Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Tadej Baša

Implementacija medijskega portala z uporabo odprtokodnega sistema za upravljanje z vsebinami Drupal

diplomsko delo
na univerzitetnem študiju

Mentor: izr. prof. dr. Viljan Mahnič

Ljubljana, 2012

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: 01868/2012
Datum: 05.09.2012

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

Kandidat: **TADEJ BAŠA**

Naslov: **IMPLEMENTACIJA MEDIJSKEGA PORTALA Z UPORABO ODPRTOKODNEGA SISTEMA ZA UPRAVLJANJE Z VSEBINAMI DRUPAL**

IMPLEMENTATION OF A MEDIA PORTAL USING THE OPEN SOURCE CONTENT MANAGEMENT SYSTEM DRUPAL

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Predstavite pomen odprte kode ter njene prednosti in slabosti pri razvoju uporabniških aplikacij. Pri tem namenite posebno pozornost odprtokodnemu sistemu za upravljanje vsebin Drupal, opišite njegovo strukturo in spremljajoče tehnologije. Nato prikažite možnosti, ki jih ta sistem nudi za razvoj portala "slovenskenovice.si". Opišite tehnološke rešitve za realizacijo uredniškega vmesnika in arhitekturne rešitve za zagotavljanje ustrezne odzivnosti in visoke stopnje dosegljivosti.

Mentor:
prof. dr. Viljan Mahnič

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:
prof. dr. Nikolaj Zimic

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Tadej Baša, z vpisno številko **63010005**, sem avtor diplomskega dela z naslovom:

Implementacija medijskega portala z uporabo odprtokodnega sistema za upravljanje z vsebinami Drupal

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Viljana Mahniča,
- 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 Braniku, dne 12. decembra 2012 Podpis avtorja:

Za pomoč, podporo in nasvete pri izdelavi diplomske naloge se zahvaljujem mentorju izr. prof. dr. Viljanu Mahniču. Hvala za čas in potrpežljivost pri pregledovanju tega izdelka.

Posebna zahvala gre mojim staršem, mami Andreji in očetu Alojzu, ki sta me dolgotrajnim študijem finančno in moralno podpirala.

Hvala tudi vsem sodelavcem za posredovano dragoceno znanje in izkušnje s področja spletnega razvoja.

Največjo zahvalo pa dolgujem ženi Tini, ki me je v zadnjih letih spodbujala in dajala moči za dokončanje študija.

Vsem mojim najbližjim

Kazalo

Povzetek

V diplomski nalogi je predstavljena implementacija spletnega portala "slovenskenovice.si" z uporabo odprtokodnih tehnologij in programske opreme; poudarek je na odprtokodni platformi za upravljanje z vsebinami Drupal. Opisani so ključni problemi, s katerimi smo se srečali pri prenovi medijskega portala z razmeroma veliko bazo uporabnikov. Prikazane so tako tehnološke rešitve na nivoju uredniškega vmesnika z uporabo prispevanih modulov platforme Drupal kot tudi arhitekturna zasnova sistema za dosego ustrezne zmogljivosti in visoke stopnje dosegljivosti.

Ključne besede:

Drupal, medijski portal, odprta koda, PHP, Apache, MySQL, Varnish

Abstract

The thesis presents the implementation of the web portal "slovenskenovice.si" using open source technologies and software with an emphasis on the open source platform for content management Drupal. Featured are key issues which were encountered in the renovation of the media portal with a relatively large user base. Described are technological solutions at the editorial interface level using contributed Drupal modules, as well as the system architecture designed to achieve an adequate performance and a high level of availability.

Key words:

Drupal, media portal, open source, PHP, Apache, MySQL, Varnish

Seznam uporabljenih kratic

- APC — Alternative PHP Cache: predpomnilnik za PHP-jevo ukazno kodo. Omogoča optimizacijo in predpomnenje vmesne kode PHP.
- API — Application Programming Interface: aplikacijski programski vmesnik.
- CMS — Content Management System: sistem za upravljanje z vsebinami.
- CSS, CSS3 — Cascading style sheets: prekrivni slogi, ki se uporabljajo za definicijo upodobitve spletne strani v spletnem brskalniku.
- CSV — Comma-separated values: format za besedilno datoteko, ki vsebuje z vejico ločene vrednosti.
- GPL — GNU General Public License: GNU splošna javna licenca.
- HTML, HTML5 — Hyper Text Markup Language: označevalni jezik za oblikovanje dokumentov.
- HTTP — HyperText Transfer Protocol: protokol za prenašanje hipertekst a.
- IRC — Internet Relay Chat: protokol za spletni klepet, eden najbolj razširjenih načinov skupinskega trenutnega sporočanja.
- JS — Javascript : objekti skriptni jezik namenjen ustvarjanju interaktivnih spletnih strani.

- JSON — JavaScript Object Notation: je enostaven in berljiv standard za zapis podatkov, izvira iz jezika JavaScript.
- LAMP — Linux, Apache, Mysql, PHP: pogosto uporabljen programski sklad za poganjanje spletnih aplikacij.
- PHP — PHP Hypertext Processor: splošnonamenski programski jezik, ki se pogosto uporablja za izdelavo dinamično generiranih spletnih strani.
- RSS — Really Simple Syndication: protokol, ki vzpostavlja okolje za objavo in distribucijo spletnih vsebin v XML-formatu.
- SQL — Structured Query Language: poizvedovalni jezik v relacijskih podatkovnih bazah.
- URL — Uniform Resource Locator: enolični naslov spletnega vira.
- XML — Extensible Markup Language: razširljiv označevalni jezik, ki nam omogoča format za opisovanje strukturiranih podatkov.

Poglavje 1

Uvod

Medijski in novičarski portali so med najbolj obiskanimi mesti na svetovnem spletu. Na lestvici najbolj obiskanih spletnih mest v Sloveniji je med prvimi dvajsetimi kar 12 takih, ki spadajo v to skupino.[1] Trend obiskanosti pa se še povečuje. Seveda iz tega sledi, da na tem "trgu" deluje močna konkurenca. Vsak izmed igralcev se (predvsem zaradi finančnih razlogov) trudi pridobiti čim več novih obiskovalcev in ob tem seveda graditi tudi mrežo zvestih uporabnikov. Za dosego teh ciljev pa je potrebno vsebine na spletnih portalih pogosto osveževati, generirati je treba veliko število novih vsebin, jih obogatiti s slikovnim in video materialom, vse čim tesneje povezati s hitro rastočimi socialnimi omrežji, ponujati interaktivne vsebine in še marsikaj ...

Ni težko ugotoviti, da je pri vsem tem treba imeti pravo orodje. Tako, ki na eni strani urednikom kar se da poenostavi objavljanje člankov in multimedijskih vsebin, na drugi strani pa uporabnikom omogoča kar najbolj zanimivo brskanje in branje.

Ponudba programske opreme za te namene je na trgu precej bogata. Aplikacije iz te skupine so bolj poznani pod kratico CMS (Content Management System). Med najbolj poznanimi in uporabljenimi najdemo: Wordpress, Joomla, Typo in, predvsem v zadnjih letih, tudi Drupal. Pomanjkljivost vseh naštetih aplikacij je, da vsaka ponuja dokaj zaprt nabor funkcional-

[1] http://www.moss-soz.si/si/rezultati_moss/obdobje/default.html?period=201209

nosti. Seveda te v večini primerov zadostujejo, saj so uporabniki takih aplikacij v največjem številu blogerji ali mala in srednja podjetja, ki se na spletu želijo tako ali drugače predstaviti. Popolnoma drugačen pa je primer uporabe takega sistema v velikih in dinamičnih uredniških skupinah, kot jih srečamo v primeru medijskih in novičarskih portalov. Tu taki sistemi le redko popolnoma ustrezajo željam in pričakovanjem njihovih uporabnikov. Zato se vsakemu založniku postavi vprašanje: razvoj lastnega sistema za urednikovanje ali prilagoditev in nadgradnja obstoječe rešitve?

Nemalo se jih odloča za prvo možnost (samo na slovenskem trgu: 24ur.com, rtvslo.si itd.). V pričujočem delu poskušamo pokazati, da je druga možnost veliko bolj smotrna. Pomembno pa je, kakšno platformo vzamemo za izhodišče: kakšne so možnosti za razširitev/dodajanje novih in spreminjanje obstoječe funkcionalnosti, kakšne so njene performančne lastnosti, možnosti integracije z zunanjimi sistemi, uporabljene tehnologije itd.

V pričujočem delu bom predstavil primer razvoja multimedijskega portala skoraj izključno z uporabo odprtokodnih rešitev. Naloga je v grobem sestavljena iz treh sklopov. V 2. poglavju je na kratko predstavljena odprta koda, njen pojav in razvoj ter njene prednosti in slabosti.

Nadaljujem z drugim delom v 3. poglavju, v katerem predstavim odprtokodno platformo za razvoj spletnih aplikacij Drupal, ki je bila uporabljena kot temelj za implementacijo prenovljenega portala “slovenskenovice.si”. Na kratko je predstavljena njegova struktura in komplementarne tehnologije.

V zadnjem delu diplomske naloge se podrobneje posvetim implementaciji portala “slovenskenovice.si”. V 4. poglavju najprej predstavim okvire projekta. Opišem tudi metodologijo razvoja (poglavje 4.2) in se nato v poglavju 4.3 posvetim opisu uporabe platforme Drupal. Vsebinski sklop zaključim z opisom infrastrukture, na kateri je postavljen prenovljeni portal.

Poglavje 2

Odprtokodni programi

Izraz "odprta koda" (angl. *open-source*) se nanaša na filozofijo, ki spodbuja prosto širjenje in dostop do načrta ter implementacijskih podrobnosti nekega produkta. Čeprav je sama ideja odprte kode že zelo stara (v neki drugi obliki jo na primer srečamo pri kuharskih receptih), je s pojavitvijo interneta dobila še poseben zagon, in to prav na področju računalniške programske opreme.

Odprtokodni program (angl. *Open-source software*) je računalniški program, ki je uporabniku na voljo v obliki izvorne kode. Ta je ponavadi objavljena pod takimi licenčnimi pogoji, ki uporabniku dovoljujejo njeno proučevanje, prilagajanje, spreminjanje in nadaljnjo distribucijo. Za odprtokodne programe je značilna tudi podpora odprtih standardov in protokolov ter hitro rastoča množica uporabnikov in razvijalcev.

Med idejnimi očeti odprte kode je najpomembnejši gotovo Richard Stallman, ki je v začetku 80. let prejšnjega stoletja zasnoval svoj operacijski sistem, kompatibilen s takrat popularnim lastniškim operacijskim sistemom Unix. Odločil se je, da bo njegov operacijski sistem brezplačen in da bo na voljo tudi njegova izvorna koda ter izvorna koda vseh v njem vključenih programov. Poimenoval ga je GNU.[2] V svojem manifestu [8], ki je izšel marca 1985, je Stallman podal razlage in definicije ciljev projekta GNU. Kmalu za tem je nastala še splošna javna licenca GNU (GPL), katere namen je

[2]GNU je rekurzivni akronim, ki pomeni "GNU is NOT UNIX".

preprečiti, da bi bilo mogoče odprtokodne programe spremeniti v lastniške. Skupaj z ljudmi, ki so delali na projektu GNU, je Stallman leta 1985 ustanovil še neprofitno organizacijo Free Software Foundation (FSF)[3]. Ta naj bi upravljala s poslovnimi vidiki projekta GNU, skrbela za donacije, prodajanje kopij *prostega programja* ter nudenje drugih storitev. Organizacija FSF zagovarja uporabo izraza *prosto programje*. Po njihovem mnenju je prosto programje stvar pravic v smislu "svobode govora" in ne cene. Nanaša se na možnost uporabnika, da programe poganja, preučuje, popravlja in izboljšuje ter distribuira [18].

Z bolj pragmatičnimi kot filozofskimi vidiki širjenja ideje odprte kode se ukvarja leta 1998 ustanovljena organizacija Open Source Initiative[4]. Ta promovira primere odprte kode pri podjetjih in jih poskuša prepričati, da bi ideje posvojil ter skrbi za certificiranje odprtokodnih licenc in na ta način zagotavlja pravilno uporabo pojma odprta koda.

Razlika med gibanjem za prosto programje in gibanjem za odprto kodo je v njunih vrednotah, pogledu na razvoj programov in licenčnih zahtevah. Gibanje za odprto kodo dovoljuje več svobode pri licencah (ali naj bo programska oprema odprtokodna, je praktično vprašanje in ne etično). Za gibanje za prosto programje predstavlja lastniško programje družbeni problem, prosto programje pa njegovo rešitev.

V nadaljevanju naloge se bom omejil le na uporabo izraza odprta koda v smislu programske opreme, za katero je na voljo izvorna koda, ki jo je mogoče svobodno uporabljati, spreminjati in razširjati naprej.

Za odprtokodnimi programi velikokrat stoji mreža prostovoljcev, ki sodelujejo pri njihovem razvoju in vzdrževanju. Zgledni primeri odprtokodne programske opreme so:

- Apache HTTP stežnik
- Spletni brskalnik Mozilla Firefox
- Operacijski sistem GNU/Linux

[3] http://www.fsf.org/

[4] http://www.opensource.org

Ker odprtokodne programe ponavadi razvija velika skupina programerjev, živečih po vsem svetu, so za razvoj potrebna orodja, ki omogočajo in olajšujejo tako delo. Pogosto so uporabljeni programi za spremljanje različic izvorne kode, orodja za avtomatsko testiranje, prevajanje, spremljanje prijav o napakah, vsakodnevno komunikacijo. Vsa ta orodja omogočajo, da sodelujoči vzdržujejo splošen nivo kakovosti, stabilnosti in tehnične podpore.

Nedvomno postaja odrta koda vedno bolj popularna, saj principe posvajajo tudi že velika tuja podjetja, med katerimi najdemo IBM[5], Google[6], Facebook[7] ... Rezultati ankete 2012 Future of Open Source Survey [14] kažejo, da kar 59 % podjetij v svojih produktih uporablja odprtokodne rešitve. Na področju spletnih tehnologij pa je med najpopularnejšimi prav Drupal CMS [15, 17].

2.1 Prednosti in slabosti odprotokodnih programov

2.1.1 Prednosti

Raziskave [13, 3], kažejo, da posamezniki in podjetja v odprtokodnih programih vidijo naslednje prednosti:

- **Zanesljivost**: To je odsotnost napak v programu. Zaradi pogostih izdaj novih verzij odprtokodni programi omogočajo pridobivanje kvalitetnih povratnih informacij uporabnikov (med njimi so tudi prijave napak). Dostopnost izvorne kode pa omogoča, da uporabniki napako, ki jo odkrijejo, kar sami tudi odpravijo in po možnosti o tem obvestijo vzdrževalca ter posredujejo popravek.

- **Stabilnost**: Uporabniki se izogibajo nadgradnjam programske opreme, če ta služi njihovim potrebam. Proizvajalci lastniške programske opreme,

[5] http://www-03.ibm.com/linux/ossstds/oss/ossindex.html
[6] http://code.google.com/opensource/
[7] https://developers.facebook.com/opensource/

ki si želijo stalnih prihodkov, uporabljajo različne prijeme, s katerimi želijo svoje uporabnike prepričati ali prisiliti v nadgradnje (sprememba formatov, odprava hroščev, umik tehnične podpore za starejše različice). Odprtokodno programje je prilagojeno odprtim standardom, ki se zelo redko spreminjajo, zato spremembe v programski opremi niso potrebne oziroma se uporabnikov v njih ne sili. Uporabniki imajo možnost, da se odločijo, ali bodo ostali na starejši različici ali prešli na novo verzijo programa.

- **Preglednost**: Uporabniki lastniške programske opreme morajo zaupati trditvam razvijalcev o varnosti, neobstoju nedokumentiranih, skrivnih načinov dostopa do programa — stranskih vrat (angl. *backdoor*), prilagojenosti standardom, fleksibilnosti in ostalih kvalitetah programske opreme. Če je izvorna koda programa objavljena, se o tem uporabnik lahko prepriča sam ali pa za to najame neko tretjo osebo.

- **Nizki stroški**: Pri izbiri programske opreme je eden ključnih kriterijev višina skupnih stroškov lastništva (angl. *Total Cost of Ownership*) (TCO). TCO za prosto programje je pogosto nizek zaradi velikokrat ničnih stroškov nakupa, brezplačne uporabe dodatnih kopij programa, manjše potrebe po nadgradnjah (oziroma so te brezplačne), daljšega delovanja brez izpadov (nižji stroški za sistemske administratorje), manjše ranljivosti na viruse (kar izloči potrebo po nakupu protivirusne programske opreme), manjšega števila varnostnih lukenj in nižje ranljivosti na vdore (znižuje stroške sistemske administracije), možnosti uporabe starejše oziroma manj zmogljive strojne opreme (nižji stroški nabave strojne opreme).

- **Fleksibilnost**: Fleksibilnost programske opreme pomeni svobodo pri izbiri rešitve, ki bo zadovoljila vse potrebe uporabnikov. Spremembe v poslovnem okolju ne smejo biti omejene s programom, ki podpira poslovanje. Še posebej je to pomembno na nivoju arhitekture informacijske rešitve. Če posamezne programske rešitve podpirajo preizkušene

standarde za vzajemno delovanje, se je mogoče izogniti preveliki odvisnosti od določenega proizvajalca programske opreme. Prosto programje ponuja svojim uporabnikom večjo svobodo pri izbiri drugih produktov. Če neka programska rešitev ne ustreza vsem zahtevam uporabnika, jo je glede na dostopnost izvorne kode vseeno mogoče ustrezno prilagoditi.

- **Podpora**: Uporabniki in razvijalci na odprtokodnih projektih delujejo na osnovi skupnih interesov in potreb po določeni programski opremi. Zato so oboji velikokrat pripravljeni vložiti nekaj svojega časa tudi za pomoč drugim. Pomoč in odgovore na vprašanje je mogoče dobiti na raznih forumih, novičarskih skupinah, klepetalnicah itd.

