UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Luka Žabkar

Razvoj sodobne storitvene spletne strani za sledenje dieti in aktivnostim

DIPLOMSKO DELO
NA VISOKOŠOLSKEM STROKOVNEM ŠTUDIJU

Mentor: doc. dr. Peter Peer

Ljubljana, 2011

Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

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: 00176/2011
Datum: 02.11.2011

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

Kandidat: **LUKA ŽABKAR**

Naslov: **RAZVOJ SODOBNE STORITVENE SPLETNE STRANI ZA SLEDENJE DIETI IN AKTIVNOSTIM**

DEVELOPMENT OF A CONTEMPORARY SERVICE-BASED WEBSITE FOR DIET AND ACTIVITY TRACKING

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Predstavite razvoj svetovnega spleta do danes, aktualne tehnologije in knjižnice za razvoj sodobnih storitvenih spletnih strani ter dobre paradigme takšnega razvoja. Ugotovitve iz predstavitve praktično demonstrirajte na razvoju storitvene spletne strani.

Mentor:
doc. dr. Peter Peer

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:
prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Luka Žabkar,

z vpisno številko 63030192,

sem avtor diplomskega dela z naslovom:

Razvoj sodobne storitvene spletne strani za sledenje dieti in aktivnostim

S svojim podpisom zagotavljam, da:

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

V Ljubljani, dne 6. 12. 2011 Podpis avtorja:

Zahvala

Za pomoč in potrpljenje bi se na tem mestu zahvalil staršema in mentorju Petru Peeru. Za lektoriranje se zahvaljujem Nataši Meh Peer.

Posvečeno Nikitty, beli milini.

Kazalo

Seznam uporabljenih kratic in simbolov

- AJAX *(angl. Asynchronous JavaScript and XML)* – Asinhroni JavaScript in XML
- CSRF *(angl. Cross-site request forgery)* – CSRF napad
- CERN *(angl. European Organization for Nuclear Research Organisation)* – Evropska organizacija za jedrske raziskave
- CGI *(angl. Common Gateway Interface)* – Skupni prehodni vmesnik
- CSS *(angl. Cascading Style Sheets)* – Prekrivne predloge
- DOM *(angl. Document Object Model)* – Objektni model dokumenta
- DoS *(angl. denial of service)* – Ohromitev storitve
- FBML *(angl. Facebook markup language)* – Označevalni jezik za delo s facebookom
- FQL *(angl. Facebook query language)* – Povpraševalni jezik za delo s facebookom
- GPL *(angl. GNU General Public License)* – Licenca GNU
- HTML *(angl. HyperText Markup Language)* – Označevalni jezik za oblikovanje večpredstavnostnih dokumentov
- JSON *(angl. JavaScript Object Notation)* – JavaScript standard za zapis objektov
- MCP *(angl. Moderator control panel)* – Nadzorna plošča za moderatorje

- MVC *(angl. Model-view-controller)* – Model-pogled-krmilnik
- PHP *(angl. Hypertext Preprocessor)* – Skriptni programski jezik
- RBAC *(angl. Role based access control)* – Seznam za kontrolo dostopa, ki temelji na vlogah
- RDF *(angl. resource description framework)* – Ogrodje za opis virov
- RSS *(angl. really simple syndication)* – Protokol za objavo in distribucijo spletnih vsebin v zapisu XML
- SEO *(angl. search engine optimization)* – Optimizacija spletnih strani
- SGML *(angl. standardized generalized markup language)* – Standardizirani splošni označevalni jezik
- SQL *(angl. structured query language)* – Strukturiran povpraševalni jezik za delo s podatkovnimi bazami
- SUPB – Sistem za upravljanje podatkovnih baz
- UCP *(angl. User control panel)* – Nadzorna plošča za uporabnike
- URI *(angl. Uniform resource identifier)* – Enotni označevalnik vira
- W3C *(angl. World Wide Web Consortium)* – Mednarodni inštitut, kjer člani organizacije, osebje s polnim delovnim časom in javnost sodelujejo ter skupaj razvijajo standarde za splet
- XHTML *(angl. extensible hipertext markup language)* – Razširljiv jezik za označevanje nadbesedila
- XML *(angl. Extensible Markup Language)* – Razširljivi označevalni jezik
- XSS *(angl. Cross-site scripting)* – XSS napad

Povzetek

Nepravilne, prekomerne in nezdrave prehrambene navade v zahodnem svetu negativno vplivajo na zdravje in počutje prebivalstva, poleg tega pa zdravljenje posledic predstavlja tudi velik strošek v zdravstvenem proračunu. Odločili smo se razviti sodobno spletno stran, ki uporabnikom ponuja storitve, kot so sledenje prehrani, beleženje aktivnosti in iskanje informacij o sestavi živilih.

V diplomski nalogi smo predstavili stanje na trgu spletnih strani, najbolj pogosto uporabljene tehnologije in pristope ter pričakovanja sodobnih uporabnikov. Z opisom koncepta spleta 2.0 smo pokazali vlogo, ki jo igrajo uporabniki pri ustvarjanju vsebine ter primernost integracije družabnega omrežja Facebook v našo spletno stran. Pozornost smo namenili tudi dostopu preko mobilnih naprav in uporabi prilagojenih vmesnikov, varnosti v spletnih aplikacijah in uporabi tehnik za optimizacijo vidljivosti v iskalnikih.

Predstavili smo uporabljene tehnologije in razloge za izbiro posameznih knjižnic ter storitev. Uporabniški vmesnik smo popestrili z uporabo tehnik AJAX in izkušnjo uporabe spletne strani približali naprednim sodobnim spletnim stranem. Skozi pregled našega podatkovnega modela smo prikazali objekte, ki smo jih modelirali v sistemu. Pokazali smo, da je objektni razvoj v programskem jeziku PHP dobra izbira za razvoj kompleksnih spletnih strani in bolj primeren za implementacijo uporabljenih načrtovalnih vzorcev, kot postopkovno programiranje. Med razvojem sistema smo ugotovili tudi, da je veliko pozornosti pri razvoju večuporabniških spletnih strani potrebno nameniti varnosti in zagotavljanju integritete spletne izkušnje.

Tekom razvoja smo bili osredotočeni na odprtost rešitev in pripravljenost na bodoče razširitve, kot je uporaba meta značk semantičnega spleta. Ugotovili smo, da je naša spletna stran primerno zastavljena za nadaljnji razvoj in dosega zadane cilje.

Ključne besede:

razvoj spletnih strani, predložni stroji, mobilne naprave, Facebook, MVC, lokalizacija, PHP, AJAX, MySQL, CSS, HTML, jQuery.

Abstract

Improper, excessive and unhealthy nutrition habits in the western world adversely affect the health and well-being of the general population, while also straining national health budgets due to increased need for medical interventions. Thus, we developed a contemporary website, offering its users services such as diet tracking, activity logging and nutrition information lookup.

In the thesis we presented the state of the website market, commonly used technologies and techniques in website development, and contemporary customer expectations. By conveying the concept of Web 2.0, we demonstrated the importance of user generated content and the applicability of integrating the social network Facebook with our website. We also focus on mobile device accessibility and use of customized interfaces, security in web applications and the use of search engine optimization techniques.

We demonstrated the technologies utilized and the arguments for integrating various libraries and services. AJAX techniques allowed us to enrich the graphical user interface and elevate the user experience to the level of rich contemporary websites. By delving into the details of our data model, we exhibited most objects used to model our system. We continued by establishing that object oriented programming in PHP is a favorable choice when developing complex websites and preferred to procedural programming when implementing design patterns. While developing our system we affirmed the importance of security and integrity maintenance in multiuser website environments.

During the development process much attention was dedicated to open solutions and expandability, such as employing semantic web meta-tags. We concluded that the developed website is appropriately coded for further development and achieves set objectives.

Key words:
website development, template engines, mobile devices, Facebook, MVC, localization, PHP, AJAX, MySQL, CSS, HTML, jQuery.

Poglavje 1

Uvod

Od sodobnih spletnih strani uporabniki pričakujejo veliko več funkcionalnosti kot pred leti, zato je potrebno za njihov razvoj poseči po kopici orodij in knjižnic, ki pohitrijo razvoj in hkrati nudijo uporabnikom občutek znanega. Uporabniki sodobnih spletnih strani so pogosto tudi soustvarjalci vsebine in pametna integracija uporabniške vsebine je bistvena za uspešen življenjski cikel spletne strani kot je naša.

Sledenje, reguliranje in uravnavanje vnosa energije in osnovnih gradnikov preko prehranjevanja in ustrezna poraba energije z gibanjem, nam v sodobni družbi pomagajo premagati mnoge težave povezane z debelostjo in nepravilno prehrano. Z našo aplikacijo je mogoče lažje slediti razmerju med porabo in zaužitimi kalorijami, takšna preventiva pa lahko pomaga preprečiti mnoge bolezni sodobne družbe, kot sta na primer sladkorna bolezen in miokardni infarkt. Odrasel človek težak 70 kg potrebuje za zadovoljitev dnevnih metaboličnih potreb od 1920 do 2900 kcal energije, odvisno od fizične aktivnosti. Večino te energije ljudje dobimo od ogljikov hidratov (40-60%), maščob (večinoma triliceridi, 30-40%) in beljakovin (10-15%), nekaj pa tudi iz alkoholov. Če je vnos metaboličnega goriva večji kot poraba energije, se odvečno gorivo hrani v obliki trigliceridov in telesne maščobe, kar vodi v razvoj debelosti in z debelostjo povezanih zdravstvenih nevarnosti. Nasprotno pa se v primeru konstantnega premajhnega vnosa metaboličnega goriva zaloge maščobe in ogljikovih hidratov ne kopičijo, obenem pa se amino kisline, ki nastanejo pri obdelavi beljakovin uporabljajo za energijo, namesto za sintezo novih nadomestnih beljakovin, kar pripelje do izčrpanosti in na koncu smrti [13]. Stradanje je v zahodni družbi na srečo redek problem, je pa zato povprečen vnos hrane zahodnega človeka energetsko prekomeren.

