovanja modulov. Na ta način lahko piše poljubne razširitve in funkcionalnosti. Tudi za našo izvedbo smo marsikatero podrobnost lahko dodelali v poljubnem modulu preko nadgradnje obstoječih funkcionalnosti. Zaradi visoke stopnje modularnosti ima projekt urejeno strukturo, kljub temu pa razvijalec nima tolikšne svobode kot pri izvedbi na podlagi knjižnice in to predvsem zaradi odvisnosti funkcionalnosti od jedra (višji nivo abstrakcije). To pomeni, da se je težje prilagajati naročnikom - upoštevanje njihovih želja lahko privede do presežka željenih funkcionalnosti zaradi prekomerne uporabe modulov. **Število točk: 4**.

4.4.3 Magento

Razvijalec lahko podobno kot pri sistemu Drupal piše samostojne module in s tem dodatne funkcionalnosti ali dodelave obstoječih. Vendar pa je pisanje le-teh težje, ker je časovno potratno že samo iskanje kode za določeno funkcionalnost, nato pa vsaj še enkrat toliko časa porabimo za samo spremembo. Magento je zato najmanj prilagodljiv, saj je najbolj kompleksen od vseh treh izvedb in za poljubno implementacijo zahteva največ poglabljanja v jedro. **Število točk: 3**.

4.5 Vzdrževanje

Pri vzdrževanju gre za pomoč pri reševanju problemov z določenim produktom. Zato smo pri tem kriteriju ocenjevali, koliko dela je potrebno vložiti v izvedbo naročniških zahtev po koncu realizacije spletnih trgovin; torej takrat, ko pri že obstoječi rešitvi pride do sprememb oziroma dodelav določenih funkcionalnosti. Za primerjavo smo izbrali 3 tipične primere uporabniških zahtev:

- sprememba uporabniškega vmesnika: stran naj bo širša za 100px,
- sprememba v logiki: skladiščnik lahko tudi dodaja sestavne dele,
- sprememba v podatkovni bazi: stranke imajo še dodatno polje za davčno številko.

Za vsako spremembo smo beležili porabljen čas ter jo prikazali s primerjavo izvorne kode pred in po izvedbi uporabniških zahtev (vrstice, pri katerih je bila potrebna sprememba, so obarvane modro).

4.5.1 Zend

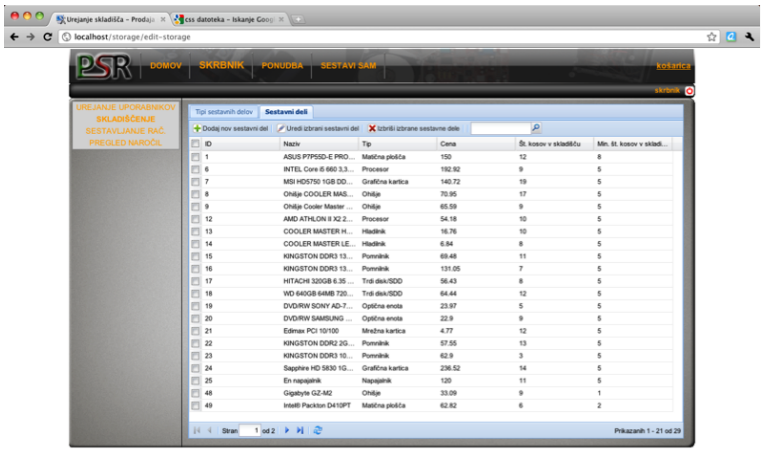
Sprememba uporabniškega vmesnika

Pri spremembi širine smo morali razširiti slike ozadja ter spremeniti stile v CSS datoteki `psr.css`, kar je razvidno na izseku izvorne kode 4.1. Rezultat vidimo na sliki 4.4. Porabljen čas: 15 minut.

```
1 .mainContent {
2    width:  960px;
3    margin: auto;
4    white-space: nowrap;
5    height: 674px;
6 }
```

```
.mainContent {
   width:  1060px;
   margin: auto;
   white-space: nowrap;
   height: 674px;
}
```

Izvorna koda 4.1: Izsek CSS predloge `psr.css` pred in po spremembi širine pri izvedbi s knjižnico Zend Framework.



Slika 4.4: Seznam sestavnih delov po spremembi širine pri izvedbi s knjižnico Zend Framework.

Sprememba v logiki

Na skladiščnikovi maski je bilo potrebno prikazati tudi gumb za dodajanje sestavnega dela, omogočiti prikaz forme za dodajanje ter v kodi za dodajanje novih sestavnih delov pri validaciji upoštevati tudi skladiščnika. Spremembe so prikazana na izsekih izvorne kode 4.2, 4.3 in 4.4. Porabljen čas: 15 minut.

```
if($isCurrentUserAdmin) {
   $toolbar[] = array(
      array(
         'text' => $label,
         'tooltip' => $tooltip,
         'iconCls' => 'add',
         'handler' => $handler
      ),
      '-'
   );
}
```

```
//if($isCurrentUserAdmin) {
   $toolbar[] = array(
      array(
         'text' => $label,
         'tooltip' => $tooltip,
         'iconCls' => 'add',
         'handler' => $handler
      ),
      '-'
   );
//}
```

Izvorna koda 4.2: Izsek izvorne kode razreda `Custom_View_Helper_Admin` pred in po spremembi logike pri izvedbi s knjižnico Zend Framework.

```
if(isset($request->getParam('ID0'))) {
   $isEdit = true;
   $id = $request->getParam('ID0');
   $adminAH->fillStPartFormData($id);
   $form->populate($post);
} else if(!$isCurrentUserAdmin) {
   throw new ErrorException(EXC_PERM);
}
```

```
if(isset($request->getParam('ID0'))) {
   $isEdit = true;
   $id = $request->getParam('ID0');
   $adminAH->fillStPartFormData($id);
   $form->populate($post);
} /*else if(!$isCurrentUserAdmin) {
   throw new ErrorException(EXC_PERM);
}*/
```

Izvorna koda 4.3: Izsek izvorne kode razreda `StorageController` pred in po spremembi logike pri izvedbi s knjižnico Zend Framework.

```
if(!$isCurrentUserAdmin) {
   throw new ErrorException(EXC_PERM);
}

$this->insert($formData);
```

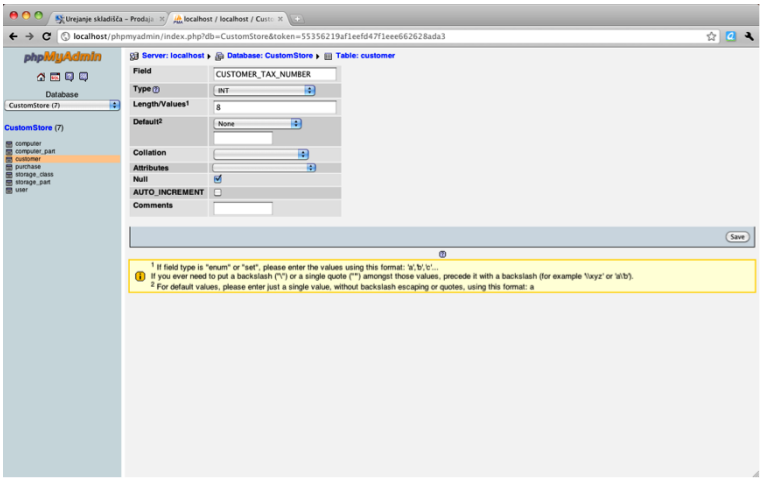
```
/*if(!$isCurrentUserAdmin) {
   throw new ErrorException(EXC_PERM);
}*/

$this->insert($formData);
```

Izvorna koda 4.4: Izsek izvorne kode razreda `Custom_Model_DbTable_StoragePart` pred in po spremembi logike pri izvedbi s knjižnico Zend Framework.

Sprememba v podatkovni bazi

Preko orodja phpMyAdmin smo v tabeli, ki vsebuje stranke, dodali polje CUSTOMER_TAX_NUMBER (slika 4.5). Poleg tega smo ga prikazali v formi za urejanje uporabnika (dodelava je prikazana v izseku izvorne kode 4.5). Porabljen čas: 10 minut.



Slika 4.5: Dodajanje polja Davčna številka za stranke preko orodja phpMyAdmin za implementacijo s knjižnico Zend Framework.

```
1  //tax number
2  $taxNumber = new Zend_Form_Element_Text(CUSTOMER_TAX_NUMBER);
3  $taxNumber->setLabel(TAX_NUMBER)
4     ->setAttrib('class', 'column formInput')
5     ->setRequired(true)
6     ->setDecorators($this->elementDecorators);
7
8  $notEmpty = new Zend_Validate_NotEmpty();
9  $notEmpty->setMessage(MISSING_TAX_NUMBER);
10
11 //tax code has length of 8 (numerical)
12 $regex = new Zend_Validate_Regex('/^(\d{8}|\s)?$/');
13 $regex->setMessage(WRONG_TAX_NUMBER_FORMAT);
14 $taxNumber->addValidators(array(array($notEmpty, true), $regex));
```

Izvorna koda 4.5: Izsek izvorne kode razreda `Custom_Form_CustomerData` za prikaz polja davčna številka pri izvedbi s knjižnico Zend Framework.

Za vse tri popravke skupaj smo porabili 35 minut. Prednost pri vzdrževanju aplikacije, ki je napisana s pomočjo knjižnice Zend Framework, je ta, da razvijalec praktično vse funkcionalnosti napiše sam in zato hitreje ve, kje je potrebna dodelava. Vendar pa je časovno lahko bolj potratno kot pri izvedbi v sistemu Drupal, saj je potrebno vse implementirati ročno. **Število točk: 4**.