2.1.2 Slabosti

Velike multinacionalke, ki v odprti kodi vidijo velikega konkurenta, poskušajo prosto programje velikokrat diskreditirati prav z argumentom, da gre za slabo spisane programe, ki jih uporabljajo le "hekerji" in "zastonjkarji". Negirajo celo v prejšnjem poglavju napisane prednosti. Objektivno gledano lahko prostemu programju očitamo naslednje probleme:

- **Nekompatibilnost**: Prosto programje ne deluje vedno najbolje z drugimi aplikacijami, na primer v okolju *Microsoft Windows*.

- **Neprijazni uporabniški vmesniki**: Prosto programje večinoma izgrajujejo inženirji za inženirje. Uporabniški vmesniki zato pogosto niso dovolj prilagojeni nezahtevnim uporabnikom. Ti končni uporabniki ne sodelujejo v procesu razvoja odprtokodne aplikacije in posledično razvijalcem ne posredujejo povratnih informacij. Posledica tega je, da imajo odprtokodni programi pogosto manj pregledne in manj atraktivne uporabniške vmesnike.

- **Slaba podpora končnim uporabnikom**: Čeprav smo dobro podporo že našteli med prednostmi prostega programja, velja omeniti, da

je dostopna dokumentacija včasih končnim uporabnikom težko razumljiva, saj je bolj namenjena razvijalcem. Običajno gre za tehnično podrobno dokumentacijo, ki zahteva določena predznanja. Velikokrat je pomanjkljiva in neažurna zaradi pogostih izdaj programja.

- **Pomanjkanje lastništva in odgovornosti**: Podjetja, ki stojijo za lastniško programsko opremo, lahko jamčijo za delovanje svojih produktov, njihovo kompatibilnost s starejšimi verzijami in varnost podatkov. V nasprotju z njimi pa prosto programje za sabo nima neke entitete, ki bi nosila pravno in finančno odgovornost.

Poglavje 3

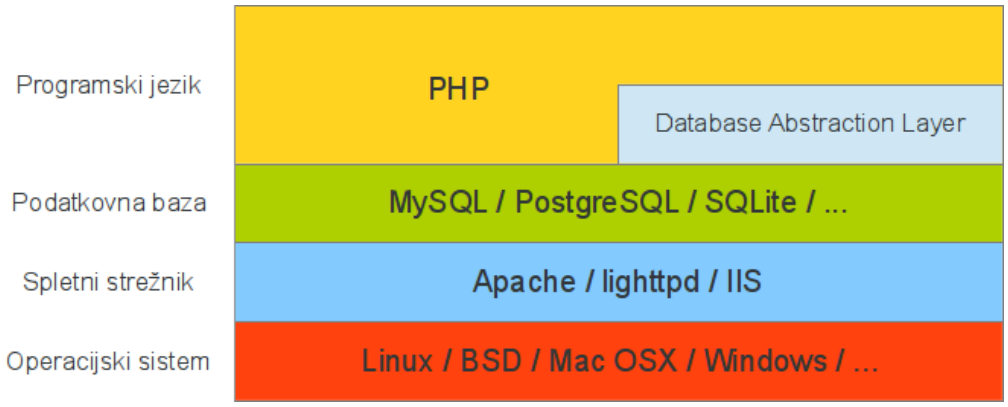
Drupal

Drupal je odprtokodni sistem za upravljanje z vsebino. Spisan je v programskem jeziku PHP in distribuiran pod licenco GPL. Odlikuje ga visoka stopnja modularnosti, čista in robustna programska koda s svojevrstnim sistemom tako imenovanih "kljuk" (angl. *hooks*), z implementacijo katerih moduli razširjajo funkcionalnost osnovnega sistema. Za Drupalom stoji močna programerska skupnost, organizacija Drupal Association, večja in manjša podjetja ter posamezniki. Drupal Association je neprofitna organizacija, ki skrbi za promocijo platforme, sodeluje pri organizaciji dogodkov (konference, izobraževanja ...), vzdržuje domačo stran drupal.org ...

Drupalova spletna skupnost razvijalcev, oblikovalcev, administratorjev in ostalih uporabnikov nudi uporabnikom podporo preko forumov[8], interesnih skupin[9], številnih poštnih seznamov[10] in prek omrežja Internet Relay Chat (IRC)[11].

Posebna skupina razvijalcev - Drupal security team[12] - je zadolžena za beleženje in odpravljanje prijavljenih varnostnih pomanjkljivosti.

[8] http://drupal.org/forum
[9] http://groups.drupal.org/
[10] http://drupal.org/mailing-lists
[11] http://drupal.org/irc
[12] http://drupal.org/security-team



Slika 3.1: Tehnologije, ki poganjajo Drupal.

3.1 Tehnologije, ki poganjajo Drupal

Drupal se za svoje delovanje zanaša na sklop tehnologij, ki omogočajo poganjanje na cenejših spletnih gostovanjih, po drugi strani pa neverjetno skalabilnost.

Drupal ni odvisen od operacijskega sistema, na katerem ga želimo uporabljati. Deluje praktično na vsakem sistemu, na katerem deluje tudi programski jezik PHP Hypertext Processor (PHP).

3.1.1 PHP

Drupal je napisan v programskem jeziku PHP [11]. PHP je široko podprt skriptni jezik, ustvarjen z mislijo na splet. Najnižja različica, na kateri teče Drupal 7.x, je PHP 5.2. Smiselno je omeniti, da kljub temu da je PHP objektno usmerjen programski jezik, je samo jedro Drupala v večji meri spisano v proceduralni obliki. V nasprotju z objektno usmerjenimi programi, ki temeljijo na vmesnikih in razredih, so Drupalovi moduli sestavljeni iz množice funkcij, ki sledijo dogovorjeni shemi poimenovanja. Ne glede na zgoraj zapisano se v jedru Drupala in prispevanih modulih pogosto najdejo tudi podsistemi, spisani na objektno usmerjen način.

3.1.2 Podatkovna baza

Drupal potrebuje za hranjenje podatkov bazo podatkov. Vmesnik za ta del je vsebovan v Plasti za abstrakcijo podatkovne baze. Ta vsebuje API, ki temelji na knjižnici PHP data object (PDO) in omogoča spisati vmesnik za podporo vsem podatkovnim bazam, ki podpirajo PHP. Samo jedro vsebuje podporo za MySQL, PostgreSQL in SQLite.

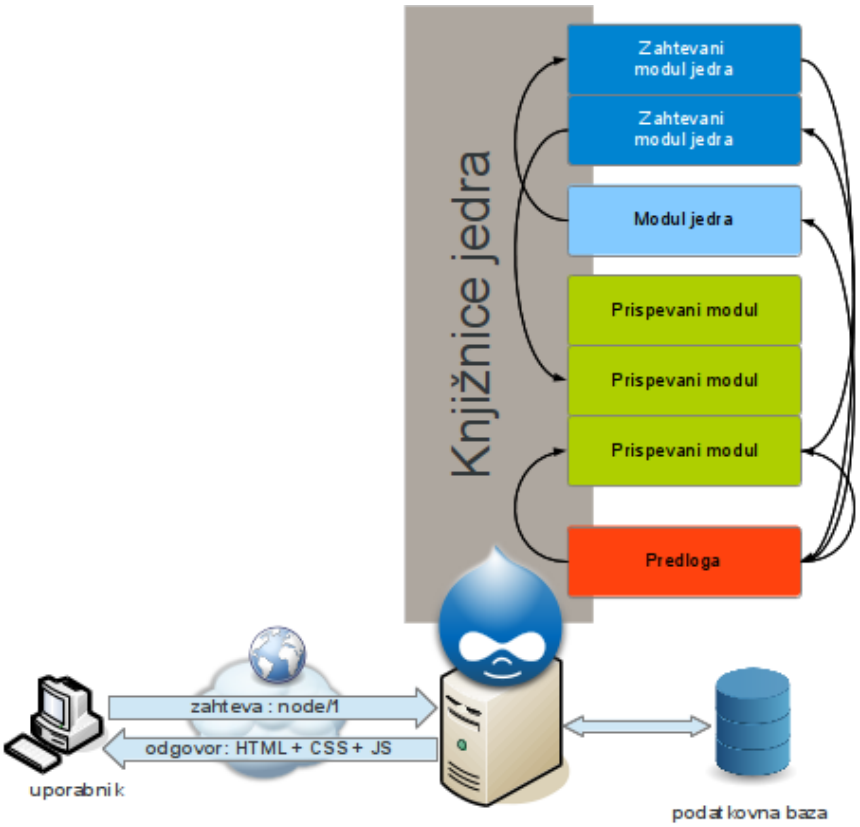
3.1.3 Spletni strežnik

Drupal v veliko pogledih temelji na zmožnostih strežnika Apache, ki je med najbolj široko uporabljenimi spletnimi strežniki na svetu [16]. Kljub temu lahko Drupal z določenimi prilagoditvami poganjamo tudi na drugih spletnih strežniki, vključno z Microsoft IIS, nginx, lighttpd ...

3.2 Arhitektura Drupala

Na sliki 3.2 je predstavljena groba struktura Drupal. Nakazuje tudi, kako Drupal streže zahtevam. Nekoliko podrobneje lahko tipično zahtevo opišemo z naslednjimi koraki:

1. Uporabnik v brskalnik vnese URL spletnega mesta `http://spletno mesto.com/node/1`.
2. Brskalnik sporoči spletnemu strežniku, da zahteva vir `/node/1`.
3. Spletni strežnik ugotovi, da mora zahtevo posredovati okolju PHP in mu jo preda.
4. PHP izvrši Drupalovo datoteko `index.php` in ji posreduje pot `/node/1`.
5. Jedro Drupala izvrši zagon, inicializira potrebne resurse in z uporabo menijskega podsistema ugotovi, kako obdelati `/node/1`.
6. Zažene se modul *Node*, ki odgovori za zahtevo tako, da naloži vsebino s številko 1 (ponavadi iz podatkovne baze) in preda informacijo podsistemu predlog.



Slika 3.2: Struktura CMS Drupal.

7. Podsistem predlog informacijo spremeni v obliko primerno za prikaz uporabniku (podatke spremeni v obliko HTML s pripetimi datotekami CSS in Javascript (JS)).
8. Jedro Drupala zaključi z obdelavo in vrne podatke odjemalcu.
9. Brskalnik prejme odgovor in podatke predstavi v obliki, vidni uporabniku.
10. Uporabnik na ekranu vidi zahtevano vsebino.

Opisani tok dogodkov je sicer najpogostejši, ni pa to edini način, na katerega Drupal streže zahteve. Izpuščenih je tudi veliko podrobnosti. Jedro Drupala in moduli namreč med obdelovanjem zahteve sprožajo kljuke in tako drugim modulom omogočajo spreminjanje podatkov, toka obdelave podatkov

ali celo toka streženja zahtevi.

3.2.1 Knjižnice jedra

Knjižnice jedra zagotavljajo osnovne funkcije in storitve za delovanje modulov. Poenostavljajo komunikacijo s podatkovno bazo, prevajanje vmesnika, graditev formularjev, kodiranje podatkov. So orodja, ki programerjem olajšujejo procesiranje podatkov, vendar ne skrbijo v celoti za rokovanje z zahtevami.

3.2.2 Moduli

Moduli so programski dodatki za Drupal, ki razširjajo, dograjujejo ali izboljšujejo funkcionalnost sistema Drupal.

Visoka stopnja modularnosti se kaže že v tem, da je tudi samo jedro Drupala sestavljeno iz modulov. Nekateri so obvezni (npr. *System*) in jih ni mogoče izklopiti. Nekateri pa so opcijski (npr. *Tracker* - sledenje objavam uporabnikov). Tako kot ostali moduli, tudi moduli jedra delujejo prek implementacije t. i. kljuk. Ko Drupal kliče kljuke, vsak modul jedra po potrebi odgovori in izvede osnovne funkcije v točno določenih trenutkih med procesiranjem zahteve.

3.2.3 Kljuke

Jedro Drupala ne poskuša izvajati operacij na vsakem koraku obdelovanja zahteve. Namesto tega modulom ponudi priložnost, da opravijo ta korak oziroma da se "zaključkajo" v življenjski cikel zahteve. Za primer si poglejmo korak, v katerem Drupal preverja, ali kateri od modulov zahteva posebno inicializacijo. Drupal v tem koraku izvede `hook_init()`. To v praksi pomeni, da pregleda vse nameščene module in preveri, ali kateri implementira kljuko `hook_init()`. Moduli implementirajo kljuke tako, da sledijo vzorcu poimenovanja funkcij. V tem primeru bi modul z imenom *Test* implementiral funkcijo `test_init()`. Tako med življenjskim ciklom zahteve Drupal

pokliče vrsto kljuk (kljuke lahko kličejo tudi ostali moduli), dokler ne doseže zadnjega koraka izvajanja, v katerem se pokliče kljuka `hook_exit()`. Po zaključku izvajanja te kljuke Drupal posreduje obdelane podatke odjemalcu in zaključi zahtevo.

3.2.4 Predloge

Drupal loči predstavitveni del od vsebine prek sistema predlog. Vsako spletno mesto se tako končnemu obiskovalcu predstavi v določeni grafični obliki, ki je implementacija neke predloge.

Del sistema predlog se nahaja v knjižnicah jedra. Te skrbijo za inicializacijo sistema, zaznavanje funkcij aktivne predloge in datotek s predlogami ter odločanje, katere predloge se v določenem trenutku aplicirajo. Večina kode kljub temu živi v modulih.

Predloga je strukturiran paket programske kode (podobno kot modul), ki skrbi za pretvorbo surovih podatkov v oblikovan izhod.

3.2.5 Prispevani moduli

Ena največjih prednosti platforme Drupal je njegova modularnost, možnost razširitve osnovne funkcionalnosti. Funkcionalnost, ki je vključena v samo jedro sistema, komaj zadostuje za postavitev osnovnih spletnih strani (blogov, spletnih brošur). Tako kompleksnega portala, kot so slovenskenovice.si, brez velikega časovnega vložka v programiranje razširitev, ni mogoče vzpostaviti. Tukaj se pokaže moč odprtokodnih skupnosti. Skupina zainteresiranih uporabnikov in razvijalcev platforme Drupal se nenehno širi (v času pisanja te naloge šteje skupnost 909676 uporabnikov[13] od tega jih je 22308 prispevalo tudi kodo). Domača stran drupal.org gosti tudi repozitorij prispevanih modulov, razširitev, ki jih izdelajo uporabniki Drupala in jih pod odprtokodno licenco posredujejo nazaj skupnosti (na dan 5. 12. 2012 smo lahko na drupal.org/project/modules našli 19408 modulov). Nekateri med njimi

[13]Podatek z naslovnice http://www.drupal.org na dan 5. 12. 2012.

pokrivajo zelo enostavne funkcionalnosti, nekateri pa so kompleksni paketi, ki po velikosti parirajo samemu jedru. Modulov je toliko, da se je v skupnosti razvila govorica "there's a module for that", kar naj bi pomenilo: kakršna koli potreba se pojavi, že obstaja prispevani modul, ki rešuje ta problem. V resnici seveda ni ravno tako. Prispevanih modulov je res ogromno število, vendar se velikokrat izkaže, da problem rešujejo ravno malo drugače, kot bi si sami želeli. Funkcionalnost prispevanih modulov je tako pogosto treba nekoliko prikrojiti našim potrebam. Na srečo je Drupal taka platforma, ki nam to na tak ali drugačen način omogoči. Nekoliko za šalo naj v tem kontekstu omenim še drugo karakterizacijo Drupala: "Drupal is like a box of Lego. You never know what piece is going to be missing" (Drupal je kot set Lego kock. Nikoli ne veš, kateri košček bo zmanjkal).

S pomočjo prispevanih modulov lahko v zelo kratkem času postavimo precej kompleksne konfiguracije. V poglavju 4.5 bom na kratko opisal tudi nekatere izmed modulov, ki so bili uporabljeni pri implementaciji portala slovenskenovice.si in so nam omogočili, da smo portal postavili v rekordno kratkem času.

3.2.6 Namestitveni profili

Drupal pozna tudi koncept t. i. "namestitvenih profilov". To so posebni programski paketi, ki združujejo jedro Drupala in več dodatnih prispevanih modulov. Predstavljajo zaključeno rešitev za neko določeno uporabniško zahtevo (npr. osebni blog, portfelj, spletna trgovina ...). Paket ponavadi implementira razširjeno namestitveno proceduro, ki poskrbi za vso potrebno konfiguracijo. Po zaključku namestitve dobi uporabnik na voljo delujoč sistem.

3.3 Glavni podsistemi

3.3.1 Vsebine

Verjetno najpomembnejši podsistem, ki ga moramo poznati za delo z Drupalom, je podsistem *vozlišč* (angl. *node*). *Vozlišče* je osnovni vsebinski element, ki ga lahko objavimo in prikažemo. Vsebuje lahko več polj različnih tipov: ponavadi vsaj naslov, glavno besedilo in razne metapodatke (datum objave, avtor ...).

V Drupalu lahko definiramo različne *tipe vsebine* (angl. *content types*) glede na to, kakšno informacijo želimo predstaviti (članek, blog, informacije o glasbeni plošči ...). Vsak tip vsebine lahko vsebuje različen nabor vsebinskih polj (članek ima na primer besedilo, naslovno sliko, kategorijo ...). Podsistem vsebin je implementiran v modulu *Node*.

3.3.2 Meniji

Drupal poleg vsebine hrani tudi informacijo o strukturi spletnega mesta. Glavni način za definiranje strukture je z uporabo podsistema menijev. Ta podsistem vsebuje programski vmesnik za ustvarjanje, branje in spreminjanje elementov, ki opisujejo zgradbo spletnega mesta. Skrbi za sistemske navigacijske menije. Meniji so hierarhični elementi - z njimi lahko ustvarimo drevesno strukturo. Na tak način lahko spletno mesto razdelimo na posamezne sekcije in podsekcije. Podsistem menijev je implementiran v modulu *Menu*.

3.3.3 Uporabniki

Drupal ni načrtovan zgolj kot CMS, ampak tudi kot platforma za družbene medije. Zato v svojem jedru vsebuje tudi podsistem za upravljanje z uporabniki. Podsistem za delo z uporabniki omogoča razvijalcem, da nadzorujejo praktično vsak vidik življenjskega cikla uporabnika: način registracije, katera polja se pojavijo v uporabnikovem profilu, katere pravice ima v sistemu ... Podsistem uporabnikov je implementiran v modulu *User*.