V nadaljevanju bomo prikazali razvoj sodobne spletne strani za sledenje

dieti oziroma pravilni prehrani, aktivnostim in posledično lažjem vodenju bolj zdravega življenja v sodobni poplavi živil. Spletno stran smo poimenovali NomNom HopHop.

1.1 Začetki svetovnega spleta in razvoj brskalnikov

Svetovni splet (angl. WWW, World Wide Web) je porazdeljen hiperteksni sistem, ki deluje v medmrežju. Začetki spleta segajo v leto 1989, ko je Tim Berners-Lee v Evropskem središču za jedrske raziskave CERN razvil sistem, ki bi poenostavil dostop do knjižničnih informacij, ki so gostovale v strežnikih raziskovalnega središča CERN. Prve spletne strani v zgodnjih devetdesetih so vsebovale le tekstovne elemente opisane z označevalnim jezikom HTML. Za bolj pregledno branje spletnih strani sta do leta 1994 na trg prispela brskalnika Mosaic in Netscape Navigator, ki sta zakoreninila osnove grafičnih uporabniških vmesnikov vseh prihodnjih brskalnikov [15].

Microsoft je leta 1995 izdal prvo različico svojega brskalnika Internet Explorer, ki ga je leta 2002 uporabljajo kar 95% uporabnikov [30]. V letih do prevlade Internet Explorerja je potekala vojna brskalnikov, v kateri je Microsoft zmagal tudi zaradi vključevanja Internet Explorerja v svoj operacijski sistem Windows.

Zaradi visoke razširjenosti Internet Explorerja 6 in njegove slabe podpore standardov smo razvijalci spletnih strani še do leta 2010 čutili posledice, ki so omejevale razvoj in uporabo modernih prijemov. Od zgodnjih let razvoja brskalnikov in nekje do sredine prvega desetletja 21. stoletja smo se razvijalci in oblikovalci borili z nestandardnim obravnavanjem HTML kode, JavaScripta in prekrivnih predlog CSS. Danes položaj še vedno ni idealen, je pa podpora standardom in razširjenost modernih brskalnikov zadovoljiva za bolj moderen razvoj spletnih strani. Najbolj popularni štirje brskalniki leta 2011 so Firefox, Chrome, Internet Explorer in Safari. Pri tem je pomembno tudi, da je uporaba problematične šeste različice Internet Explorerja, z izjemo Azije, padla pod 4% [23].

Povprečna moderna spletna stran je tako danes zgrajena z uporabo jezika HTML, prekrivnih predlog CSS in skriptnega jezika JavaScript. Za razvoj jezika HTML, spletnih standardov in prekrivnih predlog CSS skrbi inštitut W3C, kar moderni brskalniki spoštujejo in tako omogočajo lažji razvoj spletnih strani z manj oziranja na specifike uporabnikovega brskalnika.

1.2 Statične in dinamične spletne strani

V začetku devetdesetih je bila pogosta uporaba statičnih spletnih strani, kjer je bilo potrebno za spremembo prikazane vsebine spremeniti izvorno datoteko s HTML kodo, kjer je bila največkrat shranjena tako vsebino kot opis HTML strukture. Spletni strežniki so brskalnikom preprosto prenašali vsebino teh statičnih strani in povezane grafične elemente.

Že leta 1993 pa se je začel uporabljati skupni prehodni vmesnik CGI, ki je razvijalcem spletnih strani omogočil pisanje na strežniku izvajajočih skript, ki so omogočile dinamično spreminjanje vsebine spletnih strani. V drugi polovici devetdesetih so bili razviti še drugi sistemi za dinamično programiranje spletnih strani, kot so Microsoftov ASP, Python, Java servleti in PHP. Z uporabo relacijskih podatkovnih baz je svetovni splet tako postal aktivna in dinamična infrastruktura sodobne družbe [19].

Programske jezike in okolja za razvoj dinamičnih spletnih strani se od splavitve svetovnega spleta naprej aktivno razvija in izbira pravega orodja je največkrat odvisna od zrelosti jezika, usposobljenosti programerjev, velikosti zastavljenega projekta, standardov razvojne hiše in potrebnih začetnih finančnih vložkov.

1.3 Odzivnost in funkcionalnost brskalnikov

Razvoj spletnih strani in uporaba določenih tehnik sta največkrat omejena z zmožnostmi brskalnikov in nameščenih vtičnikov. Za izris in funkcionalnost osnovnih komponent grafičnega vmesnika, kot so gumbi, polja za vnos in drsniki, poskrbi že brskalnik sam. Vendar spletne strani izvirajo iz tekstovnega okolja in za prikaz video vsebine, bogatih animacij, glasbe in za boljšo odzivnost strani, je potrebno uporabiti ustrezno podprta programska orodja in vtičnike. Zaradi varnosti imajo ta orodja precej omejen nabor privilegijev za izvajanje in brskalnik si lahko predstavljamo kot peskovnik (angl. sandbox), ki omejuje njihovo delovanje. Zaradi zgodnjega zavedanja nevarnosti izvajanja kode na napravah uporabnikov, je do danes znanih in odpravljenih že mnogo varnostnih lukenj, ki izrabljajo razne razširjene funkcionalnosti brskalnikov.

Brskalniki že od zgodnjih različic omogočajo razvijalcem izvajanje skriptnih jezikov na računalnikih uporabnikov. Najbolj pogosto uporabljen in podprt je skriptni jezik JavaScript, s pomočjo katerega lahko razvijalci popestrijo grafični vmesnik, uporabljajo animacije, realizirajo napredne menije, spreminjajo elemente drevesa DOM, nadzorujejo vtičnike in še marsikaj. Na voljo je tudi precej JavaScript ogrodij in knjižnic, ki še dodatno pospešijo razvoj in

nudijo uporabnikom bolj poznane vmesnike tudi na spletnih straneh različnih razvijalcev.

Leta 1996 je podjetje Macromedia splovilo platformo Flash, ki je omogočila dodajanje animacije, videa in bolj interaktivnih vmesnikov na spletne strani. Pogosto se uporablja tudi za spletne igre in animacije, ki tečejo v brskalnikih uporabnikov. Težave pri uporabi Flasha so potreba po uporabi vtičnika za izvajanje Flash vsebin v uporabnikovem brskalniku, strojna zahtevnost zaradi vektorske narave programa in zaprtost kode. Kljub temu da je Flash omogočil rast strani kot je Youtube.com, je priporočljivo čimbolj omejiti uporabo Flash vsebin. Precej mobilnih naprav namreč Flasha ne podpira, prav tako pa pajki spletnih iskalnikov Flash vsebin ne indeksirajo.

Najbolj pa je odzivnost spletnih strani leta 2006 popestrila uporaba tehnik AJAX. Uporaba tehnik AJAX omogoča asinhrono komunikacijo med strežnikom in brskalnikom brez potrebe po spremembi videza in obnašanja spletne strani. Tako so razvijalci z uporabo tehnik AJAX uvedli novosti kot so posodabljanje le manjših delov spletne strani, razvili bogate spletne aplikacije kot je Google maps in omogočili razvoj živih namigov za iskanje [29].

Uporabno orodje je tudi javanski aplet (angl. Java applet), ki omogoča razvijalcem izvajanje omejenih opravil v programskem jeziku Java na uporabnikovem računalniku. Se pa javanski apleti danes ne uporabljajo tako množično kot so si razvijalci Jave zamislili in tako na spletu najdemo veliko več dinamičnih vsebin, narejenih za uporabo Flasha in JavaScripta.

V prihajajočih letih bo to področje precej spremenila tudi uporaba standarda HTML 5.0, ki bo med drugim odstranil potrebo po vtičnikih za predvajanje video vsebin na spletnih straneh.

1.4 Obdobje spleta 2.0 in prihodnost spleta

Pojem splet 2.0 je najbolj povezan s spletnimi aplikacijami, ki podpirajo razširjeno uporabnost, izmenjavo informacij in sodelovanje na svetovnem spletu. Sam pojem ne pomeni nove različice spleta, ampak razvoj interneta kot platforme za aplikacije. Očitna primera razlike med poslovnimi modeli starih spletnih aplikacij in modernih sta Encyclopeadia Britannica Online in Wikipedia. V primeru prve je vso vsebino izdelalo podjetje in obiskovalcem ni nudilo nobenih možnosti sodelovanja, medtem ko je Wikipedia plod sodelovanja uporabnikov iz celega sveta [8]. Mnoge spletne aplikacije so po letu 2000 prerasle v platforme, ki svoje člane spodbujajo k ustvarjanju virtualnih skupnosti. Uporabnik tako ni več le pasiven popotnik, ampak aktivno sodeluje pri ustvarjanju

vsebine, deljenju povezav in večanju priljubljenosti storitev. Primeri takšnih aplikacij so družabna omrežja kot je Facebook, blogi, vikiji, Youtube, spletni forumi in tudi naša spletna aplikacija NomNom HopHop [28]. Leta 2006 je tako za osebo leta revije TIME izbrala *tebe* (angl. you), torej maso uporabnikov, ki sodeluje in ustvarja vsebine na družbenih spletnih straneh, blogih in straneh za deljenje multimedijskih vsebin.