4.5.2 Drupal

Sprememba uporabniškega vmesnika

Pri spremembi širine smo najprej z orodjem Firebug analizirali CSS predlogo v uporabi za prednastavljeno temo (`style.css`), naredili potreben popravek, ki je prikazan v izseku izvorne kode 4.6, ter počistili predpomnilnik spletnega brskalnika. Porabljen čas: 5 minut.

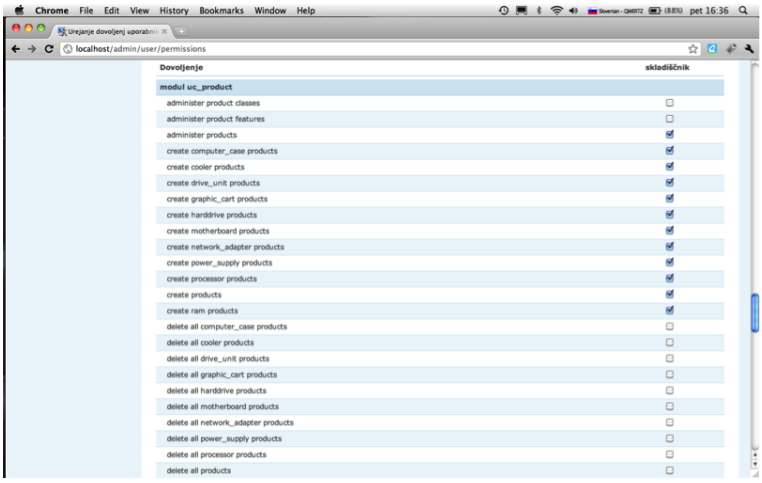
```
1  #wrapper #container {
2    margin: 0 auto;
3    padding: 0 20px;
4    max-width:  1270px;
5  }
```

```
#wrapper #container {
  margin: 0 auto;
  padding: 0 20px;
  max-width:  1370px;
}
```

Izvorna koda 4.6: Izsek CSS predloge style.css pred in po spremembi širine pri izvedbi s sistemom Drupal.

Sprememba v logiki

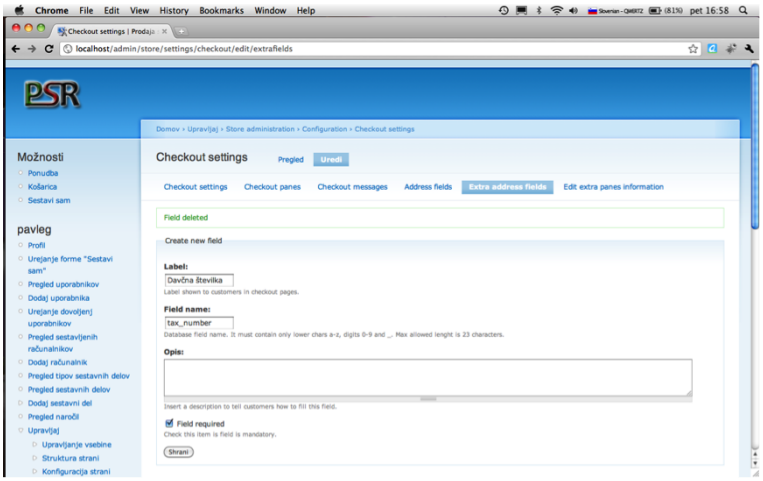
Pri spremembi logike smo morali med dovoljenji uporabnikov izbrati možnost dodajanja novega računalnika za skladiščnika, kar lahko vidimo na sliki 4.6. Porabljen čas: 5 minut.



Slika 4.6: Omogočanje dodajanja sestavnih delov skladiščnika za Drupal.

Sprememba v podatkovni bazi

Za dodajanje polja ni bil potreben direkten poseg v podatkovno bazo, ampak samo dodajanje modula `uc_extra_fields_pane`, ki omogoča definiranje dodatnih polj pri naročilu (slika 4.7). Porabljen čas: 15 minut.



Slika 4.7: Dodajanje polja Davčna številka za stranke pri izvedbi s sistemom Drupal.

Za vse tri popravke skupaj smo porabili 25 minut. Pri vzdrževanju je Drupal zagotovo zmagovalec, saj je preko množice brezplačnih modulov in manjših dodelav v kodi omogočil hitre popravke in nadgradnje. **Število točk: 5**.

4.5.3 Magento

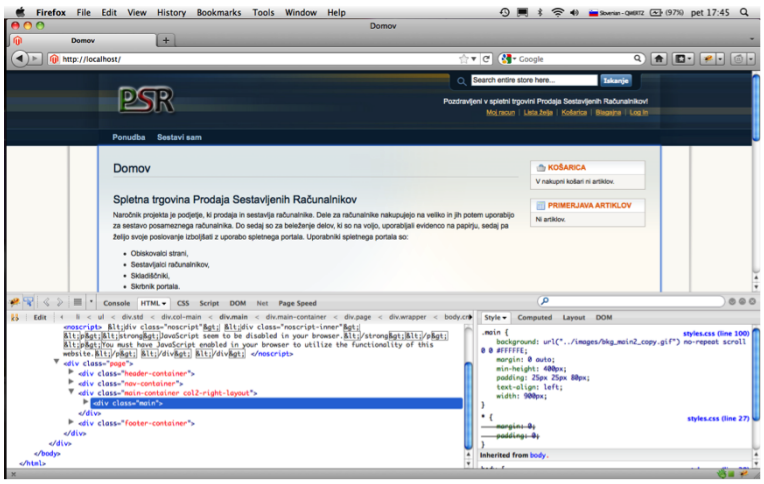
Sprememba uporabniškega vmesnika

Uporabili smo podoben pristop kot pri sistemu Drupal; z orodjem Firebug smo analizirali css predlogo `styles.css` (slika 4.8) pri prednastavljeni temi in naredili potreben popravek (izsek izvorne kode 4.7) ter razširili `.gif` slike ozadja za 100px. Porabljen čas: 20 minut.

```
.main {
   width:900px;
   margin:0 auto;
   min-height:400px;
   padding:25px 25px 80px;
   background:#fffffe ↙
        ↳ url(../images/bkg_main2.gif) ↙
        ↳ 0 0 no-repeat;
   text-align:left;
}
```

```
.main {
   width:1000px;
   margin:0 auto;
   min-height:400px;
   padding:25px 25px 80px;
   background:#fffffe ↙
        ↳ url(../images/bkg_main3.gif) ↙
        ↳ 0 0 no-repeat;
   text-align:left;
}
```

Izvorna koda 4.7: Izsek CSS predloge `styles.css` pred in po spremembi širine pri izvedbi s sistemom Magento.



Slika 4.8: Analiziranje teme z orodjem Firebug pri izvedbi s sistemom Magento.

Sprememba v logiki

Pri spremembi logike skladiščnika smo morali samo prikazati gumb za dodajanje artiklov v primeru skladiščnika v dodatnem modulu. Sprememba je

prikazana v izseku izvorne kode 4.8. Porabljen čas: 5 minut.

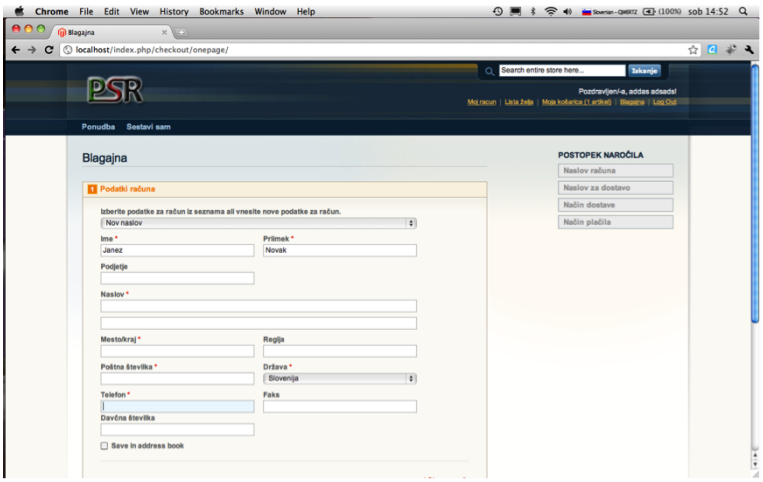
```
if(!$this->isSkladiscnik) {
   $this->_addButton('add_new', array(
      'label' => $addLabel,
      'onclick' => $addAction,
      'class' => 'add'
   ));
}
```

```
//if(!$this->isSkladiscnik) {
   $this->_addButton('add_new', array(
      'label' => $addLabel,
      'onclick' => $addAction,
      'class' => 'add'
   ));
//}
```

Izvorna koda 4.8: Izsek izvorne kode razreda `PSR_Roles_Adminhtml_Block_Catalog_Product` pred in po spremembi logike pri izvedbi s sistemom Magento.

Sprememba v podatkovni bazi

Za dodajanje polj v postopek naročila obstaja več razširitev, vendar pa so vse plačljive. Zaradi tega smo se lotili problema z dodatnim modulom, vendar nam zaradi kompleksnosti integracije z logiko za shranjevanje polj naročila v podatkovno bazo in pomankanja časa dodelave ni uspelo implementirati v celoti (slika 4.9). Kljub temu je očitno, da so lahko dodelave brez nakupa razširitev v Magentu časovno precej potratne. Porabljen čas: 4h.