3.3.4 Komentarji

Komentiranje je verjetno najpogostejša funkcionalnost vsakega družbenega spletnega medija. Drupal vsebuje podsistem, ki omogoča komentiranje vsake vsebine na osnovi *vozlišča*.

3.3.5 Entitete in polja

V Drupalu do verzije 7.x je bil podsistem *vozlišč* edini način za ustvarjanje strukturiranih enot vsebine. Z verzijo 7.x sta bila v jedro Drupala uvedena podsistema polj (angl. *fields*) in entitet (angl. *entities*). Podsistem entitet prispeva programski vmesnik za definiranje novih tipov strukturiranih podatkov, ki niso vozlišča. Programski vmesnik podsistema polj pa omogoča definicijo različnih tipov polj (na primer datumsko polje, polje za sliko ...), ki jih lahko pripnemo entitetam.

3.3.6 Bloki

Večina spletnih mest poleg glavne vsebine vsebuje tudi dodatke, ki se prikazujejo ob osrednji vsebini (bodisi ob zgornjem, stranskih ali spodnjem robu). Podsistem blokov (angl. *blocks*) upravlja s konfiguracijo in prikazom teh vsebinskih enot. Podsistem blokov je implementiran v modulu *Block*.

3.3.7 Taksonomija

Taksonomija, na splošno veda o razvrščanju, je podsistem CMS Drupal namenjen kategorizaciji vsebine. Razdeljena je na dva glavna elementa: slovarje (angl. *vocabulary*) in têrmine (angl. *term*). Slovarje si lahko predstavljamo kot množice sorodnih besed in besednih zvez. Če bi na primer ustvarili slovar "Športi", bi vanj verjetno uvrstili besede "tenis", "nogomet", "rokomet" ... Vsaka od teh besed v slovarju je têrmin. Ko v Drupalu ustvarimo slovar, lahko na tipih vsebine določimo novo polje, prek katerega označimo vsebino z besedami iz tega slovarja (vsebino kategoriziramo).

Poglavje 4

Implementacija spletnega portala "slovenskenovice.si"

Največje slovensko časopisno in založniško podjetje Delo, d.d., je z letom 2011 začelo prenovo svojih spletnih portalov. Stroški vzdrževanja portalov, zgrajenih na lastni platformi, so postali previsoki. Potreben je bil prehod na novo platformo. Cilji, ki jih je vodstvo želelo doseči, so bili: neodvisnost od enega podjetja, nizki stroški nabave, razširljivost, možnosti za pridobitev tehnične podpore pri razvoju, dosegljivost domačih strokovnjakov.

V ožjem izboru je pristalo več odprtokodnih platform. Odločitev za Drupal pa je padla ravno zaradi njegove velike in v tistem času naraščajoče popularnosti, velike skupnosti razvijalcev, ogromnega števila že razvitih modelov, možnosti za pridobitev tako domačih razvijalcev kot tudi možnost najema strokovnega svetovanja podjetij iz tujine.

V avgustu 2010 se je začel poskusni razvoj na (novih) manjših straneh časopisnih prilog (pogledi.si, polet.si, deloindom.si). Razvoj portala "slovenskenovice.si" (oziroma bolj natančno, prenova obstoječega portala) se je začela v maju 2011. Na projektu je sodelovalo 7 ljudi, od tega 5 programerjev. Oblikovno predlogo je pripravil zunanji sodelavec. Projekt je obenem tudi pilotno uvedel SCRUM kot metodologijo dela [10]. Projekt se je zaključil z objavo v poznih večernih urah 1. 12. 2011.

4.1 Opis zahtev

Uredništvo portala "slovenskenovice.si" ni skrivalo velikih pričakovanj in svojih ambicij. S ciljem, da se v enem letu prebijejo v vrh najbolj obiskanih spletnih portalov v Sloveniji, je ustvarilo na razvijalce in vodjo projekta dobršno mero pritiska. Prenovljeni portal mora biti sodoben, s privlačno obliko, da se uporabniki nanj vračajo, in vsebinsko bogat z razvejano hierarhijo kategorij. Podajati mora množico raznovrstnih informacij: od člankov do fotogalerij, video vsebin, vremenskih podatkov do vsebin partnerskih portalov. Omogočati mora čim večjo angažiranost bralcev z možnostjo komentiranja, glasovanja v anketah, sodelovanja v nagradnih igrah, možnost objave prispevkov, čestitk, prijave napak, predlogov, fotografij ...

Vse vsebine morajo biti arhivirane in enostavno dosegljive s hitrim iskalnikom. Portal mora biti hiter in odziven tudi v času povečanega obiska. Uporabljene morajo biti najnovejše spletne tehnologije (HTML5, CSS3), uporabniku pa omogočena vrhunska izkušnja.

Administracijsko okolje mora biti za urednika čimbolj intuitivno. Delovati mora hitro in piscem člankov nuditi orodja, ki jim omogočajo, da čim hitreje in učinkoviteje opravijo svoje delo.

4.2 Potek razvoja

Po zajemu osnovnih zahtev, ki jih je v tesnem sodelovanju z uredništvom starega portala izvedel vodja projekta, izdelavi skeletov in oblikovnih predlog je sledila faza implementacije.

Na podlagi dokumentacije, izdelane v fazi zajema zahtev, smo lahko sklepali, da bo med razvojem v določenih točkah prihajalo do sprememb in odstopanj. Nekatere zahteve so bile podane dokaj ohlapno, za določene segmente še niso bile dodelane končne različice oblikovnih predlog. Ker je bil naročnik projekta (uredništvo) pripravljen na tesno sodelovanje z razvijalsko ekipo med celotno fazo implementacije, smo se odločili, da se projekt vodi po metodi Scrum.

Scrum [7] je ena izmed agilnih metod razvoja programske opreme. Agilne metodologije so se formalno pojavile leta 2001 z nastankom *Manifesta za agilni razvoj programske opreme* [12]. Agilne metodologije se razlikujejo od tradicionalnih metod, ki dajejo veliko poudarka pisanju obsežne dokumentacije in izdelavi načrtov. Praksa je pokazala, da se razvoj programske opreme močno razlikuje od projektov v npr. gradbeništvu, arhitekturi, strojništvu ipd. Okoliščine, v katerih se izvaja razvoj programske opreme, so veliko bolj spremenljive, dinamične. Na te spremembe se je treba hitro odzvati, česar pa nam tradicionalne metode razvoja ne omogočajo. Agilne metodologije namesto natančnega vnaprejšnjega planiranja vzpostavljajo empirične metode za sprotni vpogled in prilagajanje trenutnemu stanju med razvojem projekta. To lahko dosežemo z iterativnim in inkrementalnim pristopom.

4.2.1 Na kratko o metodi Scrum

Za razumevanje metode Scrum moramo najprej našteti in opisati nekaj specifičnih terminov, za katere še ne obstajajo splošno sprejeti prevodi v slovenščino. V tem razdelku si bomo sposodili prevode, ki so predlagani v [4].

Sodelujoči v razvojni ekipi opravljajo eno izmed treh vlog: produktni vodja (angl. *Product owner*), razvojna skupina (angl. *Scrum team*) in skrbnik metodologije (angl. *Scrum master*).

Produktni vodja kot predstavnik naročnika zastopa vse, ki so zainteresirani za rezultate projekta. Vzdrževati mora seznam zahtev (angl. *Product nacklog*), določati njihove prioritete in jih združevati v posamezne izdaje (angl. *Release*). Seznam zahtev se lahko med projektom dopolnjuje. Vsaka zahteva je praviloma opisana v obliki uporabniške zgodbe (angl. *User story*), ki zajema opis zahteve in seznam sprejemnih testov. Opis zahtev je lahko precej ohlapen, zato mora biti produktni vodja razvijalcem vedno na voljo za pojasnila glede podrobnosti, povezanih z realizacijo. Sprejemni testi (angl. *Accepptance test*) služijo razvijalcem kot dodatne smernice in produktnemu vodji kot vodilo za sprejem oziroma zavrnitev uporabniške zgodbe.

Razvojna skupina je zadolžena za implementacijo zahtevane funkcionalnosti.

Skrbnik metodologije je zadolžen za nemoten potek projekta. Skrbi za upoštevanje pravil metode Scrum in na ustrezen način izvaja predpisane aktivnosti. Odpravljati mora morebitne ovire in zagotavljati optimalne pogoje za delo razvoje skupine.

Razvoj po metodi Scrum poteka iterativno. Vsaka iteracija (angl. *Sprint*) je dolga določeno število dni. Začne se s sestankom za načrtovanje iteracije (angl. *Sprint planning meeting*), na katerem se produktni vodja in razvojna skupina dogovorijo, katere zgodbe bodo v iteraciji izvedene. Rezultat sestanka je seznam nalog (angl. *Sprint backlog*), ki vsebuje vse naloge, potrebne za realizacijo dogovorjene funkcionalnosti do konca iteracije. Seznam se med iteracijo dopolnjuje, obsežnejše naloge pa se razdelijo na manjše enote.

Na samem začetku produktni vodja predstavi seznam vseh do tedaj znanih zahtev v obliki uporabniških zgodb, jih razvrsti po prioriteti in razdeli v predvidene izdaje. Razvojna skupina oceni zahtevnost posamezne uporabniške zgodbe z ustreznim številom točk (angl. *Story points*) in določi predvideno hitrost razvoja (angl. *Velocity*), tako da oceni, koliko točk lahko realizira v eni iteraciji. V skladu s tem nato produktni vodja izdela plan izdaje (angl. *Release plan*), tako da v skladu s prioriteto razporedi zgodbe po posameznih iteracijah. Pri tem seštevek točk vseh zgodb v neki iteraciji ne sme preseči predvidene hitrosti razvoja.

Med iteracijo se člani razvojne skupine dnevno zberejo na 15-minutnem sestanku (angl. *Daily scrum meeting*), na katerem vsak izmed njih opiše napredek pri svojem delu in morebitne težave. Namen tega sestanka je sinhronizirati delo vseh članov skupine in sproti identificirati morebitne probleme.

Razvoja skupina na koncu vsake iteracije produktnemu vodji predstavi napredek na sestanku za pregled rezultatov (angl. *Sprint review meeting*). Metoda Scrum striktno zahteva, da razvojna skupina upošteva koncept "dokončano" (angl. *Done*), kar pomeni, da mora biti vsaka zgodba v celoti

realizirana in dokumentirana, tako da jo je mogoče neposredno predati v produkcijo. Produktni vodja "sprejme" le tiste zgodbe, ki v celoti zadostijo zahtevam tega koncepta. Sprejete zgodbe se upoštevajo pri izračunu dejanske hitrosti razvoja (angl. *Actual velocity*).

Skrbnik metodologije po tem sestanku (in pred začetkom naslednje iteracije) organizira sestanek za oceno kakovosti razvojnega procesa (angl. *Sprint retrospective meeting*), na katerem razvojna skupina izpostavi opažene pomanjkljivosti v razvojnem procesu in poskuša najti izboljšave, ki bi povečale učinkovitost v naslednjih iteracijah.

4.2.2 Implementacija metode Scrum na projektu "slovenskenovice.si"

V času priprav na prenovo portala slovenskih novic in po odločitvi za delo po metodi Scrum je bil opravljen pregled izkušenj razvojnih skupin na podobnih projektih. V skupnosti uporabnikov platforme Drupal je kar nekaj podjetij, ki uporabljajo metodo Scrum. Pri uvedbi metode smo poskušali upoštevati njihove izkušnje in priporočila iz strokovne literature [2]. Posebno pozornost smo namenili razdelitvi vlog, določitvi koncepta "dokončano", določitvi dolžine iteracije, izbiri ustreznega orodja za spremljanje poteka dela, ocenjevanju zahtevnosti uporabniških zgodb ter izobraževanju vseh sodelujočih.

Za uspeh projekta je ključnega pomena vloga produktnega vodje. Ta zastopa interese zainteresiranih uporabnikov in razvojni ekipi posreduje vizijo o tem, kaj je treba narediti. Določa tudi kriterije za ocenjevanje rezultatov. Med projektom mora tesno sodelovati z razvojno ekipo in odgovarjati na vprašanja glede podrobnosti realizacije uporabniških zgodb. V našem primeru je vlogo produktnega vodje prevzel pomočnik odgovorne urednice. V preteklosti je že sodeloval pri projektih razvoja večjih spletnih mest. Bil je ustrezno tehnično izobražen in kompetenten za sogovornika z razvojno skupino.

Nalogo skrbnika metodologije je prevzel vodja oddelka spletnega razvoja.

Razvojna skupina je bila sestavljena iz 5 članov. Eden izmed njih se je s

platformo Drupal srečal prvič in je bil zato v začetnih iteracijah pri določitvi hitrosti razvoja upoštevan le polovično.

Pri določitvi koncepta "dokončano" smo postavili naslednje pogoje:

- rešitev mora zadostiti vsem sprejemnim testom,
- koda mora biti napisana v skladu s standardi kodiranja,
- koda mora biti ustrezno komentirana,
- rešitev mora biti ustrezno dokumentirana,
- celotna funkcionalnost in koda morata biti pregledani s strani drugega razvijalca,
- rešitev mora biti pregledana s strani skrbnika metodologije,
- rešitev mora biti potrjena s strani produktnega vodje.

Preverjanje ustreznosti koncepta "dokončano" smo v veliki meri opravili ročno. Pri preverjanju skladnosti s standardi kodiranja smo si pomagali z orodjema *Drupal code sniffer*[14] in *Coder*[15].

Dokumentacijo smo zapisovali v sistemu "wiki", za samo spremljanje projekta pa smo uporabljali sistem *Agilo for Scrum*[16], ki temelji na spletnem vmesniku.

Pri določitvi dolžine iteracije (angl. Sprint length) smo pristali na kompromis med obsegom režije, ki jo zahtevajo daljše iteracije, in prilagodljivostjo, ki jo lahko dosežemo, če so iteracije kratke. Metoda Scrum priporoča 30-dnevne iteracije, v praksi pa so pogoste iteracije, ki trajajo 2 tedna. Naša ekipa se je odločila za dolžino 3 tednov.

Ocenjevanje zahtevnosti uporabniških zgodb, ki smo ga opravljali na sestankih za načrtovanje iteracije, smo opravljali po metodi "Planning poker" [2]. Kot možno oceno zahtevnosti posamezne uporabniške zgodbe smo upoštevali le vnaprej predpisane dopustne vrednosti 0.5, 1, 2, 3, 5, 8 in 13 točk, ob dogovoru, da ena točka pomeni 1 delovni dan ali 6 efektivnih delovnih ur.

[14] http://drupal.org/project/drupalcs
[15] http://drupal.org/project/coder
[16] http://www.agiloforscrum.com/

Začetno hitrost razvoja smo določili kot produkt števila razvijalcev in števila delovnih dni v iteraciji. V začetnih iteracijah smo enega od razvijalcev zaradi nepoznavanja platforme Drupal upoštevali le s polovičnim številom točk. Tako smo pri prvi iteraciji prišli do hitrosti 32,5 točke. To oceno smo med projektom prilagajali dejansko doseženi hitrosti v predhodnih iteracijah.

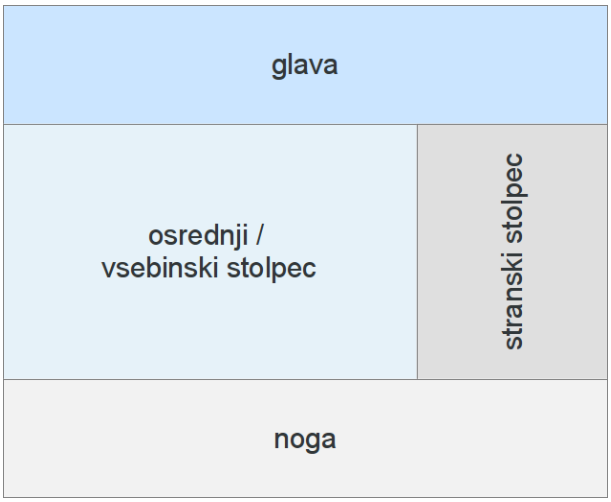
Posebnost razvojne ekipe slovenskih novic je bila v tem, da razvijalci nismo delali na skupni lokaciji. Le eden je delal na sedežu podjetja Delo v Ljubljani, ostali pa iz različnih lokacij po Sloveniji. Ta okoliščina je zahtevala, da smo dnevne sestanke izvajali preko spletnih konferenčnih sistemov, in je nekoliko oteževala komunikacijo med člani razvojne skupine. Kljub vsemu pa do večjih težav vseeno ni prihajalo. Na sestankih za načrtovanje iteracije in sestankih za predstavitev rezultatov se je ekipa v celotni sestavi zbrala na sedežu podjetja v Ljubljani.

Celoten razvoj je trajal 7 iteracij. V tem času nam je uspelo izvesti približno 80 % zastavljene funkcionalnosti. Ker smo s projektom zamujali, se je naročnik vseeno odločil za izdajo, saj smo manjkajoče funkcionalnosti zaradi modularne zasnove lahko dodali kasneje. Podrobnosti o izvedbi Scruma na projektu slovenskih novic si bralec lahko ogleda v [10].

4.3 Struktura portala

Spletni portal “slovenskenovice.si” lahko v grobem razdelimo na naslednje vsebinsko-oblikovne sklope:

- naslovnica,
- kategorijska/podkategorijska stran,
- članek,
- vremenska stran,
- horoskop,
- čestitke,
- nagradne igre,
- arhiv člankov.



Slika 4.1: Osnovna struktura strani spletnega portala "slovenskenovice.si".

Vsak od sklopov postavlja različne tehnične zahteve in prilagojene rešitve.

Vizualno lahko strukturo spletnega portala opišemo z naslednjimi vsebinskimi polji (slika 4.1):

- glava,
- osrednji/vsebinski stolpec,
- stranski stolpec,
- noga.

Posamezne podstrani predstavljajo variacije te osnovne strukture. Predvsem se razlike pojavijo na naslovnici, ki je oblikovana tako, da na kar se da pregleden in čimbolj atraktiven ter dinamičen način predstavi najnovejše in izpostavljene članke.