Del sestavine spleta 2.0 so obogatene internetne aplikacije (angl. rich internet applications), ki uporabniške vmesnike spletnih strani približajo tistim iz sveta standardnih aplikacij, predvsem z uporabo tehnik AJAX [5]. Prav tako je pomembna storitveno usmerjena arhitektura spletnih aplikacij, ki različnim spletnim aplikacijam omogoči sodelovanje in medsebojno uporabo funkcionalnosti ter ustvarjanje bolj bogate izkušnje za uporabnike. Primer teh so viri (angl. feeds), RSS in spletne storitve. Pomemben del izkušnje spleta 2.0 so tudi uporaba značk, oblakov značk, signaliziranje spremembe vsebine preko tehnologij kot so RSS, iskanje s ključnimi besedami in povezovanje uporabnikov ter strani s socialnimi zaznamki. S spletom 2.0 je postalo zelo pomembno tudi zbiranje in povezovanje ogromnih zbirk podatkov in analiziranje pomena teh podatkov.

Mnogi v prihodnosti spleta vidijo pomembno vlogo semantičnega spleta (angl. semantic web), imenovanega tudi splet 3.0, ki ga lahko opišemo kot človeško proizvedeni splet podatkov, ki omogoča napravam razumeti semantiko ali pomen informacij na svetovnem spletu. Semantični splet razširi mrežo človeku prijaznih in berljivih spletnih strani z meta podatki, ki jih lahko berejo naprave oziroma algoritmi. Z uporabo in rastjo semantičnega spleta bi tako izboljšali iskalne algoritme, omogočili boljše povezovanje podatkov med podjetji in celo proizvajanje novih informacij s strani algoritmov in naprav samih.

V primeru naše aplikacije je na mestu uporaba ogrodja za opis virov RDF, ki je model za opis meta podatkov. Za vse vnose živil bi lahko aplikacija ponujala kopico meta podatkov, kar bi lahko izboljšalo vidljivost spletne strani v iskalnikih in razširilo uporabnost spletne strani kot podatkovno storitev za druge aplikacije. Menimo, da semantični splet še ni dovolj uporaben, čeprav obstajajo že iskalniki, ki delno upoštevajo meta podatke, kot so RDF vnosi. Zato smo se odločili pustiti čimbolj odprto pot h kasnejši implementaciji z uporabo razvojnih načel kot so MVC in predložni stroj (angl. template engine), za čas, ko bo semantični splet bolj dozorel.

1.5 Pregled naloge

V uvodnem poglavju smo predstavili razvoj svetovnega spleta, trenutno stanje na trgu brskalnikov in pogosta orodja in tehnologije, po katerih posegamo razvijalci spletnih strani.

V drugem delu predstavimo uporabljene tehnologije in orodja, ki smo jih uporabili pri razvoju spletne strani ter izpostavimo argumente za njihovo ustreznost pri našem projektu.

Tretje poglavje obravnava razvojne paradigme, ki smo se jih držali tekom razvoja. Predstavimo nekaj najpomembnejših paradigem kot so uporaba arhitekture MVC, varnost pri razvoju spletnih aplikacij in objektno usmerjeno programiranje.

V četrtem poglavju predstavitev naše spletne aplikacije pričnemo z opisom podatkovnega modela in nadaljujemo z opisom pomembnejših sistemov kot sta sistem uporabnikov in sistem za kontrolo dostopa. Na primeru sistema za registracijo novih uporabnikov si ogledamo tipičen tok razvoja gradnikov naše spletne aplikacije. Predstavimo še povezavo z družabnim omrežjem Facebook, vmesnik za administracijo in za konec pokažemo še nekaj funkcionalnosti grafičnega vmesnika.

Zaključno poglavje je namenjeno sklepom in predstavitvi nekaterih možnosti za nadaljnji razvoj.

Poglavje 2

Uporabljene tehnologije in orodja

2.1 Tehnologije

2.1.1 PHP

PHP je odprto kodni skriptni programski jezik, namenjen izvajanju na strežnikih za izdelavo dinamičnih spletnih strani [25]. Večina spletnih strežnikov tolmači PHP kodo s pomočjo namenskega modula za obdelavo PHP ukazov, ki v najbolj osnovnem primeru proizvede dokument spletne strani.

Razvoj jezika PHP je leta 1994 začel Rasmus Lerdorf, ker je hotel narediti svojo domačo spletno stran dinamično, slediti obiskom in prikazovati svoj življenjepis. Od tu izvira tudi akronim PHP, ki je na začetku pomenil *Personal Home Page Tools*, torej orodja za osebno spletno stran. Dandanes je PHP rekurzivni akronim za *PHP: Hypertext Preprocessor*, zanj pa skrbi skupina *The PHP Group* [33].

Jezik PHP ima aktivno skupnost razvijalcev in je od svojih skromnih začetkov kot skupek skript zrasel v splošno namenski programski jezik, ki omogoča tako postopkovni kot objektni razvoj programov. Od različice PHP 4 dalje PHP razčlenjevalnik prevede ukaze v zlogovno kodo, ki jo nato obdela Zend Engine. To precej izboljša zmogljivost v primerjavi s prednikom, ki je ukaze le tolmačil.

PHP smo za razvoj izbrali, ker je živ jezik, podprt s strani vseh pomembnih spletnih strežnikov, na njem uspešno temelji 75% vseh spletnih strani in podpira paradigmo objektnega razvoja. Prav tako preko odprto kodnih knjižnic in modulov podpira ogromno orodij in se ponaša z eno izmed boljših doku-

mentacij programskih jezikov.

2.1.2 MySQL

MySQL je sistem za upravljanje s podatkovnimi bazami. Je odprto kodna implementacija relacijske podatkovne baze, ki za delo s podatki uporablja jezik SQL [16]. Trenutno je v lasti podjetja Oracle, ki pa je zaenkrat obdržalo možnost uporabe MySQL pod licencno GPLv2.

MySQL se je od različice 3.23 izdane junija 2000 razvil v sodoben SUPB, ki podpira tako transakcije, gnezdene poizvedbe, B-indekse, referenčne omejitve integritete itd.

Za MySQL je bilo razvito tudi precej odprto kodnih orodij. Pri razvoju smo uporabljal MySQL Workbench za podatkovno modeliranje in phpMyAdmin kot grafični vmesnik za administracijo podatkovne baze. Pomemben dejavnik za uporabo MySQL je tudi dejstvo, da je na voljo ogromno vmesnikov za različne programske jezike, med njimi tudi za PHP.

MySQL je po študijah ustrezen za manjše projekte z omejenimi začetnimi finančnimi sredstvi, a se dobro obnese tudi v primeru rasti s prehodom na poslovno licenciranje. Primeri uspešnih projektov, ki uporabljajo MySQL so na primer Facebook, Youtube in Wikipedia [32].

Za MySQL je značilna tudi praktična neodvisnost od operacijskih sistemov, kar za konkurenčne produkte kot so na primer DB2, SQL Server in Oracle DB ne moremo trditi.

2.1.3 JavaScript in AJAX

JavaScript je dinamičen skriptni programski jezik s šibkimi podatkovnimi tipi, ki temelji na prototipih. Omogoča objekten, imperativen in funkcionalen način programiranja. Razvit je bil z namenom ustvarjanja interaktivnih spletnih strani [7]. JavaScript se večinoma uporablja na strani odjemalca, kjer se izvaja v spletnem brskalniku.

V devetdesetih je JavaScript veljal za nezanesljivega, saj je bila podpora v različnih spletnih brskalnikih precej nestandardna in jezik je bil velikokrat uporabljen nestrokovno in nezanesljivo. Veliko pa je na področju JavaScripta spremenil prihod in razširjena uporaba metod AJAX ter veliko boljša podpora sodobnih brskalnikov. Dandanes je JavaScript za spletne razvijalce standard za razvoj vmesnikov na strani odjemalcev. Napisanih je tudi več knjižnic in razvojnih ogrodij, kot na primer jQuery, ki omogoča hitrejši in bolj zanesljiv razvoj JavaScript skript na različnih brskalnikih.

AJAX pomeni asinhroni JavaScript in XML in je skupina medsebojno povezanih spletnih razvojnih tehnik, uporabljenih za ustvarjanje interaktivnih spletnih aplikacij. Z AJAX-om si lahko spletne aplikacije asinhrono izmenjujejo podatke s strežnikom v ozadju, ne da bi bilo potrebno na odjemalčevi strani ponovno naložiti celotno stran [5].

Podatki se v sodobnih brskalnikih prenašajo s pomočjo objektov XMLHttpRequest. Kljub imenu za programiranje AJAX vsebin JavaScript ni obvezen, ampak se lahko uporablja na primer tudi VBScript ali drugi skriptni jeziki. Prav tako za izmenjavo podatkov ni obvezen XML. V naši aplikaciji uporabljamo notacijo JSON, ki je vedno bolj pogosto uporabljena alternativa in jo uporablja tudi Facebook, s katerim v aplikaciji komuniciramo na več mestih.