Slika 4.9: Polje Davčna številka naročnika pri izvedbi s sistemom Magento.

Za vse tri popravke skupaj smo porabili 4h 25min in kljub temu nismo uspeli dodelati vsega. To je velik minus, saj mora razvijalec zelo dobro razumeti kompleksno jedro ali pa za vsako zahtevnejšo zahtevo naročnika dokupiti drage razširitve, če le-te obstajajo. **Število točk: 2**.

4.6 Spletna podpora

Pri kriteriju spletne podpore smo primerjali spletno skupnost, dokumentacijo, forume, vodiče ipd., s katerimi smo si pomagali pri razvoju spletnih trgovin. Pogoj je bil, da so brezplačni.

4.6.1 Zend

Uradna dokumentacija bi bila lahko malo boljša, saj pokrije samo osnove z zelo preprostimi primeri [37]. Za kompleksnejše funkcionalnosti (npr. dodajanje relativne poti do lokacije razredov v Bootstrap-u, oblikovanje obrazcev s kompomento `Zend_Form` ...) pa je potrebno najti ustrezen vodič in se učiti iz primerov oz. se poglobiti v podrobnosti kode komponent knjižnice [60]. Poleg tega si lahko pomagamo z raznimi forumi, na katerih sodeluje množica razvijalcev, ki so pripravljeni pomagati. Tudi na ta način smo med razvojem večkrat dobili odgovor na razna vprašanja [53, 68]. Obstaja tudi spletna skupnost, kjer lahko vsak razvijalec prijavi potencialne hrošče platforme in predlaga izboljšave/dodelave [51]. **Število točk: 4**.

4.6.2 Drupal

Spletna podpora za Drupal je odlična, saj smo z lahkoto našli vse, kar smo potrebovali za implementacijo. Na voljo je podrobna API dokumetacija [7], množica tutorialov, zbirk dobro opisanih modulov z demo primeri, IRC kanalov, skupin [52] ter forumov[8], kjer razvijalci z veseljem odgovorijo na različna vprašanja. Poskrbljeno je tudi za prijavo hroščev, saj obstaja poseben modul Project Issue, namenjen prav temu. Če pa želi razvijalec strokovno pomoč, jo lahko dobi preko različnih skupin kot so Acquia Drupal, Hot Drupal in podobne [17]. **Število točk: 5**.

4.6.3 Magento

Spletna podpora za jedro je solidna, uporaben pa je predvsem Magento Wiki [27]. Glede izvedbe kompleksnejših funkcionalnost smo na raznih forumih pogosto bodisi našli odgovor tipa "naloži to in to dodatno razširitev", ki je seveda plačljiva ali pa sploh ni bilo nobenega odgovora na zastavljeno vprašanje. Vsak razvijalec lahko za Magento tako kot pri knjižnici Zend Framework in sistemu Drupal prijavi hrošče [43].

Glede na našo izkušnjo z razvojem menimo, da je za Magento ustrezna podpora na voljo samo v primeru uporabe Enterprise oz. Proffesional različice in plačljivih modulov. Če pa gre za izvedbo na podlagi brezplačne različice, je

časovno potratna poglobitev v kompleksno jedro praktično neizogibna (analiza logike, razhroščevanje). **Število točk: 3**.

4.7 Hitrost

Za oceno hitrosti smo primerjali povprečen čas trajanja poizvedbe na strežniku za vsako izmed izvedb. Uporabili smo orodje Apache JMeter [1], ki omogoča tako snemanje uporabniških akcij brskalnika preko funkcionalnosti HTTP Proxy kot tudi meritve. Na ta način smo kreirali testne scenarije ter nato izvajali meritve.

4.7.1 Testni scenariji

Za vsakega izmed testnih scenarijev smo uporabniške akcije razdelili glede na tip uporabnika:

- admin: prijava, dodajanje uporabnika, urejanje uporabnika, brisanje uporabnika, dodajanje tipa sestavnih delov, urejanje tipa sestavnih delov, brisanje tipa sestavnih delov, dodajanje sestavnega dela, urejanje sestavnega dela, brisanje sestavnega dela, dodajanje računalnika, urejanje računalnika, brisanje računalnika, pregled naročil,
- skladiščnik: urejanje sestavnega dela,
- sestavljalec: dodajanje računalnika, urejanje računalnika, brisanje računalnika,
- stranka: ogled sestavljenih računalnikov, dodajanje poljubnega računalnika v košarico, praznjenje košarice, sestavljanje poljubnega računalnika, naročilo, dodajanje poljubnega računalnika v košarico, naročilo.

Pri vsakem scenariju več uporabnikov istočasno izvaja akcije. Za izračun števila sočasnih uporabnikov smo uporabili naslednjo formulo:

$$C = \frac{n * L}{T} \tag{4.1}$$

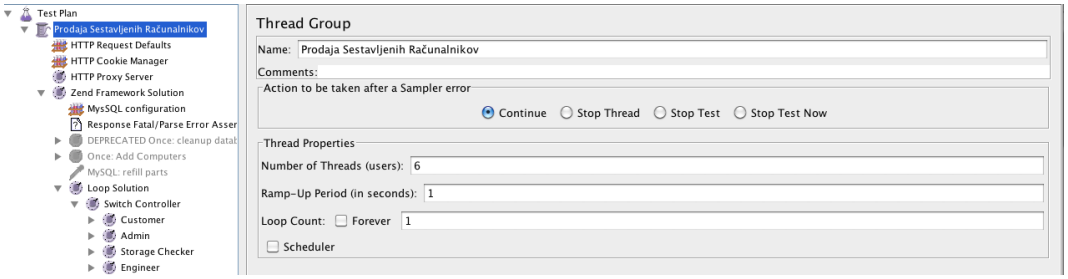
C predstavlja število sočasnih uporabnikov, n število vseh uporabnikov v določenem časovnem obdobju, L povprečen čas uporabe strani posameznega uporabnika in T časovno obdobje [5]. Za izvajanje meritev smo predvideli, da naročnik spletne trgovine na povprečen dan ocenjuje 30 administratorskih (skrbnik, skladiščnik, sestavljalec) dostopov po 90 minut ter 300 dostopov strank po 10 minut. Pri tem se upošteva čas od 8:00 do 24:00, torej 16 ur, kar znaša 960 minut. Iz formule 4.1 torej lahko izračunamo C_a (št. sočasnih

administratorskih uporabnikov) po formuli 4.2 in C_s (število sočasnih strank) po formuli 4.3:

$$C_a = \frac{30 * 90}{960} = 2,81 \tag{4.2}$$

$$C_s = \frac{300 * 10}{960} = 3,13 \tag{4.3}$$

Po izračunu smo dobili 3 administratorje in 3 stranke, ki istočasno uporabljajo spletno trgovino (slika 4.10). Administratorje smo razdelili na enega skrbnika, enega skladiščnika in enega sestavljalca. Med dvema poizvedbama posameznega uporabnika smo dodali 1000 milisekund premora s standardnim odklonom 500 milisekund, da je test bolj realen in ne pride do preobremenitve strežnika. Poleg tega smo za pridobitev boljših rezultatov uporabili zanko desetih ponovitev poizvedb vsakega izmed uporabnikov (slika 4.11).



Slika 4.10: Nastavitev števila sočasnih uporabnikov v testnem scenariju orodja Apache JMeter.

Slika 4.11: Nastavitev zanke desetih ponovitev v testnem scenariju orodja Apache JMeter.

Glede na izdelane testne scenarije je skupno število poizvedb različno pri posameznih izvedbah, kar smo tudi analizirali pri primerjavi optimizacije.

	Zend Framework	Drupal	Magento
Skrbnik	49	52	72
Skladiščnik	12	14	8
Sestavljalec	16	15	39
Stranka	20	31	35
Skupaj po št. sočasnih uporabnikov	**49*1 + 12*1 + 16*1 + 20*3 = 137**	**51*1 + 14*1 + 15*1 + 31*3 = 171**	**72*1 + 8*1 + 39*1 + 35*3 = 224**
Skupaj po 10 ponovitvah	**1370**	**1710**	**2240**

Tabela 4.11: Število poizvedb testnih scenarijev po posameznih izvedbah.

4.7.2 Meritve

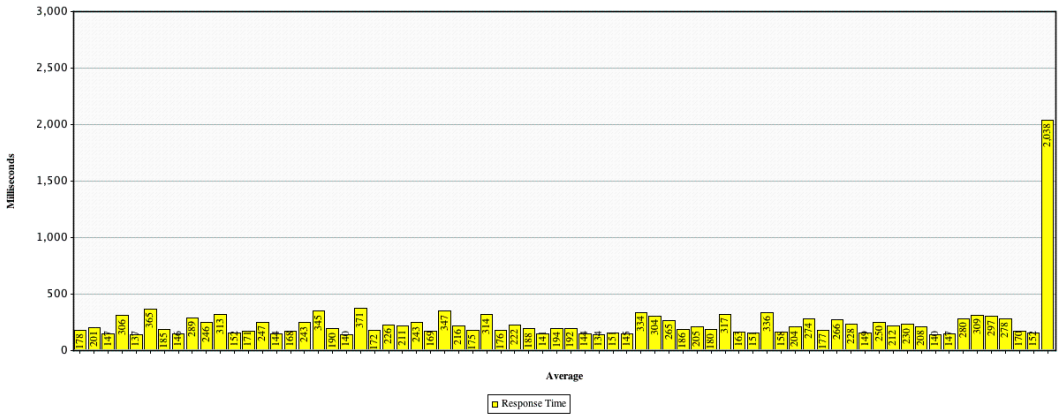
V rezultatih meritev smo za posamezne izvedbe upoštevali povprečen čas poizvedbe, minimalen čas poizvedbe, maksimalen čas poizvedbe, standardni odklon, mediano ter število poizvedb na minuto ter jih primerjali med seboj. Poleg tega smo razpršitev povprečnih časov poizvedb prikazali tudi vizualno z grafi.