Vsaka stran ima tudi določeno število oglasnih pozicij. Večino teh pozicij zapolnjujejo zakupljeni oglasi, nekatere pa lahko za namene promocije svojih akcij vnese tudi uredništvo.

4.4 Opis vsebinskih sklopov

4.4.1 Naslovnica

Naslovnica (glej sliko 4.2) je vsebinsko najbogatejša stran portala. Razdelimo jo lahko na dva sklopa: *kapitalka* in *kategorijski bloki*. V *kapitalki* so izpostavljene najpomembnejše novice dneva. Uredniki lahko v tem sklopu prikažejo različno število člankov. Glede na to, koliko so posamezne novice odmevne, se lahko odločijo za mrežo šestih, petih, štirih ali ene same novice. Če je prikazana samo ena novica, se ta razširi čez celotno področje in skrije blok *dosje*, v katerem so izpostavljene tako imenovane *vroče teme* — skupine člankov na določeno temo.

Nad *kapitalko* je *tekoči seznam novic* z možnostjo preklopa na *udarno novico*. V bloku *Uredništvo priporoča* je tekoči seznam uredniško izpostavljenih novic. Pod njim se začne osrednji vsebinski sklop s *kategorijskimi bloki*. To so sklopi uredniško izpostavljenih novic iz posameznih kategorij. Čisto na dnu najdemo še sklop novic iz *partnerskih portalov*. To so novice, ki se iz ostalih Delovih portalov nalagajo preko virov RSS.

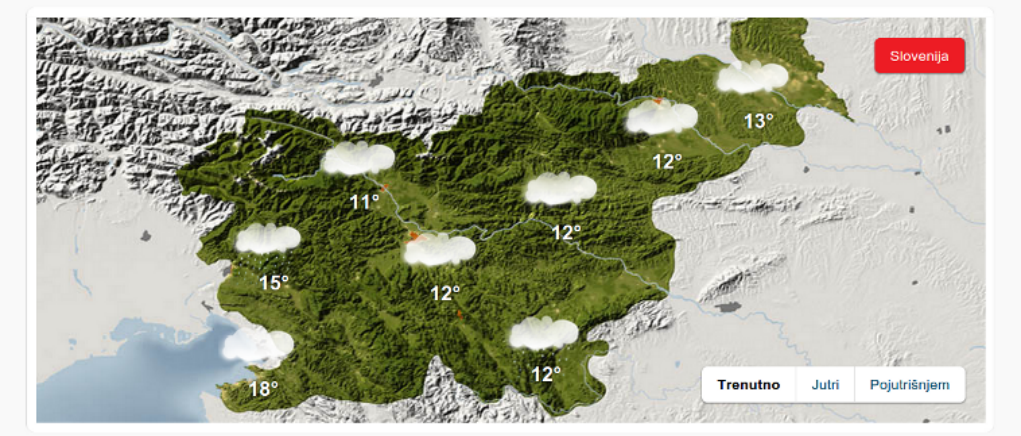
V desnem stolpcu se nahajajo blok z znamenji iz *horoskopa*, seznami zadnjih plačljivih člankov, anketa ter blok *zadnjih čestitk*.

4.4.2 Kategorijska/podkategorijska stran

Kategorijska in podkategorijska stran (slika 4.3) sledita vzorcu naslovnice (slika 4.3). V zgornjem delu se nahaja *kapitalka* s seznamom v tej kategoriji oziroma podkategoriji izpostavljenih novic. V osrednjem vsebinskem delu pa se nahajajo v enem primeru *podkategorijski bloki* oziroma seznam novic iz te podkategorije. Sledita še bloka *najbolj branih* novic iz te kategorije in *najbolj komentiranih* novic iz te kategorije. Pod njima je *arhivski blok* s seznamom starejših novic in čisto spodaj še blok *osrednjih novic*, ki vsebuje po dve zadnji novici iz vsake kategorije.

Slika 4.2: Naslovnica spletnega portala "slovenskenovice.si". Označeni so najpomembnejši elementi: 1 — vremenski blok, 2 — glavni meni z iskalnikom, 3 — tekoči seznam novic, 4 — kapitalka, 5 — blok "Dosje", 6 — blok "Uredništvo priporoča", 7 — kategorijski bloki, 8 — stranski stolpec, 9 — vsebine partnerskih portalov, 10 — noga z statičnim menijem.

Slika 4.3: Zaslonska slika celotne strani kategorije (levo) in podkategorije (desno).



Slika 4.4: Izsek iz vremenske strani. Zemljevid s prikazom trenutnega vremena po regijah.

4.4.3 Članek

Članki so kategorizirani v dvonivojski taksonomiji in so vedno uvrščeni na zadnji nivo. Poleg standardnega polja za vsebino članka ta vsebuje še vrsto metapodatkov: nadnaslov, ključne besede, vodilno besedilo, povezane vsebine. Članek lahko vsebuje tudi multimedijske vsebine, kot so fotografije ali video vsebine. Povezane vsebine uredniki vnašajo ročno prek posebnega iskalnika za vnos povezav na ostale članke v CMS.

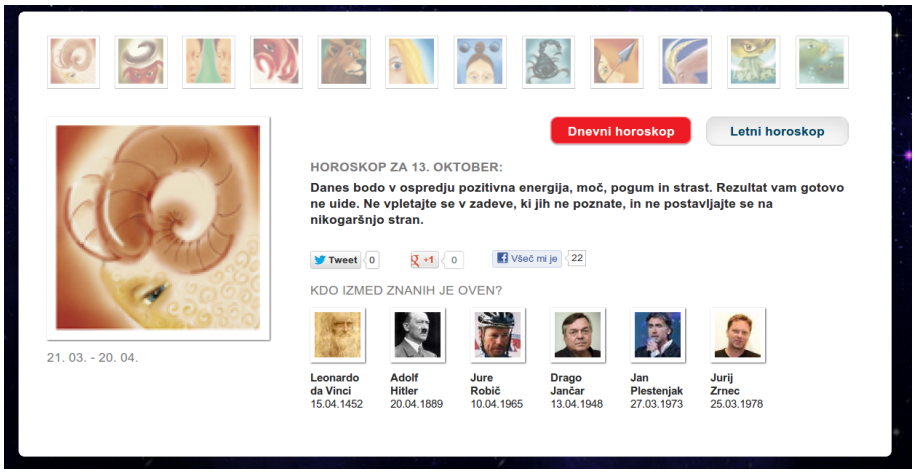
4.4.4 Vremenska stran

Na vremenski strani (slika 4.4) lahko bralci spremljajo trenutno vreme in vremensko napoved za Slovenijo. Vremenske podatke zagotavlja Državna meteorološka služba RS in se uvažajo iz njihovih javnih virov XML[17].

4.4.5 Horoskop

Horoskop se kot poseben tip vsebine nahaja v sklopu kategorije "Astro". Posamezni astrološki znaki so termini v posebni taksonomiji *Astrološki znaki*.

[17]http://meteo.arso.gov.si/met/sl/service/



Slika 4.5: Glavni blok na strani kategorije "Astro" s prikazom dnevnega horoskopa za znak "Oven" ter znanimi osebami tega astrološkega znaka.

Podatki za dnevni horoskop (slika 4.5) se dnevno berejo iz zunanjega vira v obliki XML in se v CMS shranjujejo v tip vsebine `sn_horoscope`. Za vsak znak poleg dnevnega prikazujemo tudi *letni horoskop* in seznam znanih osebnosti tega astrološkega znaka. Oba podatka sta posebna tipa vsebine, ki ju kategoriziramo v taksonomiji *Astrološki znaki*.

4.4.6 Čestitke

Del spletnega portala, na katerem so objavljene izključno vsebine, ki jih prispevajo bralci, so čestitke. Razvrščene so v tri kategorije v posebno taksonomijo. Čestitke lahko oddajo le registrirani uporabniki. Pri objavi lahko poleg besedila za voščilo vnesejo tudi več fotografij in izbirajo med več vrstami oblikovnih predlog. Prikaz je razdeljen na dva dela: pregled čestitk po kategorijah in prikaz posamezne čestitke. Na sliki 4.6 je prikazan eden izmed blokov na pregledu po kategorijah.

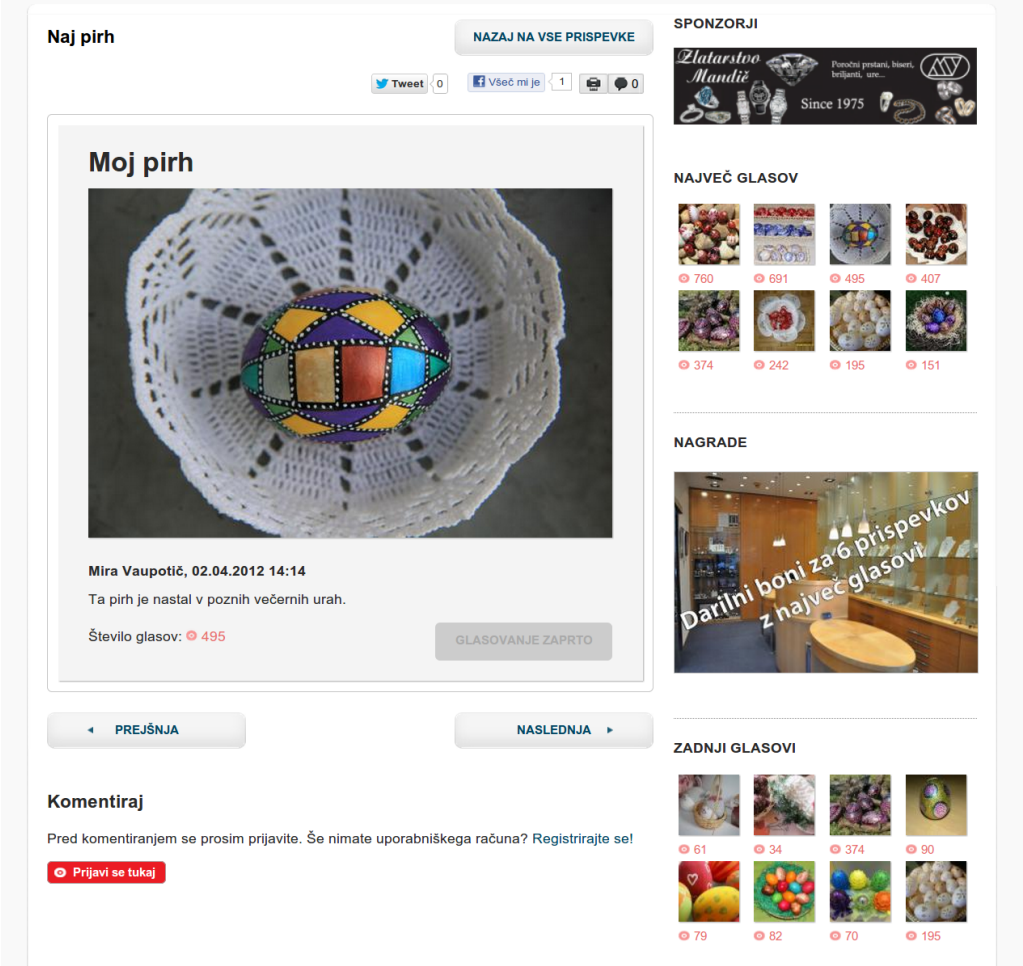
Slika 4.6: Blok z zadnjimi objavljenimi čestitkami iz kategorije "Otroci". Nad sličicami je ponazorjena izbrana oblikovna predloga.

4.4.7 Nagradne igre

Uredništvo slovenskih novic lahko na portalu izvede tudi nagradne igre. V njih uporabniki objavijo prispevke na določeno temo (besedilo in/ali fotografija), potem pa glasujejo za tiste, ki so jim všeč. Prispevek z največ glasovi uporabnikov je zmagovalec nagradne igre.

Sklop nagradne igre je sestavljen iz več komponent:

- tip vsebine "Nagradna igra",
- tip vsebine "Prispevek",
- modul za beleženje oddanih glasov,
- pogledi za prikaz prispevkov,
- večstopenjski formular za oddajo prispevka.



Slika 4.7: Prispevek v nagradni igri. V osrednjem delu je prispevek z gumbom za oddajo glasu in navigacijo med prispevki povezane nagradne igre. V stranskem stolpcu sta poleg sponzorskih blokov še bloka s prispevki, ki so prejeli največ glasov, in prispevki, ki so nazadnje prejeli glas.

4.4.8 Arhiv člankov

Ker je novičarski portal vsebinsko zelo dinamičen z visoko frekvenco objav, se članki z naslovnice zelo hitro umaknejo globlje v hierarhijo portala na, kategorijsko stran, kasneje pa na podkategorijsko stran. Lahko se zgodi, da že po nekaj dneh članki z navadnim "brskanjem" niso več dosegljivi. Arhiv

Slika 4.8: Arhiv člankov. Izsek strani s formularjem za iskanje in rezultati iskanja.

(glej sliko 4.8) je orodje, ki uporabniku omogoča iskanje člankov, ki ustrezajo določenim kriterijem. To je bodisi niz znakov, ki se nahaja v besedilu članka, ali datum objave, ali kategorija, v kateri se nahaja itd...

4.5 Ključni izzivi pri implementaciji novega portala

4.5.1 Upravljanje s konfiguracijo

Konfiguriranje modulov in graditev funkcionalnosti je v Drupalu razmeroma enostavno opravilo. Lahko bi rekli, da je to tisto področje, na katerem Drupal resnično blesti. Po mojih izkušnjah lahko večino problemov rešimo praktično brez kodiranja, s primernim izborom prispevanih modulov in nekaj kliki. Najzmogljivejši prispevani moduli (med njimi na primer tudi *Views* in *Panels*) omogočajo uporabo njihove funkcionalnosti prek uporabniškega vmesnika v brskalniku. Tak način dela je sicer res zelo hiter, težava pa je, da konfiguracija tako sestavljenega portala živi v podatkovni bazi. V podatkovni bazi pa je shranjena tudi vsebina. Postavitev funkcionalnosti v produkcijsko okolje, potem ko portal že enkrat zaživi, je v tem primeru zelo nerodna. Kopiranje podatkovne baze ne pride v poštev, saj lahko pride do izgube podatkov. Kopiranje posameznih konfiguracijskih tabel tudi, saj lahko pride do kršitve referenčne integritete. Edina rešitev je, da vso konfiguracijo še enkrat "naklikamo" na produkcijskem strežniku. Pri tako velikih projektih, kot so "slovenskenovice.si", je to seveda nesprejemljivo. Portal bi bilo treba ob vsaki nadgradnji postaviti v vzdrževalni način, v katerem bralci in uredniki portala ne morejo uporabljati. Poleg tega je konfiguracijo v taki obliki nemogoče voditi v sistemu za nadzor različic izvorne kode.

Z opisanimi problemi so se srečali vsi resnejši uporabniki Drupala, med njimi tudi podjetje *Developmentseed*, ki je marca leta 2009 predstavilo svojo rešitev. Ustvarili so modul *Features*, ki vpeljuje pojem "izvožene konfiguracije" (angl. exportable). Moduli lahko z uporabo programskega vmesnika *Ctools Exportables* definirajo, kako se konfiguracija, ki sicer živi v podatkovni bazi, izvozi v zapis, primeren za hranjenje v datotekah, in kako se iz takega zapisa konfiguracija uvozi nazaj v podatkovno bazo. Modu,l *Features* take zapise izvozi v datoteke v obliki PHP-kode, ki je obenem modul za Drupal.

Na tak način lahko izvozimo funkcionalnost, ki predstavlja neki uporabniški primer in ga zapakiramo v obliki modula. Modul *Features* ima tudi mehanizme za povrnitev izvožene konfiguracije.

Pri implementaciji portala "slovenskenovice.si" smo od samega začetka upoštevali zgoraj opisane vidike. Osnovno strukturo portala smo zasnovali v posebnem namestitvenem profilu, ki nam je omogočal, da si je lahko vsak razvijalec v nekaj minutah postavil trenutno razvojno različico portala na svojem računalniku. Konfiguracijo smo izvažali s pomočjo modula *Features*, vse ostale spremembe na podatkovni bazi smo implementirali z mehanizmom posodobitvenih kljuk (angl. *update hooks*) in spremembe vodili s pomočjo odprtokodnega sistema za upravljanje izvorne kode Git[18].

Razvoj funkcionalnosti je tipično potekal po naslednji korakih:

- pridobi zadnjo veljavno različico izvorne kode iz centralnega repozitorija,
- razvij novo funkcionalnost,
- izvozi konfiguracijo in sprogramiraj posodobitvene procedure,
- potrdi spremembe v sistemu za upravljanje izvorne kode in jih pošlji v centralni repozitorij.

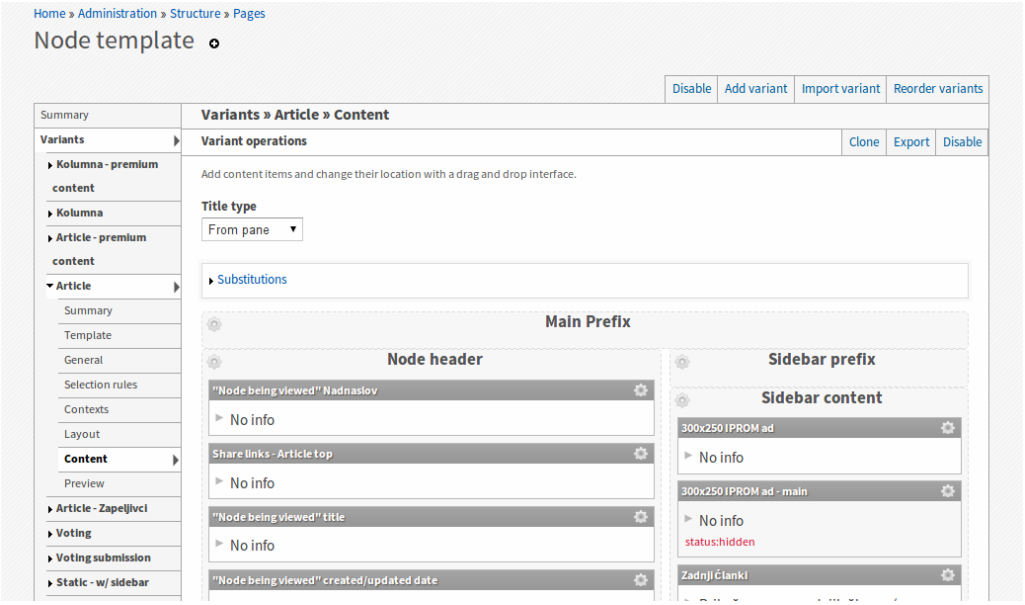
Analogno pa postavitev na produkciji po naslednji korakih:

- pridobi zadnjo veljavno različico izvorne kode iz centralnega repozitorija,
- povrni konfiguracijo iz kode,
- sproži posodobitvene procedure.