Pristop AJAX naredi spletno izkušnjo za uporabnika prijaznejšo, bolj odzivno in hitrejšo. Pomembno je tudi, da se z uporabo tehnik AJAX med strežnikom in odjemalcem prenaša precej manjša količina podatkov, kar pomembno razbremeni mrežno infrastrukturo. Takšna razbremenitev pride še posebno do izraza v modernih socialno povezanih spletnih straneh, kjer uporabljamo povezovanje podatkov iz veliko različnih virov.

Seveda pa ima uporaba tehnik AJAX tudi slabosti na katere moramo biti pri razvoju pozorni. Ker obstajajo tudi brskalniki, ki ne podpirajo skriptnih jezikov in delovanje na vseh brskalnikih ni poenoteno, je potrebno vložiti več dela v programiranje. Potrebno je zagotoviti tudi čim več alternativnih dostopov do funkcionalnosti, ki smo jih rešili s tehnikami AJAX. Prav tako spletne strani, ki jih dinamično proizvedemo s poizvedbami AJAX, brskalniki ne shranijo avtomatično v svoj pogon za zgodovino, kar lahko povzroči nevšečnosti pri delu z navigacijo po zgodovini [9]. Problematično je tudi shranjevanje strani, ki so dinamično proizvedene s poizvedbami AJAX, med priljubljene. Kljub temu pa je uporaba tehnik AJAX bistveno izboljšala izkušnjo uporabnikov na svetovnem spletu.

2.1.4 HTML, XHTML in CSS

HTML v prevodu pomeni jezik za označevanje nadbesedila in je označevalni jezik za izdelavo spletnih strani. HTML opisuje strukturo in semantični pomen delov dokumenta. Omogoča tudi vključevanje skript, slik in drugih objektov.

Za očeta jezika HTML velja Tim Berners-Lee, ki je v zgodnjih devetdesetih letih 20. stoletja pri CERN-u jezik razvil na osnovi principov SGML. Zadnja različica jezika HTML 4.01 je bila izdana leta 1999. Za standardiziran razvoj jezika skrbi inštitut W3C.

Na tem mestu velja omeniti tudi standard XHTML, ki ga je inštitut W3C

želel uvesti kot naslednika HTML v prvem desetletju 21. stoletja. XHTML ima enak namen kot HTML, vendar je usklajen s sintakso XML in prilagojen strožji podmnožici SGML-ja. Pogoni za prikaz XHTML in HTML v ozadju zgradijo drevesno strukturo objektov imenovano DOM. Razlika med XHTML in HTML je v načinu označevanju za dosego želenega stanja drevesa DOM. XHTML dokument mora biti sintaktično brezhiben, sicer se celotna stran ne prikaže, a nudi precej preprosto razčlenjevanje z razčlenjevalniki XML.

Leta 2000 je postal XHTML 1.0 priporočilo inštituta W3C, XHTML 1.1 pa leto pozneje. Večina spletnih strani ima danes tako kot deklaracijo vrste dokumenta definiran XHTML 1.0, vendar se še vedno obravnavajo kot *text/html* in tako brskalniki ne uporabljajo razčlenjevalnikov XML [22]. Glavna razloga za slab sprejem XHTML sta bila slaba podpora brskalnika Internet Explorer do vključno različice 8 in drakonska obravnava sintaktičnih napak.

Leta 2009 je inštitut W3C je opustil razvoj XHTML 2.0 in se osredotočil na razvoj standarda HTML5, ki bo nasledil tako HMTL 4.01, kot XHTML 1.1 [12]. Pomembno je razumeti, da se bo do leta 2022 začel uporabljati HMTL5, ki bo prinesel nove semantične elemente, podporo video vsebinam vključenim v dokumente HTML in še mnogo več. Tokrat se je inštitut W3C odločil uvesti nov standard postopoma in bolj tesno sodelovati z razvijalci glavnih brskalnikov na trgu.

CSS je jezik predlog, ki določajo predstavitveno semantiko spletnih strani. S prekrivnimi predlogami CSS določamo na primer velikost pisave, barve in postavitev elementov. Od leta 1998 do 2011 je inštitut W3C kot standard predlagal različico CSS 2. Leta 2011 pa je po dolgem razvoju priporočil različico CSS 2.1, vendar je že od leta 1999 v razvoju tudi različica CSS 3, ki se razvija v posameznih modulih [3]. Glavni razlog za počasen razvoj jezika CSS je kompleksna, počasna in nestandardna podpora novih različic v brskalnikih.

CSS se lahko vključi v spletni dokument kot povezavo do zunanje datoteke, kar omogoča konsistenten prikaz posameznih delov strani, lažji razvoj in zmanjša količino potrebnega mrežnega prometa za vsak obisk strani. CSS je pomemben tudi, ker loči vsebino od oblikovnega vidika, kar med drugim omogoča boljšo spletno izkušnjo uporabnikom s posebnimi potrebami.

2.2 Knjižnice

2.2.1 Zakaj knjižnice?

Spletne strani so zasnovane odprto in vsakdo lahko zelo preprosto preveri izvorno kodo dokumentov HMTL, JavaScripta, prekrivnih predlog CSS in slik.

Pomembni dejavniki razvoja spletnih strani so medsebojno učenje spletne skupnosti na projektih drugih in skupno izboljševanje tehnik pisanja strani, ki izvira iz proste narave komuniciranja na svetovnem spletu.

Razširjenost uporabe svetovnega spleta in proste komunikacije omogoča sodelovanje skupin kot še nikoli v zgodovini človeštva. Tako so bile na primer razvite odprto kodne rešitve, ki jih izboljšuje globalna skupnost in so dostopne posameznikom, ki vedo kakšne rešitve potrebujejo za reševanje določenih problemov.

Z uporabo preizkušenih in zanesljivih rešitev se lahko izognemo porodnim varnostim problemov, prihranimo ogromno časa pri razvoju in testiranju ter z lastnimi prispevki tudi pripomoremo k izboljšanju rešitve za druge uporabnike.

2.2.2 Podatkovne baze in ADOdb

Pri razvoju spletne aplikacije za podatke skrbi SUPB, s katerim komuniciramo s pomočjo ustreznih funkcij in knjižnic napisanih za PHP. PHP ima za vsak SUPB kup namenskih funkcij, zato je dobro uporabiti plast abstrakcije med PHP funkcijami in aplikacijo, da zagotovimo čim lažji prehod na drug SUPB. Bolj pomembno kot podpora zamenjavi SUPB pa je iz programerjevega pogleda uniformnost dostopa do SUPB. V ta namen uporabimo plast dostopa do SUPB, ki združi pogoste postopke v posamezne klice, vpelje sistem izjem, varnosti in omogoči objekten pristop komuniciranja s SUPB.

V ta namen imajo večje spletne aplikacije ločeno podatkovno plast, ki je po možnosti čim bolj abstraktna, kar omogoča tako lažji razvoj za več SUPB-jev kot programerju bolj poznan vmesnik in način razvoja. Podoben pomen ima v svetu Jave na primer JDBC.

Na spletu je na voljo kar nekaj knjižnic za abstrakcijo dostopa do SUPB, kot so na primer dbFacile, PDO, ADOdb in Pear MDB2. Za lažjo odločitev, katera rešitev najbolj ustreza projektu smo se lotili razvoja lastnega lahkega vmesnika za SUPB. S pomočjo razvoja vmesnika smo dobili vpogled v funkcionalnosti, ki jih od takšnega sistema potrebujemo in pričakovani padec zmogljivosti, zaradi implementacije vmesne plasti. V izpisu 2.1 je podan konstruktor in primer metode za vzpostavitev povezave s SUPB.

Po primerjavi dokumentacije, stanja razvoja, mnenj spletne skupnosti in podprtih funkcionalnosti, smo se odločili uporabiti knjižnico ADOdb, ki uporablja SQL, podpira vrsto SUPB-jev, omogoča delo z objekti in podpira izjeme. Je tudi optimizirana za hitrost in zrel produkt, ki je na trgu že od leta 2000. Z uporabo ADOdb smo si odprli tudi možnost kasnejšega razvoja lastne knjižnice z uporabo enako poimenovanih metod, kot je na primer db::Execute(). Ob

začetni splovitvi spletne strani je ponovno programiranje vseh funkcionalnosti ADOdb nesmiselno, vendar pa si še vedno pustimo odprto pot za lastno specializirano rešitev.

```
class DBInterface {
 private $User;
 ...
 private $DBServer;

 /* ... */
 public function __construct() {
  $this->USER = "USERNAME";
  $this->PASS = "PASSWORD";
  $this->DBName = "DATABASE";
  $this->DBServer = "SERVER_ADDR";
}

 /* ... */
 public function connect() {
 $db = mysql_connect($this->DBServer, $this->USER, $this->PASS)
        or die ("Cannot connect to the DB.");
 mysql_select_db($this->DBName, $db) or die("Cannot select DB.");
}
```

Izpis 2.1: Vmesnik za povezavo s SUPB.