Tabela rezultatov

	Zend Framework	Drupal	Magento
Št. poizvedb	1370	1710	2240
Povprečje(ms)	237	1176	6992
Minimum(ms)	130	359	2
Maksimum(ms)	2251	13263	115376
Standardni odklon(ms)	178,18	1091,01	10513
Mediana(ms)	208	859	3202
Št. poizvedb/min	58	46,2	26,9

Tabela 4.12: Tabela rezultatov testiranja hitrosti po posameznih izvedbah.

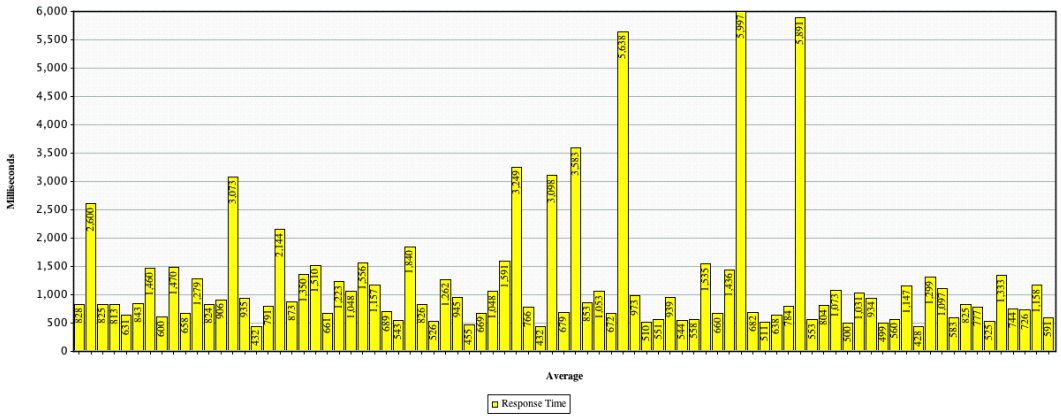
Zend



Slika 4.12: Graf povprečnih časov poizvedb za Zend Framework.

Iz tabele je razvidno, da se je izvedba na podlagi knjižnice Zend Framework najbolje odrezala pri meritvi hitrosti. Če gledamo razliko med mediano in povprečjem, je tudi razpršenost poizvedb majhna. Iz grafa na sliki 4.12 je razvidno, da iz povprečja izstopa samo ena poizvedba in sicer je to skrbniški pregled naročil, saj prebere večje število podrobnosti računalnikov iz podatkovne baze. Drugih posebnosti ni. **Število točk: 5**.

Drupal

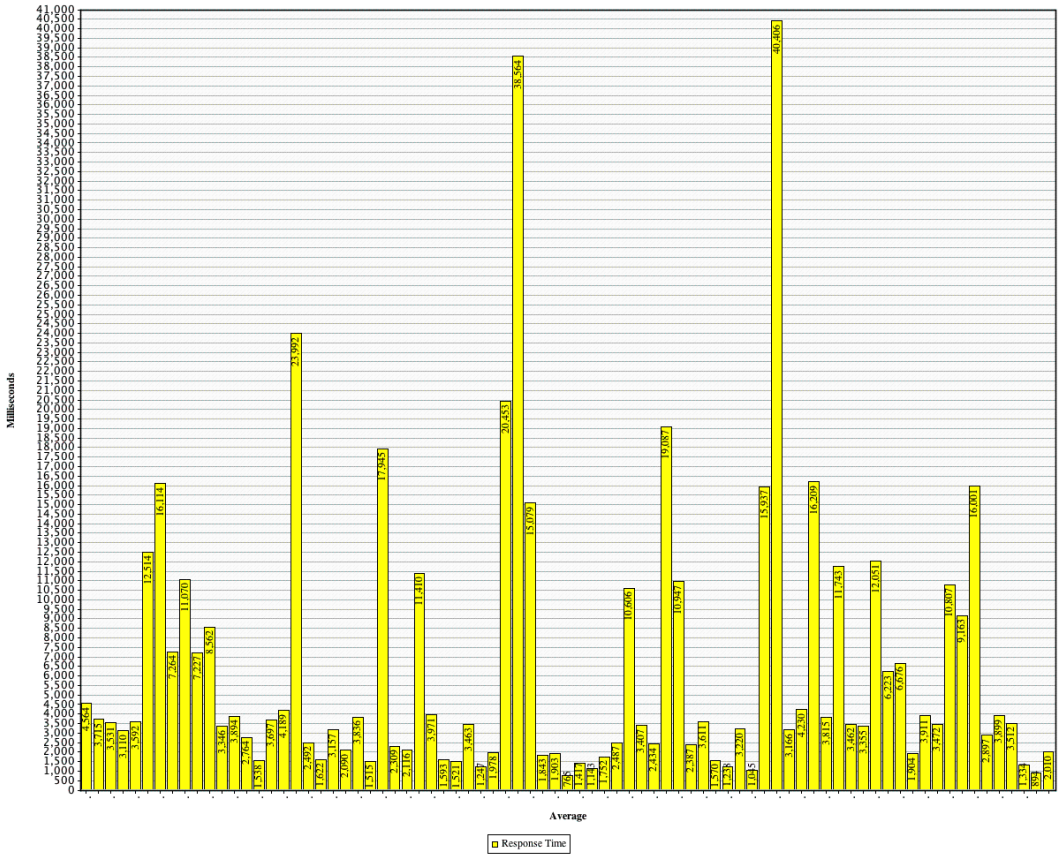


Slika 4.13: Graf povprečnih časov poizvedb za Drupal.

Iz tabele je razvidno, da je hitrost slabša kot pri izvedbi s knjižnico Zend Framework, vendar pa še vedno precej boljša kot izvedba s sistemom Magento;

ista zgodba je pri razpršenosti poizvedb. Največ časa traja izvajanje poizvedbe za urejanje tipov sestavnih delov, saj ima vsak množico prednastavljenih atributov (graf 4.13, najvišji trije stolpci na desni). Iz povprečja izstopajo še poizvedbe za zaključevanje naročila, pregled ponudbe, urejanje sestavnih delov in prijava. **Število točk: 3,5**.

Magento



Slika 4.14: Graf povprečnih časov poizvedb za Magento.

Iz tabele rezultatov meritev je razvidno, da se je najslabše izkazal Magento, ki je glede strojne opreme zelo požrešen, kar se pozna tudi na hitrosti. Iz grafa na sliki 4.14 je razvidno, da je povprečni čas poizvedbe za Magento zelo slab, poleg tega pa je kar precej poizvedb kritičnih; najbolj izstopata (najvišja stolpca) potrjevanje naročila ter shranjevanje tipa sestavnih delov, ki ima množico prednastavljenih atributov. Zelo slabe čase imajo še poizvedbe povezane s košarico, urejanjem računalnikov s strani skrbnika oziroma sestavljalca in sestavljanjem poljubnega računalnika s strani stranke. **Število točk: 1,5**.

4.8 Obremenitev linije

Za obremenitev linije smo uporabili meritve iz primerjave hitrosti posameznih implementacij, in sicer smo povprečno velikost odziva na poizvedbo pomnožili s številom vseh poizvedb ter s tem dobili vsoto poslanih podatkov strežnika uporabnikom spletne trgovine ob izvajanju testiranja.

4.8.1 Tabela rezultatov

	Zend Framework	Drupal	Magento
Povprečna velikost odziva na poizvedbo (KB)	71,86	28,00	71,38
Število poizvedb	1370	1710	2240
Obremenitev linije (MB)	**98,54**	**47,88**	**159,89**

Tabela 4.13: Tabela rezultatov izračunov obremenitve linije.

4.8.2 Zend

Število poizvedb je po pričakovanjih najmanjše, ker je implementacija popolnoma prilagojena potrebam spletne trgovine, vendar pa je velikost povprečne poizvedbe večja kot pri sistemu Drupal. To skupaj znaša 98,54 MB poslanih podatkov. Razlog za tako povprečno velikost poizvedbe je predvsem uporaba ExtJS ter pomanjkanje kompresije podatkov (več o tem pa pri primerjavi optimizacije). **Število točk: 3,5**.

4.8.3 Drupal

Drupal je zmagovalec pri obremenitvi linije z 47,88 MB prenosa, kljub temu, da je število poizvedb večje kot pri knjižnici Zend Framework. Razlog tiči v povprečni velikosti poizvedbe, ki je za 61% manjša kot pri rezultatih za Zend Framework in Magento. Taka velikost je dosežena s kompresijo datotek in manj JavaScripta. **Število točk: 5**.

4.8.4 Magento

Najslabše se je odrezal Magento z 159,89 MB, kar je predvsem posledica velikega števila poizvedb, ki so potrebne za izvedbo vseh potrebnih akcij. Največji krivec je logika jedra v sistemu Magento, ki skrbi za sestavljanje računalnikov

in zahteva precej več poizvedb za konfiguracijo kot pa Drupal in Zend Framework. Poleg tega tudi velikost povprečne poizvedbe ni prav majhna, kar je podobno kot pri knjižnici Zend Framework krivda uporabe Javascripta. **Število točk: 2,5**.