4.5.2 Struktura strani

Struktura strani, kot jo je predvidevala oblikovna predloga novega portala, je bila zastavljena precej kompleksno. Pri enostavnejših spletnih mestih ponavadi postavitev vsebine določamo kar na nivoju predloge, vendar se v tem

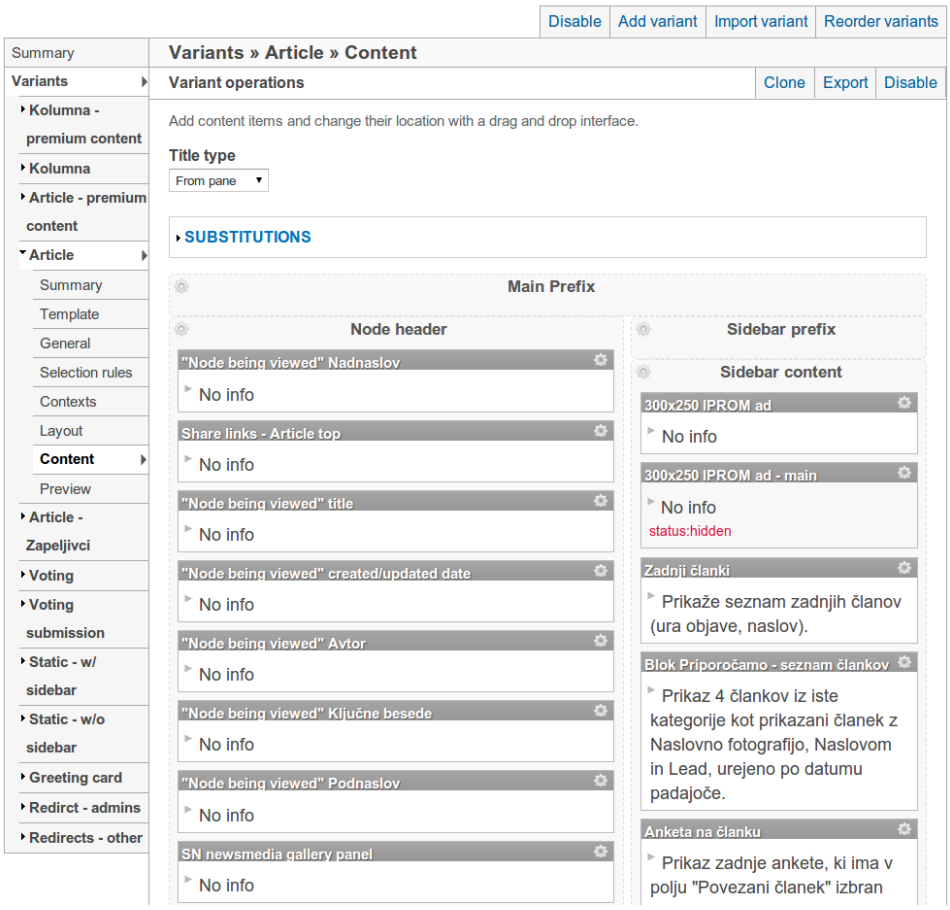
[18] http://git-scm.com/



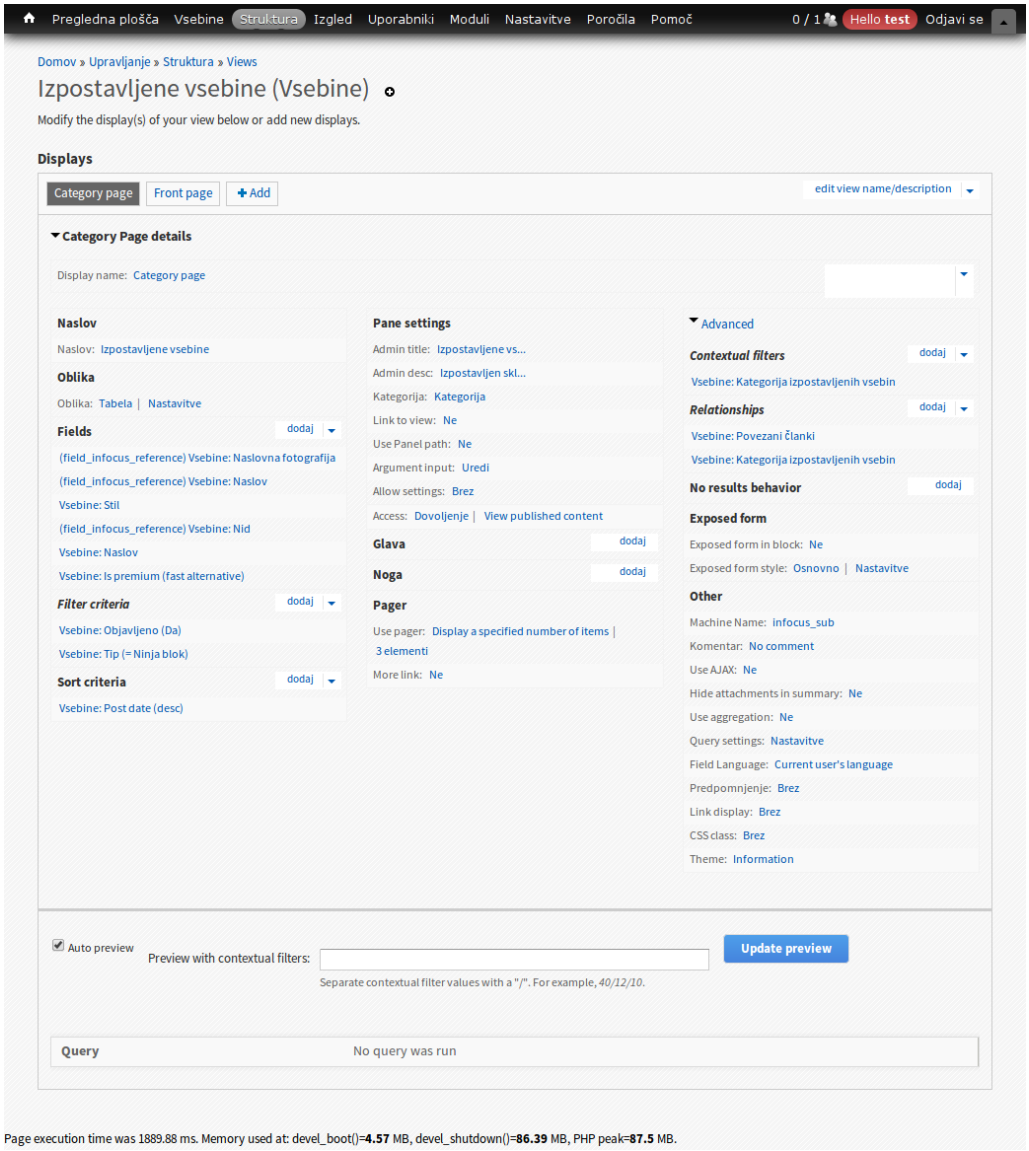
Slika 4.9: Administracijski vmesnik modula *Panels*.

primeru odpovemo dinamičnosti, saj je večina vsebinskih blokov fiksirana na točno določenih pozicijah. V primeru portala slovenskih novic je ta mehanizem absolutno preveč tog in pri implementaciji ni prišel v poštev. Prispevani modul *Panels*[19] odpravlja ravno to pomanjkljivost. Omogoča definiranje in uporabo različnih postavitev strani neposredno prek uredniškega vmesnika (slika 4.9). Integracija z ostalimi prispevanimi moduli (predvsem *Views*) nam omogoča ustvariti pametne variacije postavitev, ki se različno obnašajo glede na kontekst, v katerem se trenutno nahaja uporabnik.

Prikaz članka je izveden s paneli (slika 4.10). Ker se članek lahko na strani prikazuje kot "kolumna" ali "članek", smo za vsako obliko naredili svojo varianto panela (angl. *Panel variant*). Katera varianta panela se izbere za prikaz, definira vrednost posebnega polja na članku, ki ga določi avtor.



Slika 4.10: Konfiguracija panela za prikaz člankov.



Slika 4.11: Administracijski vmesnik modula Views.

4.5.3 Seznami člankov in ostalih vsebin

Modul *Views*[20] nam omogoča, da različne prikaze vsebine, kategorij, uporabnikov in ostalih entitet, ki jih pozna Drupal, ustvarimo hitro in enostavno. V osnovi je pameten graditelj poizvedb, ki na podlagi podanih zahtev ustvari poizvedbo v podatkovno bazo, jo izvede in na ustrezen način izpiše njene rezultate. Poglede lahko definiramo kar prek uporabniškega vmesnika (glej sliko 4.11). Skupaj z modulom *Panels* nam omogoča ustvariti zanimive razporede kontekstno odvisne vsebine v zelo kratkem času. *Views* je zelo kompleksen modul. Podrobnejši opis najdete v [6, 5].

4.5.4 Uvažanje vremenskih podatkov

Modul *Feeds*[21] omogoča uvažanje podatkov iz različnih virov (na primer v obliki XML-datotek ali datotek, ki vsebujejo z vejico ločene vrednosti (CSV), RSS/Atom viri ...). Rezultat uvoza so lahko entitete (vozlišča, uporabniki, termini ...) ali posamezni zapisi v podatkovni bazi. Fleksibilnost je dosežena z razdelitvijo posameznih korakov uvoza na komponente, ki jih lahko programsko razširjamo oziroma napišemo svoje. Uvoz se začne s korakom prevzemanja (angl. fetch). V tem koraku se izvede dostop in branje vira podatkov. Ti se predajo razčlenjevalniku, ki poskrbi, da se podatki pravilno izluščijo iz podatkovnega vira (npr. podatke iz CSV-datoteke izluščimo glede na vejico, ki predstavlja ločilo med posameznimi atributi). V zadnjem koraku nato procesor podatke interpretira in shrani vsebine v CMS.

Modul *Feeds* že vsebuje generične implementacije za vsakega od opisanih korakov. Za dostop do zunanjih virov na spletu se tako pogosto uporablja *HTTP Fetcher*, ki omogoča, da podatke pridobimo na nekem URL-naslovu, *Common syndication parser* razčleni vhod v XML-obliki, *Node processor* pa podatke shrani v CMS kot vozlišča določenega tipa. Tako lahko na zelo enostaven način sestavimo preprost agregator novic.

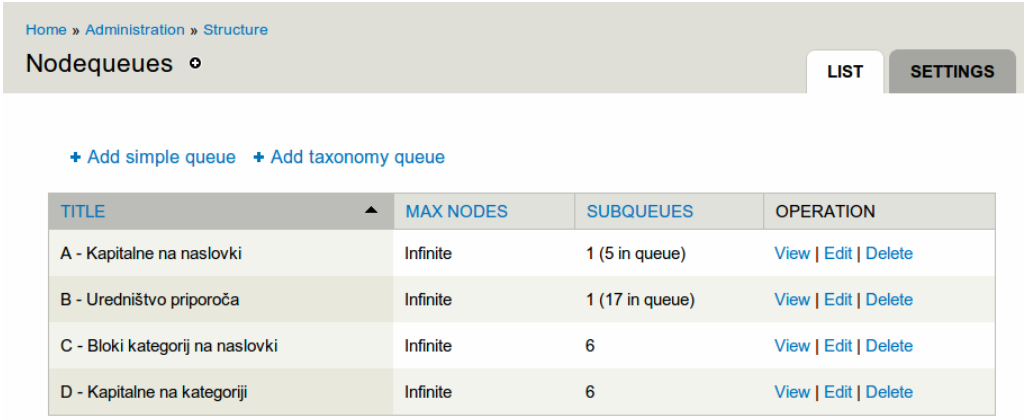
[19] http://drupal.org/project/panels
[20] http://drupal.org/project/views
[21] http://drupal.org/project/feeds

Modul *Feeds* smo izkoristili pri izvedbi vremenskih strani. Branje podatkov iz XML-vira, ki je javno dostopen na strani ARSO, je implementirano v razredu `FeedsHTTPFetcher`, ki je vključen v modulu *Feeds*. Za razčlenitev podatkov, je bilo treba razširiti razred `FeedsParser` in implementirati metodo `FeedsParser::parse()`. Tako pripravljene podatke nato v CMS-ju shranimo kot vozlišča določenega tipa. Preslikavo v razčlenjevalniku pripravljenih podatkov v polja določenega tipa vsebine v CMS-ju izvajamo v metodi `process()` razreda, ki razširja razred `FeedsNodeProcessor`.

4.5.5 Izpostavljanje vsebin

Ena od pomembnejših zahtev uredništva je bila tudi popoln nadzor nad izpostavljanjem člankov, to je uredniškim uvrščanjem pomembnejših vsebin v specifične pozicije na portalu. Večina seznamov člankov na naslovnici je uredniško sestavljenih. Na naslovnici so to: novice v kapitalki, "uredništvo priporoča" in kategorijski bloki. Podobna je situacija v kategorijski strani. Treba je bilo sestaviti mehanizem, ki urednikom omogoča fleksibilen in pregleden nadzor nad tem, kje se določen članek na strani prikaže. Pri rešitvi tega problema smo si pomagali z modulom *Nodequeue*. Ta vpeljuje pojem *vrste*. Vrsta, ki jo definira modul *Nodequeue*, je v osnovi le seznam, v katerega lahko uvrščamo vsebine in jih razporejamo. Vrsti lahko določimo tudi maksimalno število elementov, ki se v njej nahajajo. Če je vrsta polna, novo uvrščena vsebina izrine vsebino na zadnjem mestu v vrsti. Definiramo lahko več vrst. Pri implementaciji portala "slovenskenovice.si" smo za vsako pozicijo na strani definirali svojo vrsto (slika 4.12).

Vsaka vrsta ima lahko tudi podvrste. Tako dobimo dvonivojsko hierarhijo, ki smo jo pri implementaciji izkoristili za pozicije kategorijskih blokov. V kombinaciji z modulom *Smartqueue taxonomy* lahko namreč definiramo "pametne" vrste, ki so vezane na določeno kategorijo in samodejno ustvarijo podvrste za vsako od podkategorij (slika 4.13). Prednost take organizacije je v bistveno boljši preglednosti uredniškega vmesnika (slika 4.14), saj so pozicije za izpostavljanje lahko združene, članki iz različnih kategorij pa se med



Slika 4.12: Vrste za izpostavljanje vsebin na portalu "slovenskenovice.si".

seboj ne mešajo v enem seznamu.

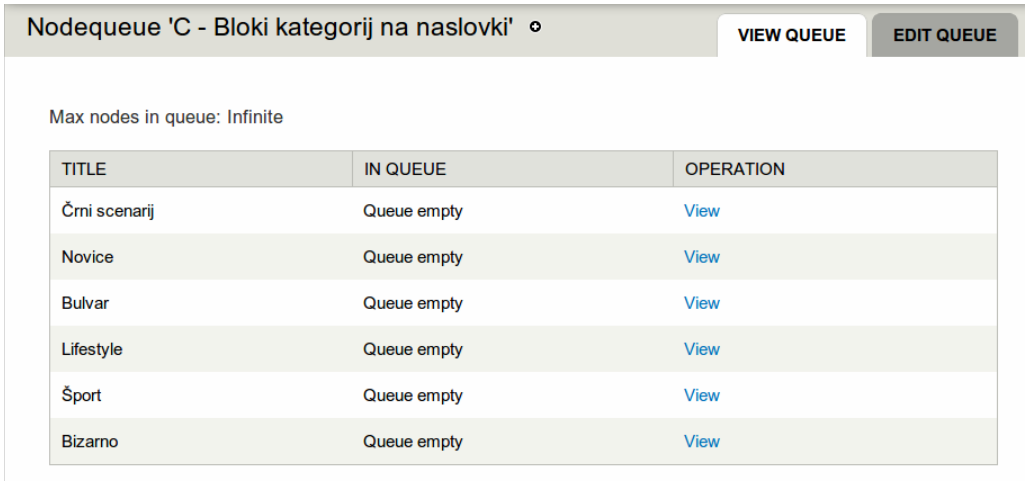
Ključna lastnost modula *Nodequeue* pa je odlična integracija z modulom *Views*. Vsi kategorijski in podkategorijski bloki na naslovnici in kategorijski strani so implementirani s pomočjo kontekstnih filtrov. To je mehanizem modula *Views*, ki omogoča, da iz trenutnega konteksta (zahteve za prikaz strani) določi filter po izbranem kriteriju.