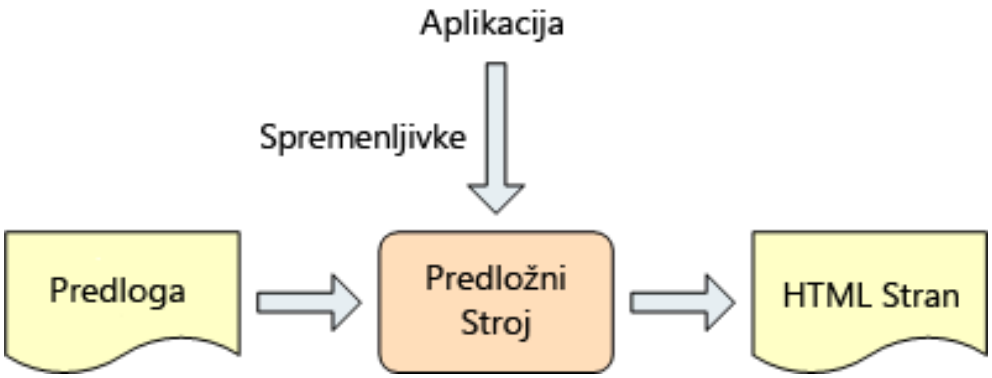
2.2.3 Predložni stroj in Smarty

Vprašanje, ki ga moramo rešiti pri vsaki večji spletni strani, je kako sestaviti HTML kodo, ki jo ponudimo uporabniku po obdelavi zahtev. Kompleksnih projektov se ne lotimo z enostavnim izpisovanjem HTML gradnikov ob izvajanju postopkovne kode, kar PHP sicer nudi, ampak uporabimo bolj napredne pristope. Tak pristop ne nudi dovolj fleksibilnosti, ne dovolj nadzora nad formatom izpisa. Še pomembneje pa je, da takšen pristop ne omogoča skupinskega razvoja, kjer za oblikovanje strani skrbi član skupine, ki ni seznanjen s programiranjem jezika PHP.

Možna rešitev, ki jo ponuja že sam jezik PHP, je vstavljanje HTML in PHP datotek v izvajajočo kodo z uporabo ustrezne družine PHP funkcij:

```
include("templates/header.php");
/* Processing */
include("templates/subpage.php");
include("templates/footer.php");
```

S takšnim pristopom izoliramo funkcionalnost gradnje HTML in omogočimo lažje deljenje dela v skupini. V takšnih predložnih datotekah obdržimo tudi vso funkcionalnost jezika PHP, kar pa je lahko za neuke razvijalce tudi težava.



Slika 2.1: Umestitev predložnega stroja v tok aplikacije.

Pristop, ki smo ga izbrali je uporaba predložnega stroja (angl. template engine). Predložni stroj (ali sistem predlog) iz zbirke predlog in podatkov iz aplikacije (slika 2.1) sestavi dokument HTML, ki ga nato predstavimo končnim uporabnikom. Kodo za lastni predložni stroj napišemo hitreje kot kodo knjižnice za abstrakcijo dostopa do SUPB. Pozornost je potrebno nameniti vstavljanju dodatnih predlog, gnezdenju, dinamičnim blokom in shranjevanju večkrat zahtevanih predlog za optimizacijo. Nevšečnost pri tem pristopu je le spoznavanje piscev predlog z lastno rešitvijo.

Zaradi razpoznavnosti, odprtosti in izpopolnjenosti sistema Smarty, smo se na koncu odločili za uporabo tega predložnega stroja. Smarty je odprto kodni predložni stroj napisan v PHP-ju in izdan pod licenco LGPL. Podpira višje nivojske predložne funkcionalnosti, kot so filtriranje, izdelavo lastnih vtičnikov, krmilne stavke in zanke.

Krmilne stavke in spremenljivke v predlogah uporabimo preprosto z uporabo sintakse jezika Smarty, prikazano v izpisu 2.2.

```
1 <ul class="form">
2 {if $RESPONSE_TRIGGERED}
3   <li><p class="successBox">{$RESPONSE_MSG}</p></li>
4 {/if}
5 {if $ERROR_TRIGGERED}
6   <li><p class="errorBox">{$ERROR_MSG}</p></li>
7 {/if}
8 <li>
```

Izpis 2.2: Primer uporabe krmilnih stavkov in spremenljivk sistema Smarty.

2.2.4 jQuery

V devetdesetih je jezik JavaScript veljal za nezanesljivega zaradi nestandardne in nezanesljive podpore jezika s strani različnih brskalnikov. To težavo in še mnogo več reši JavaScript knjižnica jQuery. jQuery je odprto kodna JavaScript knjižnica, narejena z namenom poenotenja in poenostavitve programiranja HTML strani na strani odjemalcev [2]. Prva stabilna različica je izšla leta 2006. Je najbolj popularna JavaScript knjižnica na svetu, saj jo leta 2011 uporablja kar 46% od 10.000 najbolj obiskanih strani na svetu [31]. jQuery poenostavi probleme kot so izbiranje elementov drevesa DOM, rokovanje z dogodki, rokovanje s tehnikami AJAX, izdelava animacij in navigacija po dokumentu HTML.

jQuery omogoča tudi razvoj lastnih vtičnikov in uporabo velikega števila vtičnikov objavljenih na spletu. Naša spletna stran uporablja tudi razvojno ogrodje jQuery UI, ki razvijalcem doda možnost abstrakcije nizko nivojskih animacij in uporabo visoko nivojskih, že vgrajenih, prilagodljivih gradnikov.

```
1  // Remove ingredient from list
2  $(".removeIngredient").live("click", function(event) {
3   event.preventDefault();
4
5   // Find and remove the ingredient
6   var foodID = $(this).attr("data-foodID");
7   var len = ingredients.length;
8   for(i = 0; i < len; i++) {
9    if(foodID == ingredients[i].foodID) {
10     // Remove ingredient
11     ingredients.splice(i, 1);
12     break;
13   }
14  }
15
16  $(this).parent().hide("drop", "", 400, "");
17  updateNutritionSummary();
18 });
```

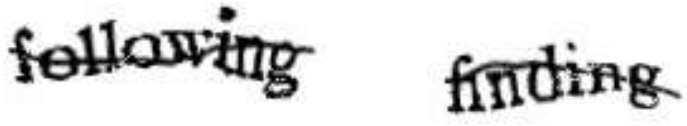
Izpis 2.3: Uporaba knjižnice jQuery za naslavljanje elementov DOM-a.

S kodo v izpisu 2.3 se v naši aplikaciji s pomočjo knjižnice jQuery in ogrodja jQuery UI odzivamo na klike uporabnika na gradnike razreda `.removeIngredient` v modulih za izdelavo receptov in mešanje hrane.

2.2.5 Druge knjižnice

Tekom razvoja naše aplikacije smo uporabili še nekaj knjižnic, ki jih bom opisal le na kratko.

Sistem reCAPTCHA uporablja koncept CAPTCHA za digitalizacijo besedil z aktivnim ščitenjem spletnih strani pred avtomatizirano oddajo obrazcev. Sistem CAPTCHA je nekakšen obrnjen Turingov test, saj računalniški program proizvaja izzive, ki jih morajo nato reševati uporabniki in računalniku pokazati, da so ljudje [4]. Največkrat je koncept CAPTCHA realiziran tako, da spletna stran za uporabnika proizvede sliko s popačenim besedilom (slika 2.2), ki ga mora uporabnik nato prepoznati, prepisati v neko polje in na strežnik poslati zahtevo s potrditvijo.



Slika 2.2: Primer testa sistema CAPTCHA.

Dobra stran uporabe široko uporabljenega sistema reCAPTCHA je prepoznavnost zahtevane akcije med uporabniki, obenem pa projekt reCAPTCHA rešuje tudi težaven problem digitalizacije besedil. Projekt pomaga zbirke besedil digitalizirati tako, da je vsak reCAPTCHA izziv sestavljen iz dveh besed, kjer je potrebno rešiti pravilno le eno. Druga beseda pa je optično prebrana slika besedila, ki jo z algoritmi optičnega prepoznavanja znakov ni bilo mogoče avtomatizirano pretvoriti v tekstovno obliko pri digitalizaciji besedil. S pomočjo množice rešitev različnih uporabnikov in uporabo statistike sliko tako spremenijo v ustrezno tekstovno obliko.

Dandanes veliko uporabnikov dostopa do spletnih strani iz mobilnih naprav, ki imajo največkrat drugačen vmesnik kot osebni računalniki. Sodobne spletne strani imajo zato možnost prilagoditve vmesnika in vsebine za mobilne platforme. Težava pa nastopi pri avtomatiziranem zaznavanju mobilnih naprav, saj je nabor mobilnih brskalnikov, operacijskih sistemov in ločljivosti zelo nehomogen. Za rešitev te težave smo posegli po odprto kodni knjižnici MobileESP, ki z majhno količino kode in nizko porabo sredstev uspešno zazna veliko količino mobilnih naprav in platform.

Pošiljanje elektronske pošte z jezikom PHP na pravilno nastavljenem strežniku je enostavno, vendar pa je še enostavneje uporabiti odprto kodno knjižnico PHPMailer, ki se v spletni skupnosti uporablja že od leta 2001. Prednost knjižnice PHPMailer je med drugim tudi popolna podpora HTML kodi v elektronskih pošti, vstavljanje datotek in dobro ravnanje z več naslovniki. Osnovni način uporabe knjižnice PHPmailer je sicer sinhrono pošiljanje, kar lahko upočasni odzivnost strani do uporabnika, vendar smo to rešili z lastnim

asinhronim pristopom pošiljanja pošte. Nalogo za pošiljanje pošte z nastavitvijo kratkih čakalnih časov pri uporabi funkcij `curl` predamo drugi zahtevi, na končanje katere uporabniku ni potrebno čakati.