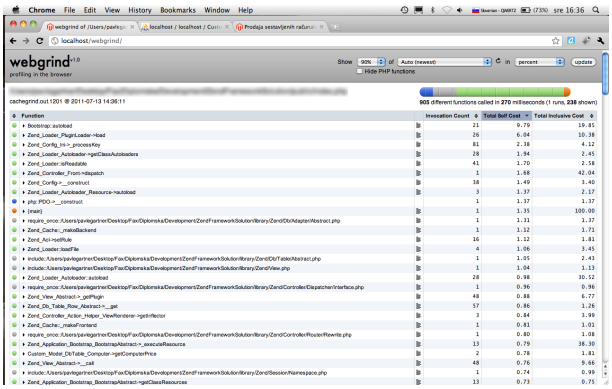
4.9 Optimizacija

Optimizacija je proces modificiranja sistema za doseganje večje učinkovitosti ter manjše porabe sredstev [38]. Spletne trgovine smo analizirali s pomočjo orodij webgrind, Firebug in PageSpeed, preko katerih smo ugotovili počasne dele kritičnih poizvedb (t.j. poizvedb strank) ter analizirali možnosti optimizacije.

4.9.1 Zend

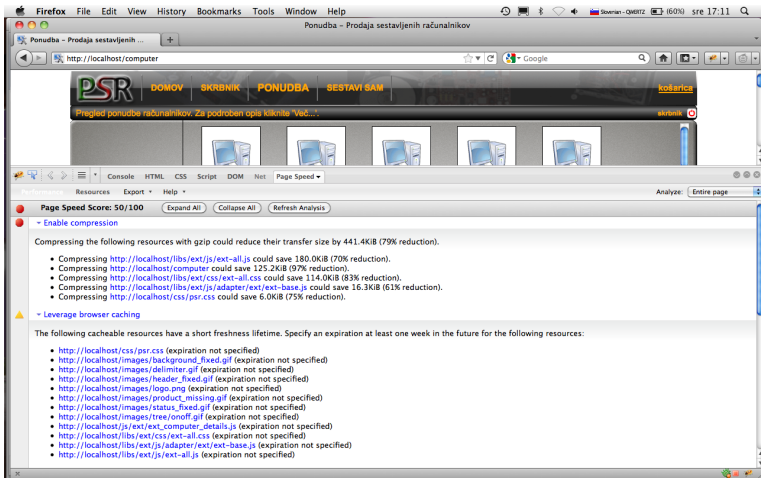
Največ je odvisno od izvedbe, saj razvijalec piše spletno stran sam in s tem lahko doseže boljšo optimizacijo kode, poleg tega pa je lažje doseči manjše število poizvedb za izvajanje želenih akcij v spletni trgovini, ki je od vseh treh izvedb najnižje ravno pri knjižnici Zend Framework (1370). Možne so tudi pospešitve same knjižnice kot npr. uporaba strežnika Zend Server Community Edition (vsebuje Zend Optimizer+).

Analiza z orodjem webgrind (slika 4.15) je pokazala, da največjo obremenitev procesorja strežnika v večini primerov (košarica, naročilo, sestavi sam) predstavlja `autoloader`, ki nalaga skripte, potrebne za delovanje strani, sicer pa je to stvar jedra, tako da je ni smiselno spreminjati, razen z obstoječimi pospešitvami same knjižnice. Naslednja stvar je branje podatkov iz baze. Strežnik je pri prikazu ponudbe najbolj obremenjen, saj se naložijo vsi računalniki s sestavnimi deli, kar bi se dalo optimizirati s prikazom po straneh, shranjevanjem podatkov v predpomnilnik (komponenta `Zend_Cache`), optimizacijo SQL poizvedb ter indeksiranjem podatkovne baze. Poleg tega je dokaj potratno generiranje raznih obrazcev, kar bi bilo prav tako možno pospešiti z uporabo predpomnilnika. Optimizacija je možna na mnogih mestih, saj so vse funkcionanosti napisane poljubno.



Slika 4.15: Rezultati, pridobljeni s pomočjo orodja webgrind pri izvedbi s knjižnico Zend Framework.

Pri analizi z orodjem PageSpeed (slika 4.16) smo ugotovili, da bi lahko stran dodatno optimizirali s kompresiranjem ExtJs knjižnice (JavaScript in CSS) ter osnovne CSS datoteke spletne trgovine (`psr.css`). Poleg tega bi stran pohitrilo definiranje časa osvežitve posameznih tipov datotek v `.htaccess` datoteki (Leverage Browser Caching).



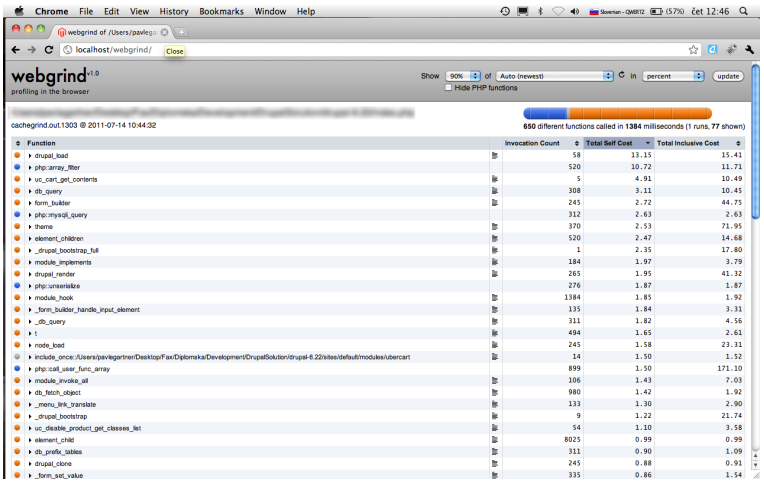
Slika 4.16: Rezultati, pridobljeni s pomočjo orodja PageSpeed pri izvedbi s knjižnico Zend Framework.

Že med razvojem je potrebno paziti, da je napisana koda čim bolj optimalna, poleg tega pa je potrebno v primeru uporabe potratnih JavaScript knjižnic poskrbeti za kompresijo le-teh [24]. To dopušča veliko prostora za optimizacijo, vendar pa posledično zahteva več časa za dober rezultat ter potencialno možnost slabe optimizacije. **Število točk: 3,5**.

4.9.2 Drupal

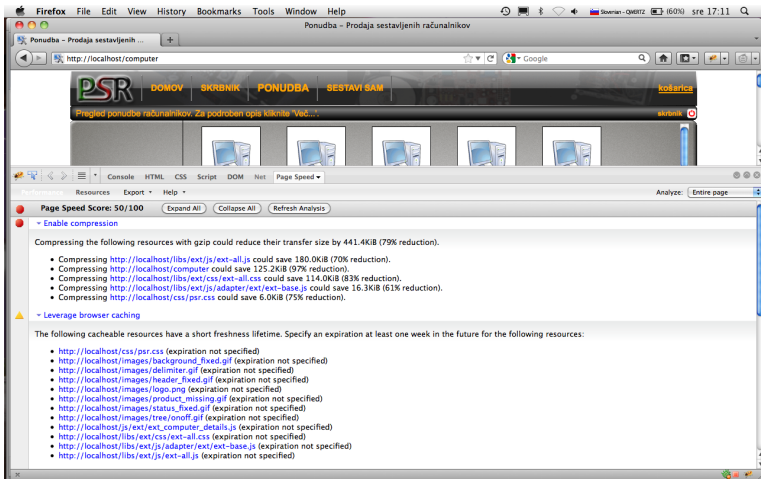
Za jedro je možno nastaviti način delovanja predpomnilnika (do katere mere se uporablja) ter uporabiti kompresijo datotek, ki je že vključena v jedro sistema Drupal. Ostalo je odvisno od razvijalca (koliko modulov uporablja, optimizacija pri pisanju svojih modulov). Same spremembe v jedru so nesmiselne, saj v primeru sprememb strani ni več možno posodobiti pri izidu novejših različic. Poleg tega je zaradi uporabe obstoječih funkcionalnosti jedra in modulov potrebnih več poizvedb za izvajanje akcij, ki so specifične za spletno trgovino Prodaja sestavljenih računalnikov. To je razvidno tudi v številu poizvedb pri meritvi hitrosti, ki je višje kot pri knjižnici Zend Framework (1710).

Webgrind je pokazal (slika 4.17), da največjo obremenitev strežnika predstavlja klic funkcije `drupal_load`, katere delovanje je podobno autoloaderju pri knjižnici Zend Framework. Ker je to del jedra, optimizacija ni smiselna. Izboljšati pa bi se dalo poljubne module, saj so nekateri klici dokaj potratni. Kot primer bomo navedli klic funkcije `uc_cart_get_contents` pri prikazu ponudbe, s katerim se za vsak računalnik preverja, ali je že v košarici. Lahko bi namesto tega klica ob vsakem dodajanju računanika v košarico le-tega označili kot obstoječega v košarici in take sfiltrirali pri prikazu. Za boljšo optimizacijo torej lahko razvijalec poskrbi pri pisanju poljubnih modulov.



Slika 4.17: Rezultati, pridobljeni s pomočjo orodja webgrind pri izvedbi s sistemom Drupal.