4.5.6 Selitev vsebin s starega portala

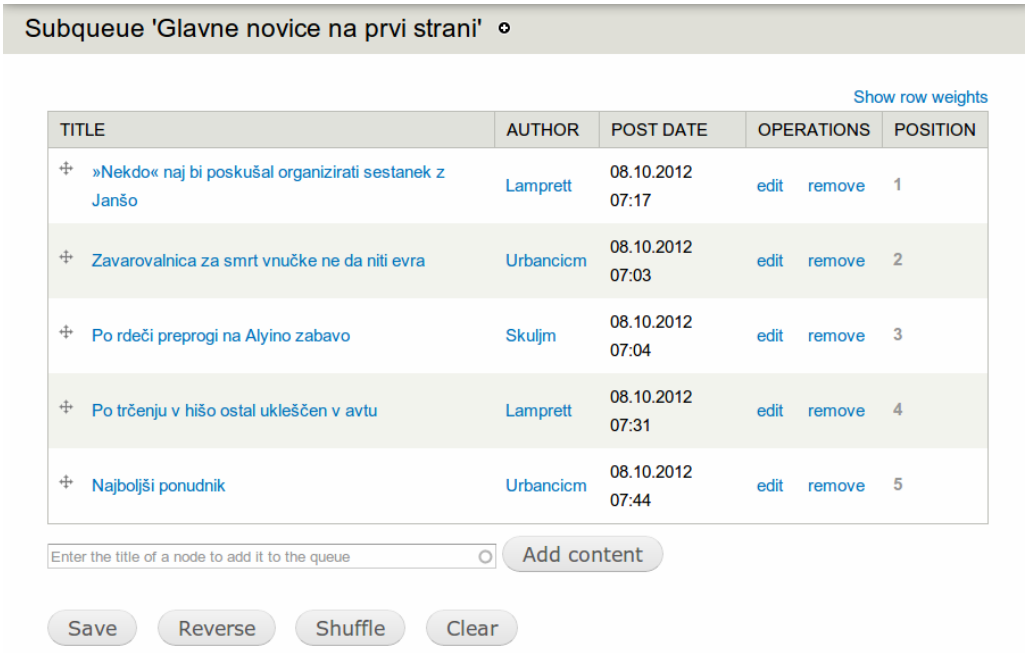
Pri prenovi portala "slovenskenovice.si" je bilo treba poskrbeti tudi za selitev vsebin iz starega sistema. Za ta namen smo izkoristili modul *Migrate*[22], ki podobno kot *Feeds* omogoča uvažanje podatkov v Drupal. Zastavljen je kot ogrodje, na podlagi katerega lahko programiramo procedure za selitve podatkov. Ko implementiramo svojo selitev podatkov, razširjamo razred `Migration`. Pri tem je pomembno, da v konstruktorju pravilno definiramo naslednje tri parametre:

- `source`: izvor podatkov — ponavadi tipa `MigrateSourceSQL`, kar pomeni, da podatke pridobimo z neko SQL-poizvedbo;

[22] http://drupal.org/project/migrate



Slika 4.13: Pametne vrste za izpostavljanje vsebin v kategorijske bloke.



Slika 4.14: Administracijski vmesnik za urejanje vrste izpostavljenih vsebin.

- `destination`: ponor podatkov — v primeru uvoza vsebin je ta ponavadi tipa `MigrateDestinationNode`;
- `map`: definiranje primarnega ključa, po katerem lahko bijektivno preslikamo podatke med ponorom in izvorom.

Poleg tega je treba definirati ustrezne preslikave polj med izvorom in ponorom.

Če hočemo izvajati inkrementalne selitve, moramo pravilno definirati še parameter `highwaterField`, polje, ki nam pove, kateri podatki v izvoru so novi. *Migrate* za vsak preseljeni zapis poleg primarnih ključev v izvoru in ponoru zabeleži tudi vrednost tega polja. Če selitev poganjamo večkrat zapored, se preselijo le tisti zapisi, katerih nova vrednost parametra `highwaterField` presega staro.

Selitve so med seboj lahko tudi soodvisne (npr. komentarjev na članke ne moremo seliti, preden ne preselimo samih člankov).

Implementirali smo tri procedure: selitev člankov, selitev komentarjev in selitev multimedijskih vsebin. Procedure smo implementirali že zelo zgodaj v procesu razvoja, saj smo ves čas vzdrževali testno namestitev prenovljenega portala z vsebinami iz produkcijskega strežnika. Na ta način smo lahko zelo hitro odkrili morebitne napake. Pri prenosu člankov je bilo treba ohraniti tudi stare URL-je člankov. Pri tem smo si pomagali z modulom *Redirect*[23].

4.5.7 Implementacija nagradnih iger

Nagradna igra je običajen tip vsebine, ki med drugimi vsebuje tudi polje *nodereference*[24] za *prispevke*. *Nagradne igre* lahko objavljajo le uredniki. Podobno ima tudi *prispevek* polje *nodereference*, ki jo povezuje z nagradno igro, v katero spada. Obojestranska referenca se vzdržuje pri shranjevanju enega in drugega tipa vsebine in je implementirana v `hook_node_presave()` modula *Voting*.

[23] http://drupal.org/project/redirect
[24] http://drupal.org/project/references

Za beleženje glasov na posamezni prijavnici skrbita prispevana modula *Votingapi*[25] in *Plus1*[26]. Uporabniki lahko za vsako prijavnico v času, ko je glasovanje odprto, glasujejo le enkrat. Ker modul *Votingapi* ne omogoča zaklepanja glasovanja za določeno časovno obdobje, je bilo to treba implementirati posebej v `hook_plus1_widget_alter()` modula *Voting*. Funkcija preveri, ali je trenutni datum v obdobju, ko je glasovanje odprto. Če ni, onemogoči gradnik za oddajo glasu. Število glasov, ki jih lahko odda en uporabnik, nadzoruje in uveljavlja sam modul *Votingapi*.

Uporabniki oddajajo prispevke preko posebnega večstopenjskega formularja. Ta je implementiran s pomočjo programskega vmesnika modula *Ctools*. *Ctools*[27] je prispevani modul, ki vsebuje kar nekaj zelo močnih programskih vmesnikov. Med drugimi je tudi vmesnik za izdelavo čarovnikov (tj. večstopenjskih formularjev). Za izdelavo čarovnika je bilo treba registrirati menijski povratni klic na naslovu `'voting/%ctools_js/%node'` in v njem definirati strukturo čarovnika (glej izsek kode 4.1) ter nad njo poklicati funkcijo `ctools_wizard_multistep_form()`, ki poskrbi, da se zgradi formular za trenutni korak. Rezultat te funkcije je upodabljalno polje (angl. *angl. render array*), ki ga nato izpišemo s funkcijo `drupal_render()`.

```
$form_info = array(
  //form id
  'id' => 'voting-form',
  //pass the step we're on to the form, step1, step2, ..etc
  'path' => 'voting/' . ($js ? 'ajax' : 'nojs') . '/' . $voting_node->nid
      . "/%step",
  //show the breadcrumb / path trail for the forms
  'show trail' => TRUE,
  //show the back button
  'show back' => TRUE,
  //show the cancel button
  'show cancel' => ($step == '' || $step == 'step1'),
  //show the update and return button
  'show return' => FALSE,
  //a callback function to run when the next button is called
  'next callback' =>  'voting_wizard_next',
```

[25]http://drupal.org/project/votingapi
[26]http://drupal.org/project/plus1
[27]http://drupal.org/project/ctools

```
    //callback when finish button is called
    'finish callback' => 'voting_wizard_finish',
    //callback for cancel action
    'cancel callback' => 'voting_wizard_cancel',
    //this controls order, as well as form labels
    'order' => array(
      'step1' => t('<strong>1</strong>Enter the content'),
      'step2' => t('<strong>2</strong>Preview'),
      'step3' => t('<strong>3</strong>Send'),
    ),
    //here we map a step to a form id.
    'forms' => array(
      //what we're doing here is telling the wizard when step1
      //is passed as arg, show the form with form_id voting_form_step_1
      'step1' => array(
        'form id' => 'voting_form_step_1'
      ),
      'step2' => array(
        'form id' => 'voting_form_step_2'
      ),
      'step3' => array(
        'form id' => 'voting_form_step_3'
      ),
    ),
  );
```

Izsek kode 4.1: Primer definicije čarovnika - večstopenjskega formularja.

4.5.8 Implementacija čestitk

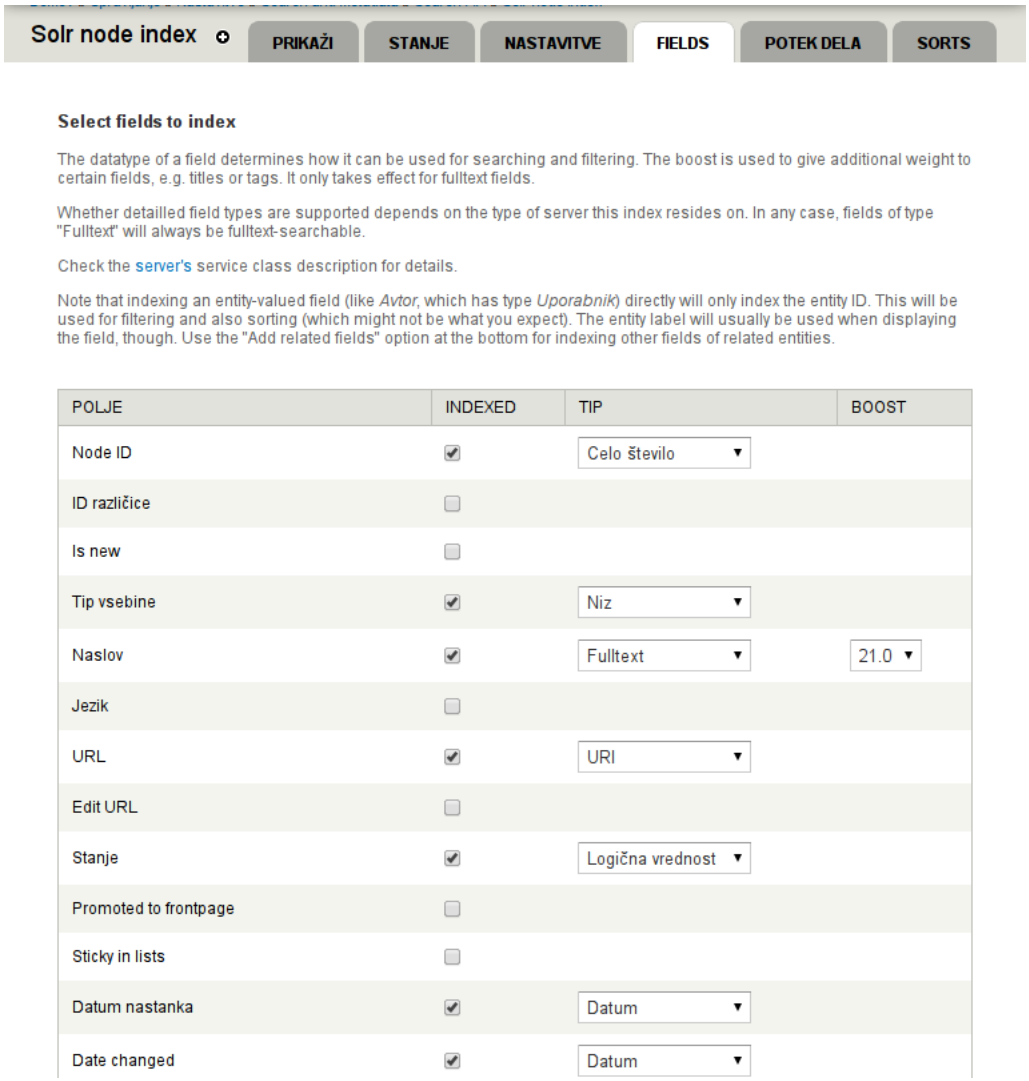
Čestitka je v CMS poseben tip vsebine. Predloge, ki so tudi poseben tip vsebine, so s čestitko povezane z poljem *nodereference*. Predloge definirajo uredniki, registrirani uporabniki pa jih lahko pri objavi čestitk izberejo prek spustnega seznama, ki ga na podlagi predlog, ki so na voljo, samodejno ustvari modul *Nodereference*. Podatki o predlogi se upoštevajo pri izrisu rezultata na ravni teme.

4.5.9 Implementacija iskalnika

Ker je volumen vsebine zelo velik, mora biti tehnologija, ki poganja iskalnik, zelo zmogljiva. Pri načrtovanju portala slovenskih novic smo upoštevali tudi

možnost razširitve iskalnika na ostale portale družbe Delo. V praksi bi to pomenilo, da se članki iz ostalih portalov prikazujejo kot rezultati na portalu slovenskih novic.

Drupal sicer v jedru vsebuje modul, ki implementira iskalnik, vendar je premalo zmogljiv že za najosnovnejše uporabniške primere, saj iskalni indeks shranjuje v podatkovno bazo, ne omogoča indeksiranja po posameznih poljih niti iskanja po delih besed.



Slika 4.15: Izsek iz nastavitvenega vmesnika za Solr indeks.

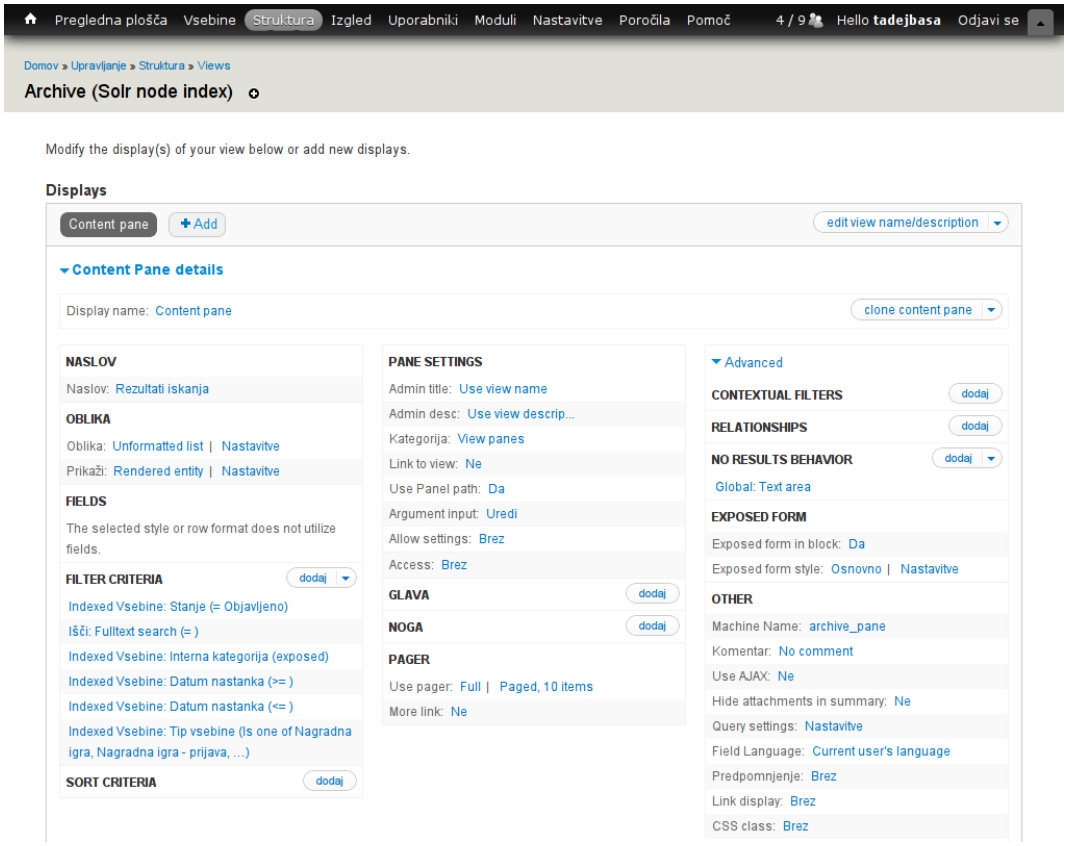
Zaradi naštetih zahtev in pomanjkljivosti, ki jih ima vgrajeni modul *Search*, je iskalnik na portalu "slovenskenovice.si" implementiran s pomočjo prispevanih modulov *Search API*[28] in *Search API Solr*[29].

Search API priskrbi programski vmesnik za implementacijo naprednejših iskalnikov. Omogoča integracijo z zunanjimi sistemi, ločevanje iskalnih indeksov, indeksiranje po posameznih vsebinskih poljih itd. *Search API Solr* omogoča integracijo s strežnikom za iskanje *Apache Solr*[30]. Več podrobnosti o *Apache Solr* najdete v poglavju 5.2.5.

Na sliki 4.15 je prikazan izsek iz nastavitvenega vmesnika za iskalnik indeks. V tabeli na sliki je seznam vseh vsebinskih polj. Pri posameznem polju določimo, ali je indeksirano, kakšnega tipa je in kakšna je njegova utež.

Prikaz člankov v arhivu dosežemo s pomočjo modula *Views*. Nastaviti je treba pogled, ki črpa rezultate iz iskalnega indeksa Solr. Kriterije iskanja vnašamo prek izpostavljenih filtrov. Konfiguracija pogleda je prikazana na sliki 4.16.

[28] http://drupal.org/project/search_api
[29] http://drupal.org/project/search_api_solr
[30] http://lucene.apache.org/solr/



Slika 4.16: Konfiguracija pogleda za prikaz iskalnika in rezultatov iskanja.

Poglavje 5

Strežniška infrastruktura

5.1 Strojna oprema

Portal "slovenskenovice.si" poganjata dva strežnika s popolnoma enako konfiguracijo strojne opreme. Gre za strežnika HP ProLiant DL380 G7, ki imata vsak dva šestjedrna procesorja Intel® Xeon® X5650 @2.67GHz, 48GB RAM in 1800GB trdega diska.

5.2 Programska oprema

Strežniki poganjajo klasični programski sklad LAMP, ki ga najpogosteje srečamo na podobnih projektih. Bistvene prednosti te rešitve so preizkušenost, stabilnost platforme. Na spletu zlahka najdemo podporo in pomoč pri reševanju vseh vrst problemov in izzivov, pred katere smo postavljeni pri implementaciji spletnih aplikacij.

5.2.1 Apache

Apache HTTP-strežnik[31] oziroma pogojsteje kar Apache je spletni strežnik, ki je odigral ključno vlogo pri hitri rasti svetovnega spleta. Od aprila 1996

[31] http://httpd.apache.org/

je najpogosteje uporabljen spletni strežnik. Ocena iz septembra 2012 kaže, da Apache postreže kar 54,98 % vseh strani na svetovnem spletu [16].

Prva različica je bila izdana leta 1995 in je bila osnovana na strežniku NCSA HTTPd. Od takrat je bil že v celoti prepisan in izdan pod odprtokodno licenco Apache License. Razvija in vzdržuje ga skupnost programerjev pod okriljem Apache Software Foundation. Teče na širokem spektru operacijskih sistemov, vključno z Unix, FreeBSD, Linux, OS X, Microsoft Windows in drugimi.