Za potrebe prikaza nekaterih podatkov smo uporabili tudi knjižnico za prikaz grafov, Google Visualization API. Google ponuja robustno knjižnico s široko paleto vizualizacij in nastavitev in obenem ponuja ogromno dokumentacije in primerov. Na trgu je sicer več ponudnikov zastonjskih knjižnic za vizualizacijo podatkov, kot na primer JCharts in HighCharts, vendar se je rešitev Googla izkazala za najbolj dostopno in dokumentirano.

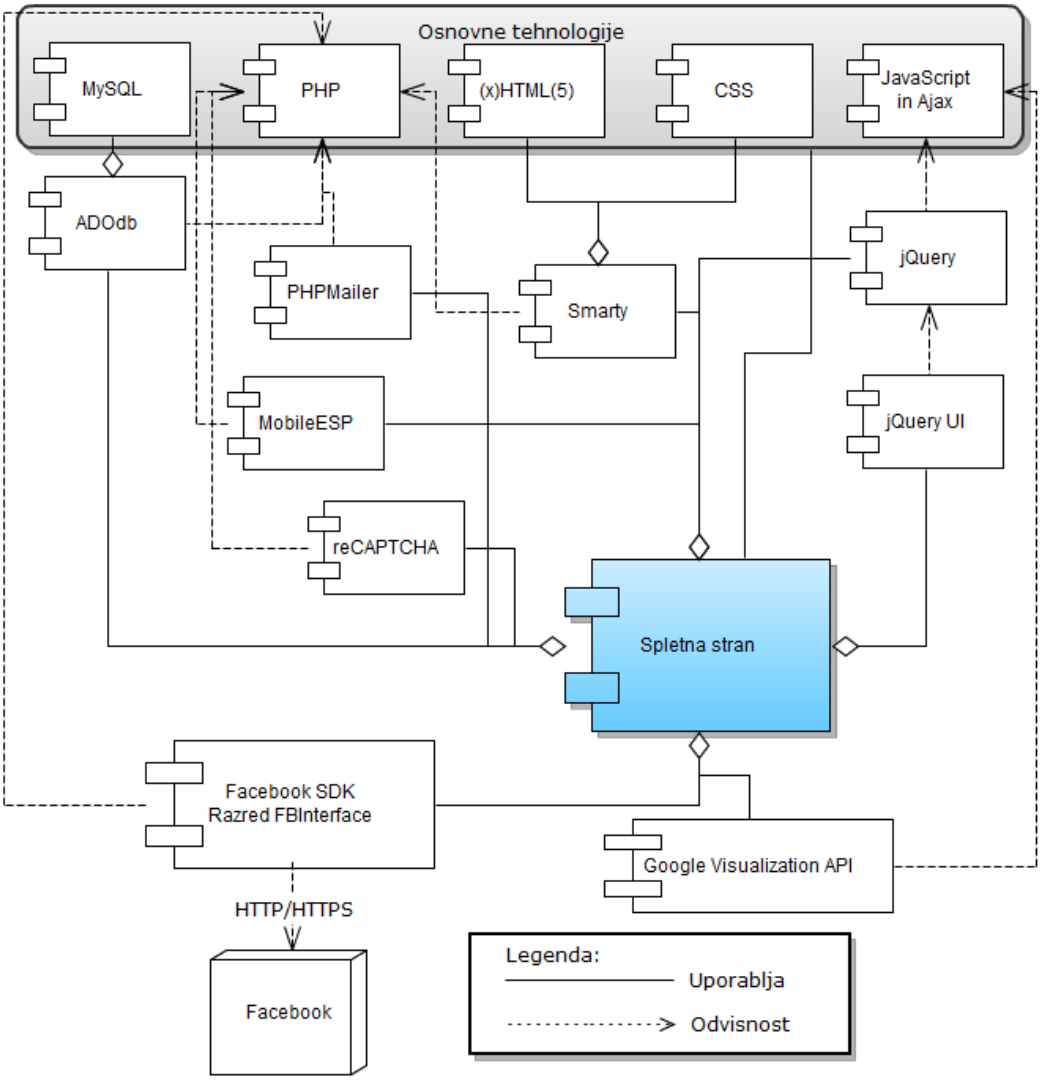
Tudi za uporabo storitev za spletne strani, ki jih ponuja družbena spletna stran Facebook, smo posegli po ustreznih knjižnicah. Za integracijo z vmesnikom Open Graph API in razvoj Facebook aplikacij z jezikom PHP, smo uporabili Facebookov PHP SDK. Ta knjižnica omogoča osnovno povezavo in komunikacijo s Facebookom. Razvili pa smo tudi lastno PHP knjižnico za delo s Facebookom, ki razširi zmogljivosti osnovnega SDK-ja in reši nekaj pogostih problemov, kot so vzdrževanje seje, dekodiranje in objavljanje v Open Graph API. Za primer v izpisu 2.4 predstavljamo metodo naše Facebook knjižnice `FBInterface`, ki vrne vrednost poljubne povezave iz vmesnika Open Graph API.

```
1  /**
2  * Fetch information on connections (Graph API Edges)
3  *
4  * @param mixed $fboid Object ID (Vertice ID)
5  * @param mixed $edgeType Connection type (Edge type)
6  * @param String $queryParams Query parameters,
7  * format: &var=val&var2=val2 ...
8  */
9  public function fetchGraphEdge(
10  $fboid, $edgeType, $queryParams = "") {
11
12  $graph_url = "https://graph.facebook.com/{$fboid}/".
13    "{$edgeType}?access_token=" . $this->getAccessToken()
14    . $queryParams;
15
16  return $this->fetchGraphData($graph_url);
17 }
```

Izpis 2.4: Metoda fetchGraphEdge knjižnice za delo s Facebookom.

Za integracijo Facebook komentarjev, gumbov *všeč mi je* (angl. like) in drugih funkcionalnosti na odjemalčevi strani smo uporabili tudi Facebook JavaScript SDK.

2.3 Povzetek



Slika 2.3: Komponentni diagram uporabljenih pripomočkov okoli spletne strani.

Za razvoj naše spletne strani smo torej uporabili precej različnih pripomočkov, tehnologij in knjižnic, kar je razvidno iz diagrama na sliki 2.3. Njihova uporaba omogoča hitrejši razvoj, vendar pa zahteva tudi poznavanje njihovega delovanja in skrb za vzdrževanje kompatibilnosti ter varnosti posameznih delov.

Poglavje 3

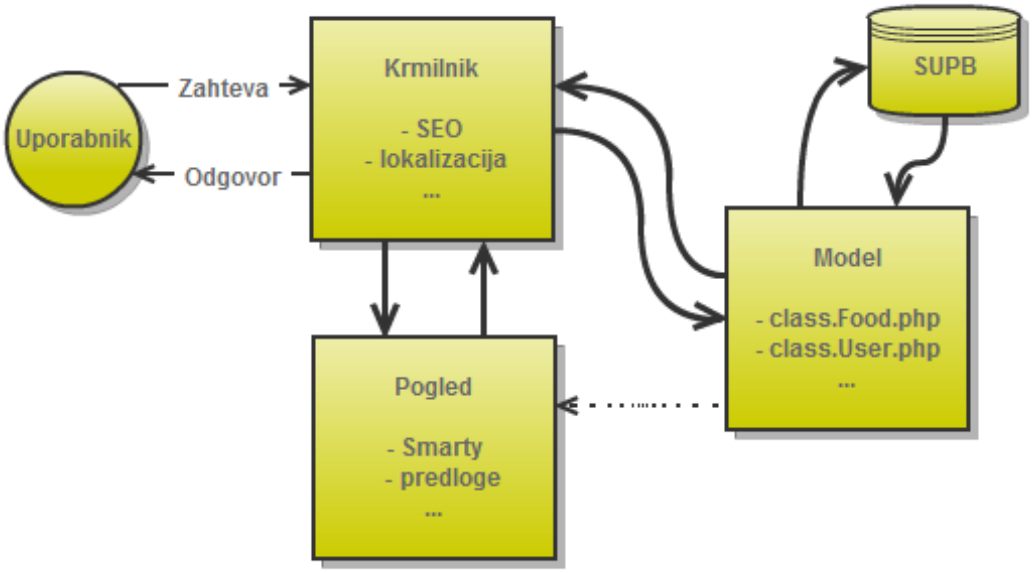
Razvojne paradigme

3.1 MVC arhitektura

Naša aplikacija temelji na znani arhitekturi MVC. Črke MVC predstavljajo v slovenščini besede model, pogled in krmilnik. Model je del aplikacije, ki skrbi za delo s podatki ter realizacijo poslovne logike in pravil [14]. Model je odgovoren tudi za podatkovno abstrakcijo, ki jo v naši aplikaciji predstavlja knjižnica ADOdb. Odziva se na povpraševanja po stanju in na napotke po spremembah stanja. Pogled spremeni model v obliko primerno za prikaz in interakcijo z uporabnikom in predstavlja predstavitveni nivo aplikacije. Krmilnik skrbi za vhod in izhod aplikacije tako, da kliče ustrezne objekte modela in komunicira s pogledom in modelom. Z uporabo arhitekture MVC lažje razdelimo delo na več članov skupine, pišemo bolj berljivo kodo ter primerno uporabimo predložni stroj za posamezne naprave.