Analiza z orodjem PageSpeed (slika 4.18) je pokazala, da je kompresiranje pri sistemu Drupal izvedena zelo solidno, tako da niti ni potrebna s strani strežnika. Če pa jo imamo vključeno s strani strežnika, je priporočljivo, da jo v jedru izključimo.



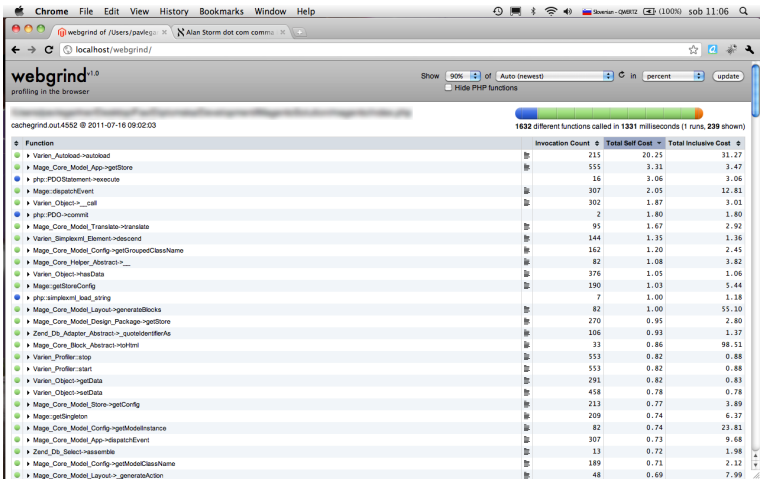
Slika 4.18: Rezultati, pridobljeni s pomočjo orodja PageSpeed pri izvedbi s sistemom Drupal.

Hitrost jedra glede na rezultate, dobljene z orodjem JMeter, in kompresija sta solidna in ju dobimo v paketu brez vloženega dela. Edini minus je ta, da ni toliko prostora za optimizacijo kot pri razvoju na podlagi knjižnice, vendar pa razvijalec na ta način prihrani čas. **Število točk: 4**.

4.9.3 Magento

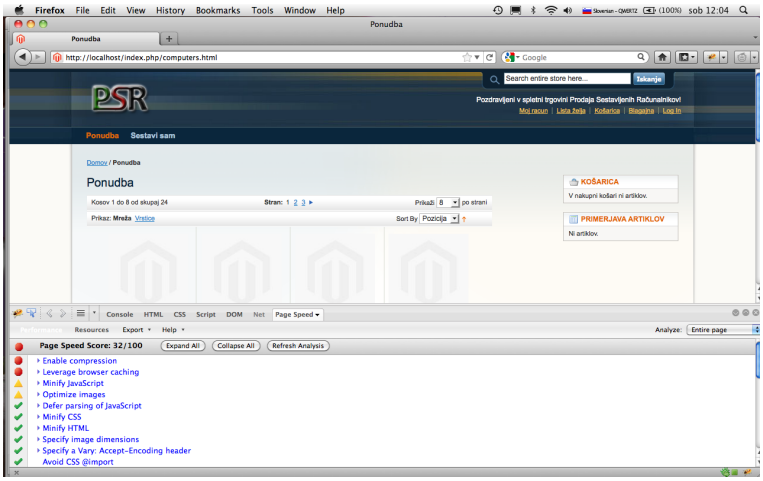
Podobno kot pri sistemu Drupal je v jedru možno nastaviti delovanje predpomnilnika, poleg tega pa še indeksiranje podatkovne baze. Sama optimizacija jedra je nesmiselna, saj v primeru sprememb strani ni več možno posodobiti pri izidu novejših verzij. Poleg tega je zaradi uporabe obstoječih funkcionalosti potrebnih več akcij za dosego rezultatov, ki so specifični za spletno trgovino Prodaja sestavljenih računalnikov. To je razvidno tudi v številu poizvedb pri meritvi hitrosti, ki je občutno najslabša v primerjavi z ostalima dvema izvedbama (2240).

Webgrind je pokazal podobne rezultate kot za Drupal in Magento (slika 4.19); največja obremenitev je funkcija `autoloader`, ki nalaga potrebne skripte in je del jedra ter kot taka predstavlja dobrih 10% celotne obremenitve. Koda v poljubnih razširitvah pri Magentu ni kritična, saj popravki niso vplivali na performanse in posledično ne spadajo med problematične pri analizi z webgrindom. Optimizacija skripte je torej podobno kot pri sistemu Drupal smiselna samo v poljubnih razširitvah.



Slika 4.19: Rezultati, pridobljeni s pomočjo orodja webgrind pri izvedbi s sistemom Magento.

Pri analizi s PageSpeedom (slika 4.20) smo ugotovili, da bi bilo potrebno vključiti kompresijo (podobno kot pri Zend Framework knjižnici), saj ima Magento najslabše rezultate izmed vseh treh izvedb. To je posledica dejstva, da Magento privzeto uporablja veliko JavaScripta za vsako poizvedbo na strežnik. Kot primer prilagam rezultat analize prikaza ponudbe.



Slika 4.20: Rezultati, pridobljeni s pomočjo orodja PageSpeed pri izvedbi s sistemom Magento.

Glede na to, da Magento predstavlja obstoječo rešitev za spletno trgovino, bi razvijalec pričakoval solidno hitrost ter vgrajen mehanizem za kompresijo datotek, čeprav je res, da se da to na dokaj preprost način vključiti na strežniku. Shranjevanje v predpomnilnik je že vključeno v jedru, sama skripta

jedra pa je zelo slabo optimizirana glede na rezultate meritev z orodjem JMeter. Za solidno hitrost delovanja bi torej potrebovali optimizacijo (kompresija) ter močnejši strežnik kot Zend Framework ali Drupal. **Število točk: 3**.

4.10 Plačljive opcije

Osebno nismo preizkusili plačljivih opcij, tako da smo primerjavo izvedli z raziskavo obstoječih možnosti za dopolnitve in izboljšave pri doseganju boljših rezultatov glede na pomankljivosti posameznih izvedb spletnih trgovin; mednje spadajo plačljive razširitve, gostovanja z boljšo podporo, strokovna pomoč, ipd. To pride v poštev predvsem takrat, ko ima naročnik spletne trgovine na voljo dovolj finančnih sredstev za ta namen.

4.10.1 Zend

Podjetje Zend ima tako za knjižnico kot tudi za ostale PHP aplikacije na voljo center za podporo [67], v katerem je na voljo mnogo plačljivih opcij, in sicer: Zend Server strežnik (1195-9166), razvojno okolje Zend Studio (299$), ki imata knjižnico Zend Framework že vključeno v paket in pa tudi podporo razvijalcem ter strežnikom gostovanja [71]. Poleg tega obstaja množica gostovanj, namenjenih knjižnici Zend Framework [69]. Torej lahko s plačljivimi rešitvami razvijalec naročniku zagotovi predvsem boljšo optimizacijo preko uporabe namenskega strežnika in gostovanja s podporo ter pomoč pri uporabi knjižnice in s tem potencialno tudi hitrejši razvoj aplikacij. **Število točk: 3**.

4.10.2 Drupal

Veliko podpore v obliki spletne skupnosti in razširitev z brezplačnimi moduli je za Drupal na voljo brezplačno. Vendar pa vseeno obstaja množica podjetij, ki ponujajo različne storitve kot so razvoj tem, dodatnih modulov, konfiguracija, podpora ... Poleg tega se lahko naročnik odloči tudi za razna gostovanja in strežnike, ki imajo posebno podporo za Drupal [11]. Kot zanimivost bomo navedli tudi opcijo gostovanja na strežniku Zend Server [72]. **Število točk: 3**.

4.10.3 Magento

Poleg brezplačne sta na voljo še dve plačljivi različici, in sicer Magento Professional ter Magento Enterprise [46]. Magento Professional stane 2995$ letno ter ima v primerjavi z brezplačno različico tudi podporo s strani partnerjev (Magento Solution Partners), garancijo, darilne bone za spletno trgovino in

enkripcijo podatkov. Magento Enterprise stane 12990$ letno in ima v primerjavi z brezplačno različico iste prednosti kot Magento Professional, poleg tega pa ima še izboljšan CMS (CMS+), urejanje dinamične podatkovne baze strank, beleženje administrativnih akcij, arhiviranje naročil, itd. Na voljo je tudi množica plačljivih razširitev za vse tri različice, s katerimi razvijalec lahko močno poveča obseg funkcionalnosti spletne trgovine (cene se gibljejo od nekaj deset dolarjev do nekaj tisoč dolarjev). Gostovanja, namenjena sistemu za elektronsko trgovanje Magento, niso ravno poceni [44], vendar pa večinoma vsebujejo posebno optimizacijo za Magento, VPS in druge funkcionalnosti. Če ima naročnik veliko podjetje ter na voljo finančna sredstva za vlaganje v samo platformo (brez dodatnega razvoja), lahko dobi profesionalno spletno trgovino, gostovanje in podporo. **Število točk: 5**.

5. Analiza rezultatov

5.0.4 Povprečna ocena

Rešitev	Zend Framework	Drupal	Magento
Krivulja učenja	4,50	4,00	3,00
Čas realizacije	2,50	4,50	4,50
Nadgradnje	4,50	3,50	3,00
Prilagodljivost	5,00	4,00	3,00
Vzdrževanje	4,00	5,00	2,00
Spletna podpora	4,00	5,00	3,00
Hitrost	5,00	3,50	1,50
Obremenitev linije	3,50	5,00	2,50
Optimizacija	3,50	4,00	3,00
Plačljive opcije	3,00	3,00	5,00
Skupaj	**3,95**	**4,15**	**3,05**

Tabela 5.1: Ocene posameznih izvedb po kriterijih.