5.2.2 MySQL

MySQL[32] je najpogosteje uporabljen odprtokodni sistem za upravljanje z relacijskimi podatkovnimi bazami (angl. RDBMS). Poganja se kot strežnik, ki omogoča večuporabniški dostop do podatkovnih zbirk, iz katerih odjemalci z uporabo programskih vmesnikov berejo podatke s pomočjo jezika SQL. Razvilo ga je Švedsko podjetje MySQL AB in ga izdalo pod odprtokodno licenco GNU GPL. Prva različica je bila izdana 23. maja 1995. Podjetje MySQL AB je od leta 2010 last podjetja Oracle.

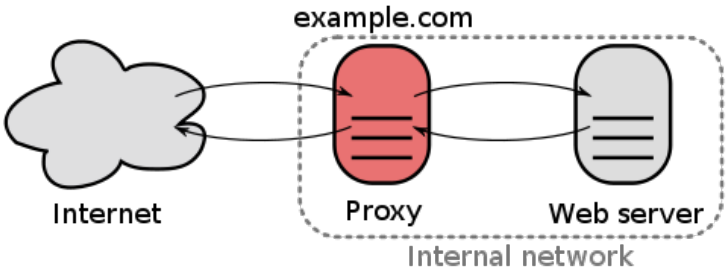
MySQL se pogosto uporablja v spletnih aplikacijah, saj je osrednja komponenta pogosto uporabljenega programskega sklada za spletne aplikacije LAMP (Linux, Apache, Mysql, PHP).

MySQL srečamo tako pri manjših spletnih aplikacijah in sistemih za upravljanje z vsebinami, kot so Wordpress, Typo3, Joomla, Drupal ... kot tudi pri večjih projektih ko so Wikipedia, Facebook, Twitter ...

5.2.3 Varnish

Varnish[33] je obrnjeni proks strežnik, ki omogoča velike pohitritve vsebinsko bogatih spletnih aplikacij. Zastavljen je kot HTTP-pospeševalnik in je, za razliko od ostalih proksi strežnikov (kot so Squid, nginx ...) osredotočen

[32] http://www.mysql.com/
[33] https://www.varnish-cache.org/



Slika 5.1: Obrnjen proksi strežnik prejme zahtevo s spleta in jo posreduje strežniku v privatnem omrežju ter jo nato vrne odjemalcu, kot da bi jo izvedel sam. Odjemalci se morda sploh ne zavedajo obstoja privatnega omrežja za proksi strežnikom.

izključno na HTTP protokol. Nastal je pri optimizaciji podobnega projekta kot je portal "slovenskenovice.si", in sicer na Norveškem v povezavi s spletno izdajo tabloida "Verdens Gang"[34]. Kasneje je bil programski paket izdan pod odprtokodno licenco BSD. Različica 1.0 pa je bila izdana leta 2006.
Obrnjeni proksi strežnik je tip proksi strežnika, ki v imenu odjemalca posreduje zahtevo enemu ali več strežnikom. Odjemalcu odgovor izgleda, kot da izvira iz obrnjenega proksi strežnika. Obrnjeni proksi strežnik si najlažje predstavljamo kot vmesnik med odjemalcem in njemu dodeljenimi strežniki (slika 5.1).

Varnish podatke shranjuje v navidezni pomnilnik in podrobnosti rokovanja s pomnilnikom prepušča operacijskemu sistemu. Deluje večnitno, vsako zahtevo odjemalca izvede svoja nit. Ko prekoračimo v konfiguraciji določeno število delovnih niti, se vse nadaljnje zahteve postavijo v čakalno vrsto. Šele, ko presežemo v konfiguraciji nastavljeno dolžino čakalne vrste, začne *Varnish* zahteve zavračati.

Varnish za svoje nastavitvene datotek uporablja poseben jezik *Varnish Configuration Language* (VCL). V nastavitvenih datotekah lahko pišemo neke vrste procedure, ki se pokličejo ob vsaki točki izvajanja zahteve.

[34]http://www.vg.no/

Varnish omogoča tudi izenačevanje obremenitve strežnikov, tako da zahteve posreduje strežnikom po principu Round-Robin ali po naključni metodi z obteževanjem. Pri tem lahko tudi preverja stanje strežnikov, tako da zahteve posreduje samo tistim, ki so dosegljivi.

5.2.4 Memcached

Memcached[35] je visoko zmogljiv, porazdeljeni sistem za predpomnenje podatkov v računalniškem pomnilniku. Kljub splošni zasnovi je v osnovi namenjen za uporabo pri razbremenitvi podatkovnega strežnika z namenom pohitritve spletnih aplikacij. Programski vmesnik *Memcached* nam omogoča dostop do shrambe v obliki velike razpršene tabele, ki se lahko razteza tudi fizično prek več računalnikov. *Memcached* deluje po principu strežnik-odjemalec. *Memcached* strežniki vzdržujejo shrambo tipa ključ-vrednost, odjemalci pa v shrambo pošiljajo podatke in iz nje berejo. Čeprav je shramba lahko fizično porazdeljena preko več strežnikov, vsi *Memcached* strežniki vidijo isto navidezno pomnilniško zalogo. Neki podatek je zato vedno shranjen na isti lokaciji v tem navideznem pomnilniku. Pri dostopu ali zapisu neke vrednosti odjemalec iz ključa najprej določi, kateri od *Memcached* strežnikov hrani dotičen podatek. Strežniki v glavnem pomnilniku rezervirajo zalogo nespremenljive velikosti. Če strežniku pri vpisu nove vrednosti zmanjka prostora, *Memcached* zavrže vrednost z najstarejšim časom dostopa, deluje po principu LRU (angl. least recently used)[36]. Aplikacije, ki uporabljajo *Memcached*, ga morajo obravnavati kot neobstojni pomnilnik.

5.2.5 Apache Solr

Solr[37] je hitra odprtokodna iskalna platforma, ki izhaja iz projekta Apache Lucene[38]. Njegove glavne prednosti so zmogljiv iskalnik po polnem besedilu,

[35] http://memcached.org/
[36] http://en.wikipedia.org/wiki/Cache_algorithms#Least_Recently_Used
[37] http://lucene.apache.org/solr/
[38] http://lucene.apache.org/

poudarjanje zadetkov, usmerjeno iskanje, integracija s podatkovno bazo ... *Solr* je visoko skalabilen, saj omogoča porazdeljeno iskanje in replikacijo indeksa.

Solr je napisan v Javi, poganjamo pa ga kot samostojen strežnik znotraj servlet vsebnika (npr. *Tomcat*). Uporabniki z njim komunicirajo s pomočjo programskih vmesnikov REST, HTTP/XML ali JSON. Indeksiranje se lahko vrši z vlaganjem dokumentov v obliki XML, JSON ali binarni obliki prek HTTP-protokola. Povpraševanje se izvaja z HTTP GET metodo, ki zopet vrne odgovor bodisi v obliki XML, JSON ali binarni obliki.

Za uporabo strežnika *Solr* iz jezika PHP, si lahko pomagamo s knjižnico *solr-php-client*[39]. Ta je tudi potrebna, če hočemo strežnik *Solr* uporabiti v povezavi z modulom *SearchAPI Solr* v Drupalu.

5.3 Opis konfiguracije

Arhitekturo sistema smo želeli že ob prvi izdaji karseda optimalno zastaviti. Iz izkušenj na manjših portalih, ki smo jih izdali v začetku leta 2011, smo sicer sklepali, da do večjih težav ne bi smelo prihajati. Ker pa je bilo takrat in v naslednjih mesecih pričakovano število obiskov za red velikosti večje, vseeno nismo želeli tvegati. Z zastavljeno arhitekturo smo želeli doseči pri danem številu obiskov čim manjšo obremenitev strežnikov, obenem hitro odzivnost (minimizirati čas čakanja uporabnika na zahtevano vsebino) in hkrati zagotoviti tudi delovanje strani ob primeru nenačrtovanih izpadov (okvara strojne opreme, vzdrževanje strežnikov itd.). Nujna je bila uvedba redundance na nivoju strojne opreme. Portal "slovenskenovice.si" zato trenutno teče na dveh strežnikih (glej poglavje 5.1), ki poganjata klasični programski sklad LAMP. Generirane strani za anonimne uporabnike predpomni strežnik *Varnish*, ki obenem skrbi za porazdeljevanje obremenitve med strežnikoma. Strežnike nadalje razbremenjujemo s predpomnilnikom (ki ga koristi Drupal) v gruči Memcache strežnikov. Slika 5.2 ponazarja arhitekturo portala "sloven-

[39] http://code.google.com/p/solr-php-client/

skenovice.si". Redundanco na nivoju izenačevalnika obremenitve dosežemo s podvojitvijo strežnika *Varnish* (več o tem še v poglavju 5.3.2).

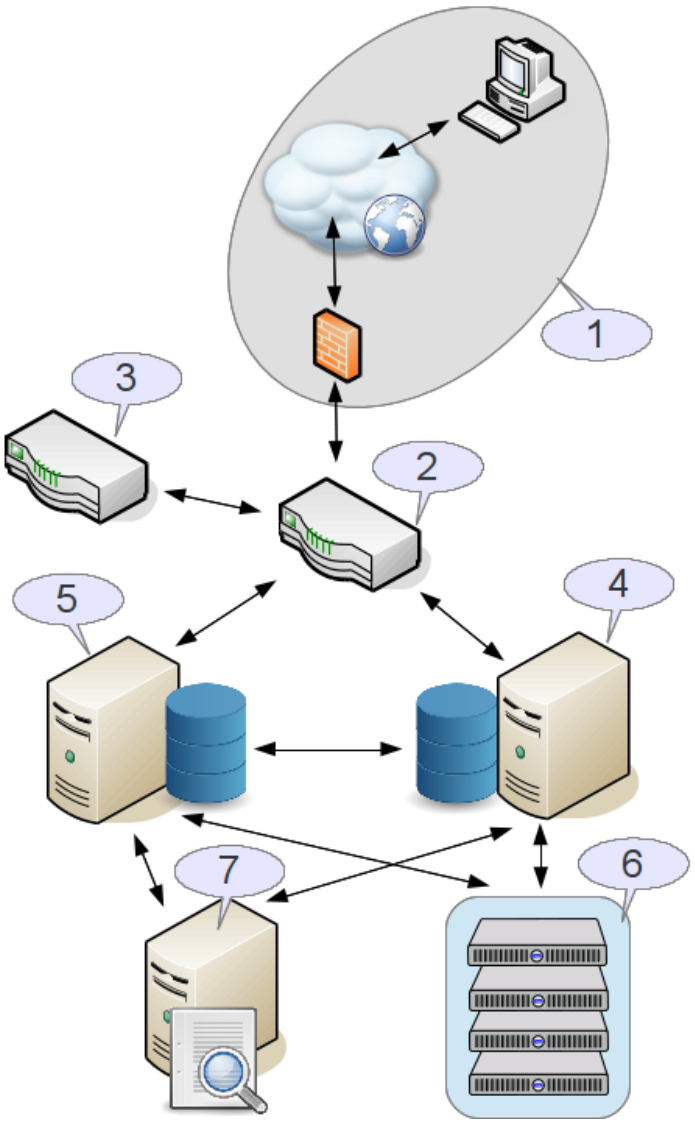
5.3.1 Predpomnenje

Eden osnovnih načinov za izboljšanje zmogljivosti je uvedba predpomnenja. Bistvo predpomnenja je v tem, da ne ponavljamo že opravljenega dela in na ta način razbremenimo strežnike. V kontekstu spletnih portalov to pomeni, da že prikazanih strani ne upodabljamo ob vsaki zahtevi, ampak upodobitev izvedemo samo ob prvi zahtevi, rezultat shranimo in pri vseh naslednjih zahtevah po tej isti strani, vrnemo shranjeno vrednost. Drupal omogoča predpomnenje na več nivojih (poljubnih vmesnih rezultatov, blokov in celotnih strani), za vse pa uporablja isti mehanizem, ki je razvijalcem na voljo preko programskega vmesnika *Cache*. Najpomembnejši sta funkciji: `cache_set()` in `cache_get()`.

```
cache_set($cid, $data, $bin = 'cache', $expire = CACHE_PERMANENT)

cache_get($cid, $bin = 'cache')
```

V predpomnilniku lahko posamezne zapise organiziramo v skupine oziroma množice. Zato je v zgornjih deklaracijah zanimiv predvsem parameter `$bin`. Z njim povemo, v katero množico hočemo shraniti dani podatek. To je pomembno, ker lahko za posamezne množice določimo način hranjenja podatkov. Vsak način hranjenja podatkov mora implementirati vmesnik `DrupalCacheInterface`. Drupal privzeto uporablja hranjenje v podatkovni bazi, ki je implementirano v razredu `DrupalDatabaseCache`. Ta implementacija hrani množice v istoimenskih tabelah v podatkovni bazi. Slabost te implementacije je, da je počasna. Vsak klic funkcije `cache_set()` ali `cache_get()` namreč pomeni novo SQL-poizvedbo. Če hočemo pohitriti izvajanje Drupala, moramo hraniti čim več vmesnih rezultatov. Ker pa so ti vmesni rezultati največkrat rezultat neke druge SQL-poizvedbe, prihranki v tem primeru ne bi bili tako veliki. Zato smo želeli za hranjenje vmesnih rezultatov uporabiti hitrejšo tehnologijo. Odločili smo se za prej opisani *memcached* (glej



Slika 5.2: Arhitektura sistema portala "slovenskenovice.si". Zahteva (1) se porodi pri odjemalcu na svetovnem spletu in se prek požarnega zidu posreduje strežniku *Varnish* (2 ali 3). *Varnish* zahtevo v primeru, da jo že ima v predpomnilniku, takoj vrne odjemalcu, sicer pa jo posreduje strežniku LAMP (4 in 5). MySQL-strežnika delujeta v konfiguraciji master-master (glej poglavje 5.3.3). Obremenitev strežnikov poskušamo minimizirati z uporabo predpomnilnika Memcache (6). Če se zahteva nanaša na arhiv vsebin, se posreduje naprej strežniku Solr (7).

poglavje 5.2.4), katerega vmesnik je implementiran v modulu Memcache[40]. Če ga hočemo uporabiti, je treba v datoteki `settings.php` določiti, katere množice želimo, da se vanj shranjujejo. Primer, kako uporabimo *memcached* kot privzeti način hranjenja podatkov za predpomnilnik, je prikazan v izseku kode 5.1.

```
include_once('./includes/cache.inc');
include_once('./sites/default/modules/memcache/memcache.inc');
$conf['cache_default_class'] = 'MemCacheDrupal';
```

Izsek kode 5.1: Konfiguracija modula Memcache.

Kot sem že prej omenil, *memcached* ni trajen predpomnilnik. To pomeni, da se nekateri podatki lahko po določenem času iz njega izbrišejo (če je strežnik zelo obremenjen, je to načeloma lahko že ob naslednji zahtevi), to pa lahko privede do marsikatere težave. Če vanj shranjujemo predpomnilnik formularjev, lahko izgubimo vnesene podatke. Če vanj shranjujemo podatke o sejah prijavljenih uporabnikov, so ti lahko naključno odjavljeni,... Zato je pomembno, da temeljito razmislimo, kateri podatki morajo biti obstojni in takih ne shranjujemo v *memcached*. Izsek kode 5.2 prikazuje konfiguracijo predpomnilnika za portal "slovenskenovice.si".

```
#Default cache in Memcache
include_once('./includes/cache.inc');
include_once('./sites/all/modules/contrib/memcache/memcache.inc');

$conf['memcache_servers'] = array(
  'x.x.x.x:y' => 'default',
  'x.x.x.x:y' => 'menu',
);
$conf['memcache_bins'] = array(
  'cache' => 'default',
  'cache_menu' => 'menu',
);

$conf['memcache_key_prefix'] = 'prefix';

# Move bins to backends.
$conf['cache_class_cache'] = 'MemCacheDrupal';
$conf['cache_class_cache_path'] = 'MemCacheDrupal';
$conf['cache_class_cache_rules'] = 'MemCacheDrupal';
```

[40]http://drupal.org/project/memcache

```
$conf['cache_class_cache_styles'] = 'MemCacheDrupal';
$conf['cache_class_cache_token'] = 'MemCacheDrupal';
$conf['cache_class_cache_update'] = 'MemCacheDrupal';
$conf['cache_class_cache_image'] = 'MemCacheDrupal';
$conf['cache_class_cache_hirerchical_select'] = 'MemCacheDrupal';
$conf['cache_class_cache_field'] = 'MemCacheDrupal';
$conf['cache_class_cache_block'] = 'MemCacheDrupal';
$conf['cache_class_cache_admin_menu'] = 'MemCacheDrupal';
$conf['cache_class_cache_libraries'] = 'MemCacheDrupal';
$conf['cache_class_cache_metatag'] = 'MemCacheDrupal';
$conf['cache_class_cache_menu'] = 'MemCacheDrupal';
$conf['cache_class_cache_views'] = 'MemCacheDrupal';
$conf['cache_class_cache_bootstrap'] = 'MemCacheDrupal';
```

Izsek kode 5.2: Konfiguracija modula *Memcache*.

5.3.2 Konfiguracija strežnika *Varnish*

V konfiguraciji, prikazani na sliki 5.2, *Varnish* igra dvojno vlogo:

- Predstavlja **izenačevalnik obremenitve**, ki skrbi, da zahteve enakomerno obremenjujejo oba strežnika. V primeru izpada enega ali drugega aplikacijskega strežnika poskrbi, da se vse zahteve posredujejo delujočemu strežniku.
- **Predpomni** v predhodnih zahtevah že prikazane strani in jih neprijavljenim uporabnikom vrača neposredno ter tako razbremenjuje strežnike LAMP.

Izenačevalnik obremenitve v naši arhitekturi predstavlja kritično točko odpovedi[41]. Ta problem smo rešili s podvojitvijo tudi na tem nivoju. Sekundarni strežnik *Varnish*, enako kot primarni, predpomni strani, ki jih zgenerira Drupal, obenem pa neprestano preverja stanje primarnega *Varnish* strežnika. Če pride do neodzivanja primarnega strežnika *Varnish*, sekundarni zamenja svoj IP-naslov in tako prevzame vlogo primarnega. Sistemski administrator ima tako čas, da odpravi napako na okvarjenem strežniku, ne da bi pri tem portal utrpel daljši časovni izpad.