V naši aplikaciji uporabljamo MVC pristop tako na strežniški strani v PHP-ju, kot na strani odjemalca. Arhitekturo MVC smo realizirali z objektnim pristopom programiranja PHP. Strežniški krmilnik komunicira z uporabnikom preko GET in POST zahtev (slika 3.1), nato pa z ustrezno logiko uporablja objekte modela kot so *Food*, *User* in *Page*, ki s svojimi metodami dvosmerno komunicirajo s SUPB-jem. Tako lahko na primer enak klic za preverjanje veljavnosti uporabnikovega gesla kličemo iz toka za uporabo tehnik AJAX in v primeru odgovarjanja na celotne GET in POST zahteve. Za pripravo ustreznega pogleda pa v sodelovanju s krmilnikom skrbi predložni stroj Smarty, ki lahko z uporabo različnih tematskih predlog uporabniku ponudi osebno izkušnjo, mobilnim napravam pa ponudi bolj prijazen vmesnik.

Pri spletnih straneh se na dejanja uporabnika najprej odziva brskalnik, kot razvijalci pa imamo tudi določen nadzor nad lokalnimi spremembami z



Slika 3.1: Uporabnik s sistemom komunicira preko komponent krmilnika.

uporabo JavaScripta. Za čim boljši približek arhitekturi MVC in ločenost posameznih nivojev skrbimo z ločenostjo vsebine in gradnikov, ki jih opišemo z jezikom HTML, predstavitveno plast realizirano s prekrivnimi predlogami CSS in knjižnico jQuery. Z uporabo knjižnice jQuery se lahko z metodami kot so `.bind` in `.live` odzivamo na dogodke kot so uporabnikovi klik in vnosi. Z ustrezno logiko lahko nato na primer izvajamo preverjanje veljavnosti vnesenih podatkov že v brskalniku in s spremembo gradnikov HTML, ali CSS komuniciramo z uporabnikom brez potrebe po obdelavi obrazca na strežniku.

3.2 Varnost v spletnih aplikacijah

Spletne strani in aplikacije so izpostavljene napadom na varnostne luknje 24 ur na dan. Leta 2011 so napadalci našli in izkoristili varnostne luknje za strani velikanov kot sta mysql.com in sony.com. Oba napada sta bila izvedena z uporabo enega najbolj poznanih in redno uporabljenih napadov, to je vrivanje poizvedb SQL [27]. Takšni napadi pomenijo ogromno izgubo zaupanja uporabnikov in v primeru napadov na velike strani znižajo varnost drugih spletnih strani, saj mnogo uporabnikov za več spletnih strani uporablja ista gesla. Tako je bilo na primer po napadu na mysql.com javno objavljenih ogromno

uporabniških gesel.

Ker so spletne strani na voljo uporabnikom iz celotnega sveta, je napadalce tudi težko izslediti. Izkušeni hekerji za izvajanje napadov uporabljajo računalnike nedolžnih ljudi, nad katerimi potuhnjeno prevzamejo nadzor s trojanskimi konji in drugimi orodji. Heker lahko tako z brisanjem sledi na okuženih računalnikih deluje skoraj neopazno in neizsledljivo. Zaradi narave interneta se zato ne moremo zanašati na kazenski pregon kot zagotovilo varnosti, ampak moramo, za čim večjo mero, te poskrbeti z ustreznimi tehničnimi ukrepi.

Nekatere varnostne luknje izkoriščajo napake v operacijskih sistemih in programih, ki poganjajo strežnike. Spet druge so posledica napadov na nepoznane ranljivosti (angl. 0-day attacks) v odprto kodnih in drugih knjižnicah, ki jih uporabljamo pri razvoju spletne strani [17]. Takšne probleme največkrat rešujemo z vestnim posodabljanjem programov in knjižnic, obenem pa se z uporabo zrelih in dobro vzdrževanih operacijskih sistemov, programov in knjižnic rešimo trivialnih napak mladih sistemov.

Za bolj problematične se v praksi izkažejo varnostne luknje v sami spletni strani, za kar pa smo odgovorni razvijalci. Tako je za razvoj spletnih strani bistveno postaviti varnost na prednostno mesto in dodobra testirati odpornost rešitev proti najbolj znanim napadom.

Obstaja več družin napadov z vrivanjem poizvedb SQL, vsi pa bazirajo na tehniki vrivanja kode, ki izkorišča varnostno luknjo na podatkovnem nivoju aplikacije [17]. Največkrat se v ta namen izrablja nepravilno filtriran vnos uporabnikov, vstavljen v poizvedbe SQL. Primer napada z vrivanjem SQL je uporaba tavtologij. Za primer podajmo preprosto poizvedbo SQL za preverjanje ustreznosti uporabniškega imena in gesla:

```
1 query = "SELECT * FROM user WHERE user = '". user ."'
2     AND pass = '" . pass "';
```

Napadalec lahko sedaj za uporabniško ime vnese preprosto tavtologijo:

```
1 user = ' or '1'='1
```

Če takšen sistem prijavo pogojuje s preprostim preverjanjem števila vrnjenih rezultatov, se bo napadalec uspešno prijavil v sistem.

V naši aplikaciji je za izvajanje poizvedb SQL zadolžen model, za komunikacijo s SUPB pa uporabljamo ADOdb. Posebno pozornost smo namenili standardiziranemu preverjanju vseh poizvedb SQL zgrajenih z uporabo uporabniških vnosov. Za varen vnos nizov smo poskrbeli s funkcijo `qstr` za ubežne znake iz knjižnice ADOdb. Primer uporabe `qstr` je prikazan v izpisu 3.1

```
sql = "INSERT INTO `" . DB_ACTIVITY_APPROVAL_TABLE . "`
    (`aa_activityID`, `aa_reviewedBy`, `aa_comment`)
       VALUES
       (
        {$activityID},
        {$reviewedBy},
        {$db->qstr($comment)}
       )";
```

Izpis 3.1: Uporaba funkcije `qstr` za obravnavanje ubežnih znakov.

Pozornost smo namenili tudi numeričnim podatkovnim tipom. Tako je tudi vsaka dinamična numerična spremenljivka v poizvedbah SQL preverjena in včasih tudi spremenjena v ustrezen numerični tip. Prav tako je pozornost treba nameniti poizvedbam, kjer lahko uporabnik izbere datume. V teh primerih je za varnost največkrat poskrbljeno s funkcijo, ki za vnos zagotovi ustrezen format.

Pri XSS napadih (angl. cross-site scripting) napadalec izkorišča ranljivosti spletnih strani z vrivanjem zlonamerne kode napisane v skriptnem jeziku, ki se izvede na strani odjemalca, z namenom kraje piškotkov in drugih zasebnih podatkov [17]. XSS napadi zaobidejo varnostne mehanizme, ki jih pod normalnimi pogoji vzdržujejo moderni brskalniki in tako napadalcu omogočijo dostop do zasebnih podatkov, ki jih za uporabnika hrani brskalnik . Ranljivosti na XSS napade so v zadnjih letih postale najbolj pogoste javno objavljene ranljivosti in nekateri raziskovalci menijo, da je na XSS napade ranljivih do 68% vseh spletnih strani [26].

Najbolj pogosta tarča XSS napadov so obrazci s polji za vnos nizov. V naši aplikaciji tako saniramo vnose nizov s primerjavo z naborom veljavnih znakov ali pa z odstranjevanjem neželenih značk s funkcijo `strip_html_tags`, del katere je prikazan v izpisu 3.2.

```
text = preg_replace(
        array(
            // Remove invisible content
            '@<head[^>]*?>.*?</head>@siu',
            '@<style[^>]*?>.*?</style>@siu',
            '@<script[^>]*?.*?</script>@siu',
            '@<object[^>]*?.*?</object>@siu',
            '@<embed[^>]*?.*?</embed>@siu',
           /* ... */
```

Izpis 3.2: S funkcijo `preg_replace` zaščitimo nize pred vrivanjem nezaželenih značk.

Uporaben prijem za večjo varnost pred XSS napadi s strani odjemalcev je tudi uporaba IP naslovov v piškotkih uporabnikov, vendar lahko ta ukrep

ustrezno podkovan napadalec zlahka zaobide. V HTML kodi se poslužujemo tudi metode kodiranja HTML entitet, s katero nekatere znake spremenimo v znake HTML entitet, ki se nato v brskalniku izpišejo kot želeni znak. Primer je uporaba entitet < in > namesto > in <.

Medtem ko XSS napad izkorišča zaupanje uporabnika v določeno spletno stran, napad s potrjevanjem spletnih zahtev ali CSRF (angl. cross-site request forgery) izkorišča zaupanje spletnih strani v uporabnikov brskalnik. Napadi CSRF izvajajo neavtorizirane ukaze v imenu uporabnika, ki mu stran zaupa brez njegovega vedenja [21]. Ranljivost izhaja iz narave uporabe brskalnikov. Ko brskalnik pošlje zahtevo na določeno stran, vedno poleg pošlje tudi vse piškotke za to stran, ne glede na to od kod ta zahteva izvira. Prav tako strežniki v večini primerov ne morejo ločiti med zahtevami, ki jih uporabnik izvede zavestno in med avtomatiziranimi. Primer preprostega napada je spletna stran ali pošta kjer napadalec za vir slike uporabi URI, ki izvede določeno akcijo na napadeni strani:

```
<img src="http://www.target-site.com/deleteArticle?articleID=ARTICLEID"
     style="display:none;" />
```

Če ima uporabnik, ki obišče to stran, veljavno sejo ali piškotke za izvajanje brisanja člankov na strani *www.target-site.com*, bo članek na tej strani izbrisan in v dnevniku dostopov na strežniku bo izpisan veljaven uporabnikov IP naslov. Proti CSRF napadom se v naši aplikaciji borimo s tehnikami kot so preverjanje vrednosti postavke *refer* v glavi zahtev, uporaba POST namesto GET zahtev za določene akcije in izmenjevanje skrivnih nizov [21].