Glede na povprečno oceno kriterijev v tabeli 5.1 je zmagovalec Drupal, sledi Zend Framework, najslabši pa je Magento.

5.0.5 Ocena glede na zahteve stranke

Če razvijalec s strani naročnika ni omejen glede izbire načina izvedbe, je smiselen premislek načina izvedbe glede na zahteve. Elemente seznama zahtev naročnika smo sestavili glede na svoje izkušnje z razvijanjem spletnih aplikacij:

- čas (rok izvedbe, čas, ki ga ima razvijalec na voljo za izvedbo),
- finance (sredstva za razvoj, strežnik in vzdrževanje),
- količina različnih funkcionalnosti in
- dodelanost specifikacije ob začetku razvoja.

Za vsak element smo določili dve skrajnosti. Nato smo za vsako izmed kombinacij skrajnosti elementov zahtev sestavili nabor kriterijev, ki jih razvijalec upošteva pri odločitvi za določeno izvedbo. Tako smo dobil 2^4 različnih možnosti.

Če ima razvijalec na voljo malo časa, sta pomembna kriterija čas realizacije in krivulja učenja zavoljo čim hitrejše izvedbe. V nasprotnem primeru pa se lahko ukvarja tudi z obremenitvijo linije, optimizacijo, in zaradi daljšega razvojnega časa z nadgradnjami. Malo finančnih sredstev pomeni, da je bolj pomemben kriterij podpore, v nasprotnem primeru pa pridejo v poštev tudi plačljive opcije. Veliko število raznolikih funkcionalnosti lahko vpliva predvsem na hitrost aplikacije, tako da je na le-to potrebno biti pozoren. Na koncu pa pride na vrsto še dodelanost specifikacije ob začetku razvoja. Če naročnik točno ve, kaj hoče, potem načeloma ni problematična, v nasprotnem primeru pa je koristno upoštevati kriterija prilagodljivosti ter vzdrževanja, saj je ob konstantnem spreminjanju specifikacije potrebna čim večja prilagodljivost, poleg tega pa je večja verjetnost potrebe popravkov po uradnem zaključku projekta. Zato pride v poštev tudi vzdrževanje. Rezultati tega razmisleka so prikazani v tabeli 5.2.

	Čas		**Finance**		**Št. funkc.**		**Dodelanost spec.**	
	manj	**več**	**manj**	**več**	**manjše**	**večje**	**majhna**	**velika**
Krivulja učenja	✓	✗	✗	✗	✗	✗	✗	✗
Čas realizacije	✓	✗	✗	✗	✗	✗	✗	✗
Nadgradnje	✗	✓	✗	✗	✗	✗	✗	✗
Prilagodljivost	✗	✗	✗	✗	✗	✗	✓	✗
Vzdrževanje	✗	✗	✗	✗	✗	✗	✓	✗
Spletna podpora	✗	✗	✓	✗	✗	✗	✗	✗
Hitrost	✗	✗	✗	✗	✗	✓	✗	✗
Obremenitev linije	✗	✓	✗	✗	✗	✗	✗	✗
Optimizacija	✗	✓	✗	✗	✗	✗	✗	✗
Plačljive opcije	✗	✗	✗	✓	✗	✗	✗	✗

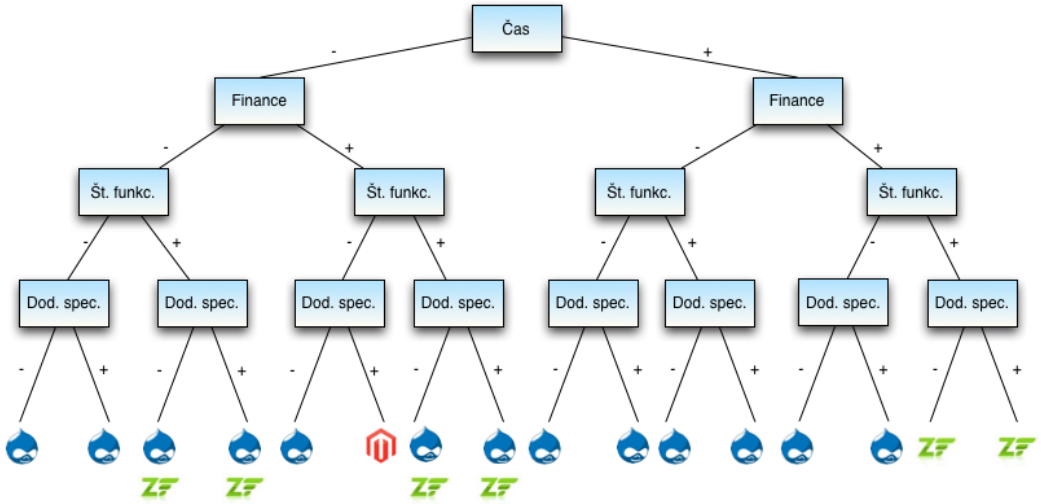
Tabela 5.2: Porazdelitev kriterijev primerjav po elementih naročniških zahtev.

Za vsak nabor kriterijev smo izračunali povprečno oceno. Rezultati so prikazani v tabeli 5.3.

#	Nabor kriterijev	Zend	Drupal	Magento
1	čas realizacije, krivulja učenja, spletna podpora, prilagodljivost, vzdrževanje	4,00	4,50	3,10
2	čas realizacije, krivulja učenja, spletna podpora	3,67	4,50	3,5
3	čas realizacije, krivulja učenja, spletna podpora, hitrost, prilagodljivost, vzdrževanje	4,17	4,17	2,83
4	čas realizacije, krivulja učenja, spletna podpora, hitrost	4,00	4,00	3,00
5	čas realizacije, krivulja učenja, plačljive opcije, prilagodljivost, vzdrževanje	3,80	4,10	3,50
6	čas realizacije, krivulja učenja, plačljive opcije	3,33	3,83	4,17
7	čas realizacije, krivulja učenja, plačljive opcije, hitrost, prilagodljivost, vzdrževanje	4,00	4,00	3,17
8	čas realizacije, krivulja učenja, plačljive opcije, hitrost	3,75	3,75	3,50
9	nadgradnje, obremenitev linije, optimizacija, spletna podpora, prilagodljivost, vzdrževanje	4,08	4,42	2,75
10	nadgradnje, obremenitev linije, optimizacija, spletna podpora	3,88	4,38	2,88
11	nadgradnje, obremenitev linije, optimizacija, spletna podpora, hitrost, prilagodljivost, vzdrževanje	4,21	4,29	2,57
12	nadgradnje, obremenitev linije, optimizacija, spletna podpora, hitrost	4,10	4,20	2,6
13	nadgradnje, obremenitev linije, optimizacija, plačljive opcije, prilagodljivost, vzdrževanje	3,92	4,08	3,08
14	nadgradnje, obremenitev linije, optimizacija, plačljive opcije	3,63	3,88	3,38
15	nadgradnje, obremenitev linije, optimizacija, plačljive opcije, hitrost, prilagodljivost, vzdrževanje	4,07	4,00	2,86
16	nadgradnje, obremenitev linije, optimizacija, plačljive opcije, hitrost	3,90	3,80	3,00

Tabela 5.3: Povprečne ocene izvedb vseh možnih naborov kriterijev primerjave glede na zahteve stranke.

Glede na povprečno oceno nabora kriterijev za vsako izmed možnosti v tabeli 5.2 smo izdelali tudi odločitveno drevo za potencialnega razvijalca, prikazano na sliki 5.1. Vozlišče je element seznama zahtev naročnika, ki se veji v dve skrajnosti. Desna veja pomeni pozitivno skrajnost, leva pa negativno (npr. pri času + pomeni več časa, - pa manj časa za izvedbo). V listih drevesa je priporočen način izvedbe.



Slika 5.1: Odločitveno drevo za pomoč razvijalcu pri izbiri načina izvedbe glede na zahteve naročnika.

Zend Framework pride v poštev, če je zahtevano večje število funkcionalnosti in ima razvijalec na voljo dovolj časa za razvoj, saj je bil pri primerjavi nahitrejši, vendar se je slabo odrezal pri merjenju časa realizacije. Več časa pomeni tudi upoštevanje kriterija nadgradenj, kjer je prav tako imel najboljšo oceno.

Drupal največkrat pride v poštev zaradi solidne ocene pri praktično vseh kriterijih. Slabše se odreže, če stranka zahteva večje število funkcionalnosti, saj pri primerjavi ni bil najhitrejši. Poleg tega v primeru, da je na voljo dovolj finančnih sredstev, ne ponuja profesionalne podpore na nivoju Magenta.

Magento glede na ocene pride v poštev samo pri enem izmed naborov zahtev, in sicer v primeru, da ima razvijalec na voljo malo časa, veliko finančnih sredstev, manjše število funkcionalnosti ter dodelano specifikacijo. Razlogov za to je več. Pri tem naboru sta pomembna kriterija časa realizacije in plačljivih opcij, pri katerih se je Magento odrezal najbolje, poleg tega pa se ne upoštevata kriterij prilagodljivosti in vzdrževanja zaradi dodelane specifikacije ter kriterij hitrosti zaradi manjšega števila funkcionalnosti.

6. Sklepne ugotovitve

Jedro diplomske naloge vsebuje primerjavo posameznih načinov izvedb po različnih kriterijih. Pomembnost le-teh je odvisna od zahtev naročnika spletne aplikacije. Zato je bil cilj diplomskega dela izdelava pomoči, namenjene razvijalcem pri izbiri načina izvedbe spletne aplikacije glede na zahteve naročnika.