[41] http://en.wikipedia.org/wiki/Single_point_of_failure

Ker opis konfiguracije strežnika *Varnish* presega okvire te naloge, si več o tem lahko bralec prebere v [19]. Za integracijo s strežnikom *Varnish* potrebujemo Drupalov modul *Varnish HTTP Accelerator Integration*[42]. Izsek kode 5.3 iz nastavitvene datoteke `settings.php` predstavlja potrebno konfiguracijo za vključitev Drupalovega predpomnilnika strani v strežniku *Varnish*.

```
## Add Varnish as the page cache handler.
$conf['cache_backends'][] = 'sites/all/modules/varnish/varnish.cache.inc';
$conf['cache_class_cache_page'] = 'VarnishCache';
# Drupal 7 does not cache pages when we invoke hooks during bootstrap.
# This needs to be disabled.
$conf['page_cache_invoke_hooks'] = FALSE;
```

Izsek kode 5.3: Konfiguracija modula za integracijo s strežnikom Varnish.

5.3.3 Replikacija podatkovne baze

V naši arhitekturi LAMP strežnika delujeta vzporedno. Neodvisno sprejemata zahteve in jih vračata strežniku *Varnish*. Očitno je seveda, da morata oba hraniti enako množico podatkov. V primeru izpada enega ali drugega mora portal nemoteno delovati naprej. Podatkovna strežnika morata biti zato neprestano sinhronizirana. Na nivoju podatkovne baze to dosežemo z replikacijo. Najlažje implementiramo replikacijo tipa master-master, saj ta ne zahteva nobene spremembe spletne aplikacije. V tem načinu delovanja spletna aplikacija normalno dostopa do podatkovne baze, spremembe pa se samodejno podvajajo med podatkovnima strežnikoma.

Problemi, ki se pojavijo pri taki obliki replikacije so kolizije `AUTO_INCREMENT` ključev. To rešimo tako, da začetne vrednosti ključev `AUTO_INCREMENT` nastavimo na različne zaporedne vrednosti, vrednost povečevanja ključev `AUTO_INCREMENT` pa nastavimo na 2. Tako bodo ključi `AUTO_INCREMENT` na enem strežniku vedno sodi, na drugem pa vedno lihi.

```
[mysqld]
server-id = 1
replicate-same-server-id = 0
auto-increment-increment = 2
```

[42] http://drupal.org/project/varnish

```
auto-increment-offset = 1

master-host = <IP address of Server 1>
master-user = <slave user>
master-password = <slave password>
master-connect-retry = 60
replicate-do-db = <database name>
```

Izsek kode 5.4: Nastavitev strežnika 1.

```
[mysqld]
server-id = 2
replicate-same-server-id = 0
auto-increment-increment = 2
auto-increment-offset = 2

master-host = <IP address of Server 2>
master-user = <slave user>
master-password = <slave password>
master-connect-retry = 60
replicate-do-db = <database name>
```

Izsek kode 5.5: Nastavitev strežnika 2.

Drugi problem je urejanje iste vsebine na različnih strežnikih. V teh primerih lahko pride do velikih težav, saj bi ob sočasnem urejanju vedno obveljala nazadnje potrjena sprememba. Obnašanje sistema bi bilo zelo netransparentno. Takim situacijam smo se želeli izogniti. V naši konfiguraciji zato Drupal vedno uporablja le en podatkovni strežnik, drugi pa je namenjen le redundanci. V primeru težav s primarnim strežnikom, lahko preprosto zamenjamo zapis v Drupalovi nastavitveni datoteki in portal deluje nemoteno naprej. Izsek relevantne konfiguracije iz datoteke `settings.php` je prikazan spodaj.

```
$databases['default']['default'] = array(
      'driver' => 'mysql',
      'database' => 'databasename',
      'username' => 'xxxxx',
      'password' => 'xxxxx',
      'host' => 'mysql-server',
      'prefix' => '',
);
```

Izsek kode 5.6: Nastavitev povezave s podatkovnim strežnikom v datoteki

`settings.php`.

5.3.4 Posebni izzivi, ki so posledica izbrane arhitekture

- **Določitev IP-uporabnika**: Ker v našem primeru aplikacijski strežnik komunicira vedno le s strežnikom *Varnish* , iz njegovega stališča zahteve uporabnikov vedno prihajajo iz istega IP-naslova. V sistemu Drupal tako odpovejo vse funkcije, ki so vezane na ta podatek: ne moremo več ločevati med uporabniki, ki so že glasovali v anketi ali nagradni igri, ne moremo ustrezno slediti komentatorjem itd. Izkaže se, da je Drupal že ustrezno pripravljen na to situacijo. V nastavitveni datoteki `settings.php` je potrebno nastaviti parameter `$conf['reverse_proxy-']=TRUE;`. V tem primeru Drupal za IP-uporabnika vzame vrednost, ki jo v glavi HTTP zahteve posreduje *Varnish*.

- **Razporeditev predpomnilnika**: Kot sem zapisal že prej, je *Memcache* predpomnilnik neobstojen. Zato je pomembno, da vanj ne usmerjamo množic, ki se morajo ohranjati preko več zahtev. Primer take množice je predpomnilnik formularjev. Drupal vanj zapisuje podatke o vnosih v formular in varnostne žetone, ki jih preverja ob predložitvi forme. Če se ta predpomnilnik izbriše, se podatki izgubijo in Drupal "zazna" neveljaven vnos. Množico `cache_form` moramo zato nujno usmeriti v obstojen pomnilnik (v našem primeru je to podatkovna baza).

  Po drugi strani so v specifičnih konfiguracijah nekatere množice zelo velike. V našem primeru je bila to množica `cache_menu`, v kateri Drupal predpomni razrešene podatke o upravljalcu specifične interne poti (npr. `node/1`). Zaradi velikega števila člankov in obiskov strani je ta množica naraščala na velikost 2GB in več, kar je posledično povzročalo velike težave pri postavitvi novih funkcionalnosti, ko je največkrat treba tudi izprazniti predpomnilnik.

- **Usmerjanje prijavljenih uporabnikov na pravi strežnik**: V predstavljeni arhitekturi sistema se *Varnish* odloča, na kateri strežnik usmeri neko zahtevo. V primeru zahteve anonimnega uporabnika to ne predstavlja nobene težave, saj so te zahteve vedno le bralne. Težave pa nastanejo v primeru prijavljenih uporabnikov, še posebej urednikov. Ti imajo namreč pravico nalagati datoteke, ki jih prikažejo v člankih. Ker je nalaganje asinhrono (med urejanjem članka lahko naložimo več datotek), se lahko zgodi, da se datoteka shrani v imenik "/tmp" na prvem strežniku, uporabnik pa formular predloži na drugi strežnik, kjer te datoteke v imeniku "/tmp" seveda ne bo. Nalaganje datoteke bo spodletelo, saj je Drupal na pričakovanem mestu ne bo našel. Prijavljene uporabnike je zato treba usmerjati vedno na isti strežnik. To dosežemo tako, da ob prijavi uporabniku pustimo piškotek (angl. *cookie*), v katerem si zapomnimo, na kateri strežnik je bil usmerjen. Ob vseh nadaljnjih zahtevah lahko z upoštevanjem vrednosti v tem piškotku uporabnika vedno usmerimo na pravi strežnik.

- **Ignoriranje nepomembnih piškotkov**: *Varnish* vrača rezultate neposredno iz predpomnilnika le za zahteve neprijavljenih uporabnikov. Vse ostale zahteve posreduje strežniku LAMP. Ali gre za zahtevo prijavljenega uporabnika ali ne, *Varnish* določi na podlagi prisotnosti kakršnegakoli piškotka. Ker pa le en piškotek, ki ga pusti Drupal, pomeni, da je uporabnik prijavljen, lahko *Varnish* vse ostale ignorira. Upoštevati je treba le piškotek z imenom, ki se začne z nizom "SESS" (npr. SESS797294cd3a93256631fb852630ae867a).

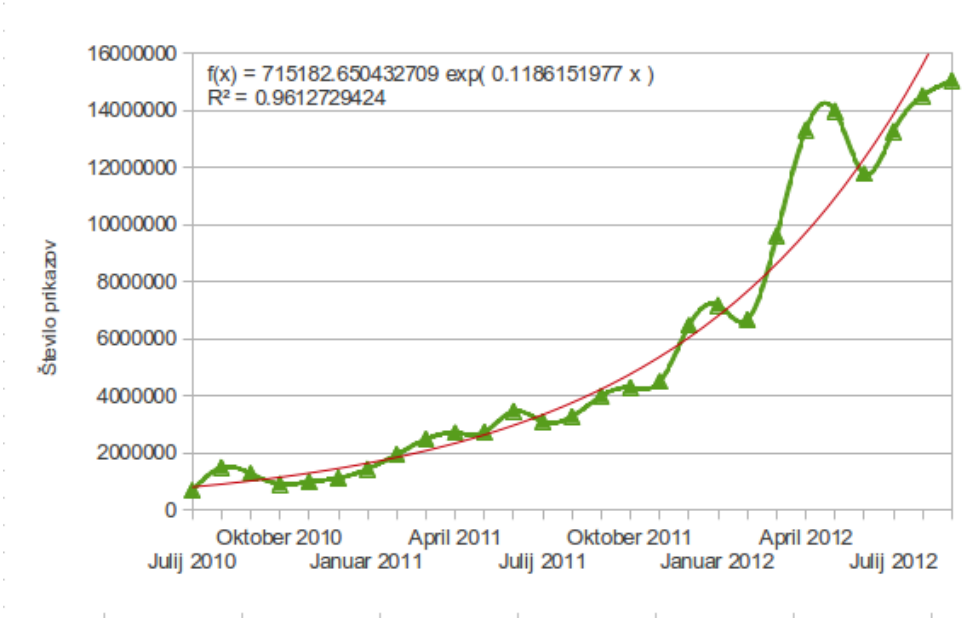
Poglavje 6

Zaključek

S pomočjo odprtokodnega sistema za upravljanje z vsebinami Drupal in ostalih odprtokodnih programov smo uspeli v relativno kratkem času implementirati razmeroma zahteven medijski portal. Menim, da so bili glavni cilji projekta doseženi. Odziv uporabnikov ob izdaji prenovljenega portala je bil pozitiven, število obiskov vztrajno narašča. Iz grafa na sliki 6.1 je razvidno, da je obisk po prenovi skokovito narasel. Trend rasti ostaja okrog 11 % mesečno. Iz tega lahko sklepamo, da so obiskovalci spletnega portala s prenovljenim portalom zadovoljni.

Ker je projekt predstavljal prenovo obstoječega portala in prehod na novo, uredniški ekipi neznano platformo, nas je seveda zanimal tudi njihov odziv. Že med razvojem, v katerega je bil kot produktni vodja vključen pomočnik odgovornega urednika portala, smo poskušali upoštevati čim več pripomb, želja in navad te ključne skupine uporabnikov. Kljub vsemu je menjava sistema zahtevala uvedbe novih konceptov in novega načina dela. Uredniška ekipa jih je brez velikih težav v zelo kratkem času vzela za svoje. Ugotavljajo celo, da so na novem sistemu, v novem okolju in z novim načinom dela povprečno 25 % učinkovitejši kot na starem sistemu.

Z vidika razvojne skupine se je izbira odprtokodne programske opreme, predvsem platforme Drupal, izkazala kot ključen dejavnik za uspešno izvedbo projekta. Zaradi razširljivosti platforme Drupal in velike skupnosti uporab-



Slika 6.1: Graf obiskanosti spletnega portala "slovenskenovice.si". Vir: MOSS: September 2012.

nikov, ki svoje rešitve objavlja na spletni strani drupal.org, smo lahko v svojo rešitev vključili številne prispevane module. Z njimi smo lahko pokrili večino naročnikovih zahtev. Kjer so bile zahtevane dodatne spremembe, smo lahko zaradi močnega programskega vmesnika, dobro dokumentirane kode in na sploh množice dokumentacije in receptov, ki jih najdemo na svetovnem spletu, hitro sprogramirali ustrezno rešitev. Za težave, ki smo jih odkrili med razvojem, so velikokrat že obstajali opisi rešitev na forumih ali celo programski popravki na seznamu odprtih problemov določenega modula.

Negativna plat Drupala je predvsem kompleksnost in specifičnost platforme. Zaradi nekaterih razvojnih prijemov, ki so lastni Drupalu in niso široko uveljavljeni v ostalih odprtokodnih projektih (na primer mehanizem kljuk), je uvajanje novih razvijalcev, ki se z Drupalom srečajo prvič, precej oteženo in dolgotrajno. Velikost odprtokodne skupnosti za Drupalom pomeni tudi raznovrstnost vključenih razvijalcev, njihovih delovnih navad, odnosa do

ostalih razvijalcev, natančnost in doslednosti pri razvoju itd. Kvaliteta kode prispevanih modulov lahko zelo variira in treba biti je zelo previden, saj lahko ob nepremišljenem vključevanju prispevanih modulov nehote odpremo varnostne luknje ali povzročimo izgubo podatkov. Preveriti je treba kodo vsakega modula ter opraviti integracijsko testiranje.

Kot razvojna ekipa smo se poskušali čimbolj vključiti v skupnost uporabnikov Drupala. Lastne module smo se trudili zastaviti čimbolj splošno ter jih, kjer je bilo smiselno, objavili v repozitoriju drupal.org. Aktivno smo sodelovali v seznamih odprtih problemov s testiranjem objavljenih programskih popravkov in prispevanjem svojih.

Kljub vsem novim in inovativnim prijemom, ki smo jih uporabili pri razvoju portala, še vedno ostaja tudi prostor za izboljšave. Najprej je treba omeniti uporabo omrežnega datotečnega sistema za shranjevanje datotek v CMS. Trenutni mehanizem s periodično sinhronizacijo je časovno in prostorsko potraten, vodi pa tudi do zgoraj opisnih problemov in posledično v slabšo razporeditev obremenitve strežnikov. Več pozornosti bi bilo smiselno posvetiti tudi čelnemu delu sistema, to je tistemu, ki ga opazi končni uporabnik. Na tem segmentu je zelo pomembna prilagoditev oblikovne predloge za mobilne naprave in izdelava prave odzivne spletne strani (angl. *responsive design*), ter optimizacija kode JS in CSS. Z optimizacijo grafičnih elementov je možno dodano zmanjšanje števila zahtev, ter tako pohitriti nalaganje strani v brskalniku. Dodatne razbremenitve sistema bi lahko dosegli z uvedbo predpomnenja posameznih blokov tudi za registrirane uporabnike z uporabo tehnologije ESI[43].

Za konec lahko rečem, da je bil razvoj portala "slovenskenovice.si" pozitivna izkušnja, pri kateri smo imeli vsi vpleteni priložnost pridobiti nova znanja tako na področju vodenja projekta, razvoja s pomočjo odprtokodnih programov, pa tudi na področju načrtovanja in implementacije arhitekture sistema visoke zmogljivosti. Za vse vključene razvijalce je bil to projekt z do tedaj največjim številom uporabnikov. Menim, da smo udeleženi svoje

[43]http://en.wikipedia.org/wiki/Edge_Side_Includes

delo opravili kvalitetno, o čemer pričajo tudi statistike obiska novega portala. Izkazano zaupanje in zadovoljstvo uporabnikov je najboljša popotnica za prihodnost.

Literatura

[1] M. Butcher, G. Dunlap, et al, *Drupal 7 Module Development: Create your own Drupal 7 modules from scratch*, Packt publishing, 2010.

[2] M. Cohn, *User stories applied*, Adison-Wesley, 2004.

[3] C. A. Kenwood, *A Business Case Study of Open Source Software*, Julij 2001, Dostopno na:
http://www.mitre.org/work/tech_papers/tech_papers_01/kenwood_software/kenwood_software.pdf

[4] V. Mahnič, *Problemi in rešitve pri uvajanju metode Scrum v proces razvoja programske opreme*, Zbornik referatov Dnevi slovenske informatike, Portorož, april 2011.

[5] D. Mercer, *Drupal 7: Create and operate any type of website quickly and efficiently*, Packt publishing, 2010.

[6] E. Miles, L. Miles, et al, *Drupal's Building Blocks: Quickly building web sites with CCK, Views, and Panels*, Addison-Wesley, 2010.

[7] K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2004.

[8] R. Stallman, *GNU Manifest*, Dostopno na:
http://www.gnu.org/gnu/manifesto.html

[9] T. Tomlinson, J. K. VanDyk, *Pro Drupal 7 Development: Third Edition*, Apress, 2010.

[10] J. Urevc, *Agilni razvoj časopisnega portala po metodi Scrum*, Diplomska naloga. Fakulteta za računalništvo in informatiko. Ljubljana, 2012.

[11] M. Zandstra, *PHP Objects, Patterns and Practice (Expert's Voice in Open Source)*, Apress, 2010.

[12] *Agile software development*, Wikipedia, Dostopno na: http://en.wikipedia.org/wiki/Agile_software_development

[13] *Benefits of using Open Source*, Dostopno na: http://open-source.gbdirect.co.uk/migration/benefit.html, 5. 12. 2012

[14] *Future of open-source survey*, Dostopno na: http://www.slideshare.net/blackducksoftware/the-2012-future-of-open-source-survey-results.

[15] *2011 Open Source CMS Market Share Report (Nov. 11)*, Dostopno na: http://www.waterandstone.com/book/2011-open-source-cms-market-share-report, 5. 12. 2012

[16] *September 2012 Web Server Survey*, Dostopno na: http://news.netcraft.com/archives/2012/09/10/september-2012-web-server-survey.html, 5. 12. 2012

[17] *Wappalyzer*, Dostopno na: http://wappalyzer.com/categories/cms?utm_source=chrome&utm_medium=extension&utm_campaign=extensions, 5. 12. 2012

[18] *What is free software?*, Dostopno na: http://www.fsf.org/philosophy/free-sw.html, 5. 12. 2012

[19] N. Haug, *Configuring Varnish for High-Availability with Multiple Web Servers*, Dostopno na: http://www.lullabot.com/articles/varnish-multiple-web-servers-drupal