Ker se posamezne izvedbe močno razlikujejo med seboj, samo povprečna ocena vseh kriterijev ni dovolj dobro merilo za izbiro. Zato smo sestavili še seznam pogostih naročniških zahtev. Glede na kombinacije različnih skrajnosti le-teh smo razdelili kriterije primerjave v posamezne nabore. Za vsakega izmed naborov smo izračunali povprečno oceno in s tem dobili seznam povprečnih ocen glede na naročniške zahteve ter tako dosegli cilj diplomskega dela. Na podlagi teh ocen smo izdelali še preprosto odločitveno drevo, preko katerega se lahko razvijalec lažje odloči za določen pristop k izvedbi. Poleg tega velja omeniti še to, da je celoten postopek primerjave dokaj abstrakten in tako uporaben tudi za druge tipe spletnih aplikacije.

V diplomskem delu bi se marsikaj dalo tudi izboljšati - sami kriteriji so na določenih mestih ocenjeni glede na naše izkušnje. Enako velja za tipične naročniške zahteve. Poleg tega bi pri sami izbiri naborov kriterijev lahko uporabljali uteži namesto izločanja, saj bi s tem dobili bolj realno oceno. Rezultat dela je torej osnova, na podlagi katere se lahko razvijalec lažje odloči za način izvedbe.

Slike

Tabele

Literatura

[1] Apache JMeter. Dostopno na:
http://jakarta.apache.org/jmeter/

[2] Arhiv verzij knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/download/archives

[3] Cake PHP. Dostopno na:
http://cakephp.org

[4] Codeigniter. Dostopno na:
http://codeigniter.com/

[5] Concurrent users estimation. Dostopno na:
http://www.scribd.com/doc/38577399/Concurrent-Users-Estimation

[6] Diagram podatkovnih tokov. Dostopno na:
http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.stf.powerdesigner.eclipse.docs_

[7] Drupal 6 API. Dostopno na:
http://api.drupal.org/api/drupal/6

[8] Drupal Forum. Dostopno na:
http://drupal.org/forum

[9] Drupal modul Upgrade Status. Dostopno na:
http://drupal.org/project/upgrade_status

[10] Drupal sistem kljuk. Dostopno na:
http://api.drupal.org/api/drupal/includes–module.inc/group/hooks/6

[11] Drupal storitve. Dostopno na:
http://drupal.org/drupal-services

[12] Elektronsko trgovanje. Dostopno na:
http://en.wikipedia.org/wiki/Electronic_commerce

[13] ExtJS. Dostopno na:
http://www.sencha.com/products/extjs/

[14] Prilagodljivost. Dostopno na:
http://en.wikipedia.org/wiki/Flexibility_(engineering)

[15] Html Purifier. Dostopno na:
http://htmlpurifier.org/

[16] Izvedba. Dostopno na:
http://www.islovar.org/izpisclanka.asp?id=6378&oznaci=1

[17] J. Redding, *Beginning Drupal*, Indianapolis: Wiley Publishing Inc.

[18] JavaScript. Dostopno na:
http://sl.wikipedia.org/wiki/JavaScript

[19] Joomla. Dostopno na:
http://www.joomla.org/

[20] KCachegrind. Dostopno na:
http://kcachegrind.sourceforge.net/cgi-bin/show.cgi

[21] Kaj je Ubercart. Dostopno na:
http://www.ubercart.org/what_is_ubercart

[22] Knjižnica. Dostopno na:
http://sl.wikipedia.org/wiki/Knjižnica_(računalništvo)

[23] Komponente knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/about/components

[24] Kompresija datotek na strežniku Apache z uporabo modula mod_deflate. Dostopno na:
http://httpd.apache.org/docs/2.0/mod/mod_deflate.html

[25] Logični podatkovni model. Dostopno na:
http://en.wikipedia.org/wiki/Logical_data_model

[26] *Magento User Guide*, What is Magento, Inubin Consulting Inc.

[27] Magento Wiki. Dostopno na:
http://www.magentocommerce.com/wiki/

[28] MySQL. Dostopno na:
http://sl.wikipedia.org/wiki/MySQL

[29] Nadgradnja. Dostopno na:
http://en.wikipedia.org/wiki/Upgrade

[30] Najnovejša verzija knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/download/latest

[31] Navodila za migracijo verzije knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/manual/en/migration.html

[32] Navodila za migracijo verzije sistema za upravljanje vsebin Drupal. Dostopno na:
http://drupal.org/project/migrate

[33] Navodila za nadgradnjo Drupal modula. Dostopno na:
http://drupal.org/node/672472

[34] Navodila za nadgradnjo sistema elektronskega trgovanja Magento do verzije 1.4.2.0. Dostopno na:
http://www.magentocommerce.com/wiki/1_-_installation_and_configuration/upgrading_magento

[35] Navodila za nadgradnjo sistema elektronskega trgovanja Magento na verzijo 1.6.0.0. Dostopno na:
http://www.magentocommerce.com/blog/comments/magento-preview-version-ce-1600-rc2-now-available/

[36] Navodila za nadgradnjo sistema za upravljanje vsebin Drupal. Dostopno na:
http://drupal.org/upgrade

[37] Navodila za uporabo knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/manual/en/

[38] Optimizacija. Dostopno na:
http://simple.wikipedia.org/wiki/Optimization_(computer_science)

[39] osCommerce. Dostopno na:
http://www.oscommerce.com/

[40] PHP. Dostopno na:
http://sl.wikipedia.org/wiki/PHP

[41] Page Speed. Dostopno na:
http://code.google.com/speed/page-speed/

[42] Pogosta vprašanja za Eclipse PDT. Dostopno na:
http://wiki.eclipse.org/PDT/FAQ

[43] Prijava hroščev za sistem elektronskega trgovajna Magento. Dostopno na:
http://www.magentocommerce.com/bug-tracking

[44] Primerjava gostovanj za Magento. Dostopno na:
http://magebenchmark.sonassi.com/

[45] Primerjava PHP knjižnic. Dostopno na:
http://www.phpframeworks.com/php-framework-comparison/

[46] Primerjava različic sistema za elektronsko trgovanje Magento. Dostopno na:
http://www.magentocommerce.com/product/compare#comparison-chart

[47] Primerjava sistemov za urejanje vsebin. Dostopno na:
http://www.cmsmatrix.org/matrix

[48] R. Allen, N. Lo, S. Brown, *Zend Framework In Action*, Manning Publications

[49] Sistem za upravljanje vsebin. Dostopno na:
http://sl.wikipedia.org/wiki/Sistem_za_upravljanje_vsebin

[50] Skriptni jezik. Dostopno na:
http://en.wikipedia.org/wiki/Scripting_language

[51] Spletna skupnost, ki razvijalcem omogoča sodelovanje pri razvoju knjižnice Zend Framework . Dostopno na:
http://framework.zend.com/community/contribute

[52] Spletna skupnost, ki razvijalcem omogoča sodelovanje pri razvoju sistema za upravljanje vsebin Drupal. Dostopno na:
http://groups.drupal.org

[53] Stackoverflow. Dostopno na:
http://stackoverflow.com/

[54] Symfony. Dostopno na:
http://www.symfony-project.org/

[55] TinyMCE. Dostopno na:
http://tinymce.moxiecode.com/

[56] Typo3. Dostopno na:
http://typo3.org/

[57] Uvod v zahteve knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/manual/en/requirements.introduction.html

[58] Učna krivulja. Dostopno na:
http://www.cockeyed.com/lessons/learning_curve/learning_curve.php

[59] Vmesnik za dostop do podatkovne baze Zend_Db_Adapter. Dostopno na:
http://framework.zend.com/manual/en/zend.db.adapter.html

[60] Vodiči za knjižnico Zend Framework. Dostopno na:
http://www.zftutorials.com/

[61] Webgrind. Dostopno na:
http://code.google.com/p/webgrind/

[62] Wordpress. Dostopno na:
http://wordpress.org/

[63] XAMPP za Mac OS X. Dostopno na:
http://www.apachefriends.org/en/xampp-macosx.html

[64] Zahteve sistema za elektronsko trgovanje Magento Commerce. Dostopno na:
http://www.magentocommerce.com/system-requirements

[65] Zahteve sistema za urejanje vsebin Drupal. Dostopno na:
http://drupal.org/requirements

[66] Zen Cart. Dostopno na:
http://www.zen-cart.com/

[67] Zend center za podporo. Dostopno na:
http://www.zend.com/en/support-center/

[68] Zend Framework forum. Dostopno na:
http://www.zfforums.com/

[69] Zend Framework gostovanja. Dostopno na:
http://framework.zend.com/wiki/display/ZFUSER/Zend+Framework+Web+Hosts

[70] Zend Framework: cena prilagodljivosti je kompleksnost. Dostopno na:
http://blog.fedecarg.com/2009/02/22/zend-framework-the-cost-of-flexibility-is-complexity/

[71] Zend podpora razvijalcem. Dostopno na:
http://www.zend.com/en/support-center/support

[72] Zend Server za Drupal. Dostopno na:
http://www.zend.com/en/solutions/packaged-php-applications/zend-server-drupal

[73] Življenski krog knjižnice Zend Framework. Dostopno na:
http://framework.zend.com/wiki/display/ZFDEV/Zend+Framework+Version+Lifecycle