Pri skrbi za varnost uporabniških računov je pomembna tudi pravilna uporaba zgoščevalnih funkcij v primeru varnostnega vdora v sistem. Ena izmed najbolj popularnih zgoščevalnih funkcij je 128 bitna MD5, ki pa ne velja več za varno. Obstaja namreč precej projektov, ki so objavili mavrične tabele za MD5 in se lahko uporabijo za razbijanje gesel. V naši aplikaciji uporabljamo 160 bitno zgoščevalno funkcijo SHA-1. Uporabljamo tudi koncept soljenja gesel [17], ki doda dodaten nivo kompleksnosti uporabniškim geslom, ki so mnogokrat prekratka in zato prelahka tarča napadov z grobo silo (angl. brute force). Za soljenje uporabljamo primerno dolg niz, kar onemogoči uporabo mavričnih tabel za razbijanje naših gesel. Če napadalec dobi dostop do izvorne kode, lahko še vedno uporabi napad s slovarjem [17], vendar je to nesprejemljiv scenarij.

Izpostavljenost spletnih strani omogoča napadalcem tudi napade z grobo silo, smetenjem (angl. spam) in ohromitvijo storitve (angl. denial of service, DoS). Pred napadi s smetenjem se borimo z uporabo sistema CAPTCHA, ki poskusi določiti, če je na strani odjemalca človek. Pravilna uporaba sistema

CAPTCHA pomaga tudi pred napadom z ohromitvijo storitve, saj že v zgodnjem delu obravnavanja zahteve, prekine preverjanje in tako sprosti sistemska sredstva. Za zaščito pred smetenjem je uporabljen tudi sistem potrjevanja preko elektronske pošte. Napadi z grobo silo so največkrat usmerjeni na prijavni del aplikacije in uporabljajo napade z uporabo permutacij in slovarjev. V naši aplikaciji imamo za preverjanje prijave sistem, ki po določenem številu poskusov uporabniku glede na njegov IP naslov za določen čas onemogoči prijavo, prikazan v izpisu 3.3.

```
1  if( ($clientIP = getClientIP()) === false) {
2    throw new Exception("IP Conflict.", LOGIN_IP_CONFLICT);
3  }
4
5  // If false, maximum number of login attempts
6  // for IP has been reached for CONFIG time limit
7  if(!$this->checkLoginIP($clientIP, $db)) {
8    throw new Exception("Too many loggin attempts.",
9                LOGIN_TOO_MANY_ATTEMPTS);
10   return false;
11 }
```

Izpis 3.3: Z beleženjem IP naslovov neuspelih prijav omogočimo zaščito pred napadi z grobo silo.

Takšen pristop naredi napade z grobo silo pri omejenem številu IP naslovov neučinkovite.

Pomemben vidik varnosti je tudi skrivanje podrobnosti o napakah in izjemah. Tako v naši aplikaciji uporabljamo lasten sistem izjem in nikakor ne izpisujemo podrobnosti napak neposredno uporabniku.

3.3 Krajevna prilagoditev

Krajevna prilagoditev (ali lokalizacija) ni le prirejanje ustreznih prevodov in uporaba različnih pravopisnih pravil, ampak tudi primerna uporaba datumov, zapisov in prilagajanje krajevnemu trgu.

Pri razvoju aplikacij se je dobro že pri načrtovanju odločiti za ciljni trg. Če si želimo omogočiti najboljšo možnost rasti, sploh pri razvijanju za majhen trg kot je Slovenija, je dobro že vnaprej vključiti možnost prilagoditve aplikacije za različne jezike in kulture. Za podporo različnih trgov smo se pri naši aplikaciji odločili za strategijo postavitve različnih samostojnih in lokaliziranih različic aplikacije. Naša aplikacija je namenjena tesnemu sodelovanju uporabnikov pri oblikovanju zbirke živil, prehrambenih navad in motiviranju. Različne države in trgi imajo tudi precej nehomogene prehrambene navade, blagovne znamke

in možnosti dostopa do pridelkov, zato se zdijo lokalizirane postavitve bolj optimalna rešitev kot globalna postavitev. Vsaka krajevno prilagojena postavitev bo tako gradila lastno bazo živil, združevala uporabnike iste kulture in obenem omogočila sodelovanje tudi uporabnikom, ki ne poznajo angleškega ali slovenskega jezika.

Seveda si je bilo potrebno za realizacijo takšne strategije zastaviti smernice, ki omogočajo enostavne krajevne prilagoditve. Vso kodo in komentarje smo spisali v angleščini, medtem ko smo pripravo strani za slovenski trg prepustili sistemu krajevne prilagoditve. Tako za vse nize, ki jih izpisuje aplikacija uporabljamo lastno knjižnico, ki črpa vire za prevode iz ustreznih datotek. Ob nalaganju strani aplikacija izbere ustrezno datoteko z viri, pri tem pa uporablja šifre ISO 639-1 [24], ki zagotavljajo standardizirano sklicevanje na kode ustreznih lokalizacij.

Naša spletna stran tako pri obdelavi zahtevkov naloži vse potrebne prevode za zahtevane skupine in jih servira predložnemu stroju. V primeru vnosa hrane vsebuje datoteka s sredstvi za prevajanje na primer tudi naslednje prevode:

```
$_['food']['cf_current_image'] = "Trenutna slika:";
$_['food']['cf_delete_image'] = "Odstrani sliko?";
$_['food']['cf_listed_public'] = "Hrana je objavljena javno.";
```

Pisec predloge za vnos hrane bo potem te vnose uporabil, kot je prikazano v izpisu 3.4, v svojih predlogah z uporabo predpone L in ustrezne skupine, v tem primeru L_FOOD_.

```
{if 'approved' == $FOOD_PUBLIC_MODE}
  <div class="infoBox">{$L_FOOD_cf_listed_public}</div>
  <input type="hidden" name="food_public" value="1" />
{else if 'rejected' == $FOOD_PUBLIC_MODE}
  <div class="errorBox">{$L_FOOD_cf_listing_rejected}</div>
  <input type="hidden" name="food_public" value="1" />
{else}
```

Izpis 3.4: Primer uporabe oznak za lokalizacijo.

Večjezičnost je zelo pomembna tudi pri optimizaciji za iskalnike, zato naša spletna stran uporablja tudi sistem datotek s sredstvi za krajevno prilagojeno optimizacijo za iskalnike. Več o tem v razdelku o optimizaciji za iskalnike.

3.4 Optimizacija za iskalnike

Optimizacija spletnih strani za iskalnike je postopek izboljševanja vidljivost spletne strani v iskalnikih prek naravnih ali algoritemskih iskalnih rezultatov. Spletni iskalniki imajo sicer svoja lastna pravila za razvrščanje zadetkov,

vendar je za večino trgov najbolj pomembna optimizacija za iskalnik Google. Veliko pravil za razvrščanje za določene spletne iskalnike je objavljenih javno, nekatera pa so strogo varovana tajnost [11]. Vodilni spletni iskalniki, kot so Google, Bing in Yahoo!, uporabljajo spletne pajke, da odkrijejo spletne strani primerne za izpis rezultatov. Zato je zelo pomembno, da ima spletna stran dobro zasnovano navigacijo, ki jo lahko berejo tudi pajki, ki ne izvajajo skript. SEO tehnike se ločijo na dve kategoriji: tehnike, ki jih priporočajo spletni iskalniki ter tehnike, ki jih spletni iskalniki ne odobravajo in jih poskušajo zmanjševati. Tehnike v skladu s pravili tako imenujemo bela optimizacija (angl. white hat SEO), tiste ki pravila kršijo pa črna optimizacija (angl. black hat SEO) [1].

V naši aplikaciji uporabljamo izključno belo optimizacijo. Na podstraneh izvajamo krajevno prilagojeno optimizacijo SEO nad naslovi, z uporabo meta značke `title`. Prav tako optimiziramo tudi vsebino meta značk `description` in `keywords`. Uporabljamo tudi optimizirane URL naslove; tako na primer povezava do funkcionalnosti mešanja hrane kaže na http://ime-domene.si/-mesalec-hrane. Za uporabo takšne optimizacije uporabljamo datoteko z viri za krajevno prilagajanje.

```
1 // Description
2 $_['g']['seo_desc_food_mixer'] = "Ustvari hrano z mesanjem sestavin in ↩
      poglej preracunano vrednost.";
3 // Keywords
4 $_['g']['seo_keywords_food_mixer'] = "Priprava hrane, hranilne vrednosti,↩
       kalorije v hrani";
5 // Title
6 $_['g']['seo_title_food_mixer'] = "Mesanje hrane";
7 // URL
8 $_['g']['seo_food_mixer'] = "mesalec-hrane";
```

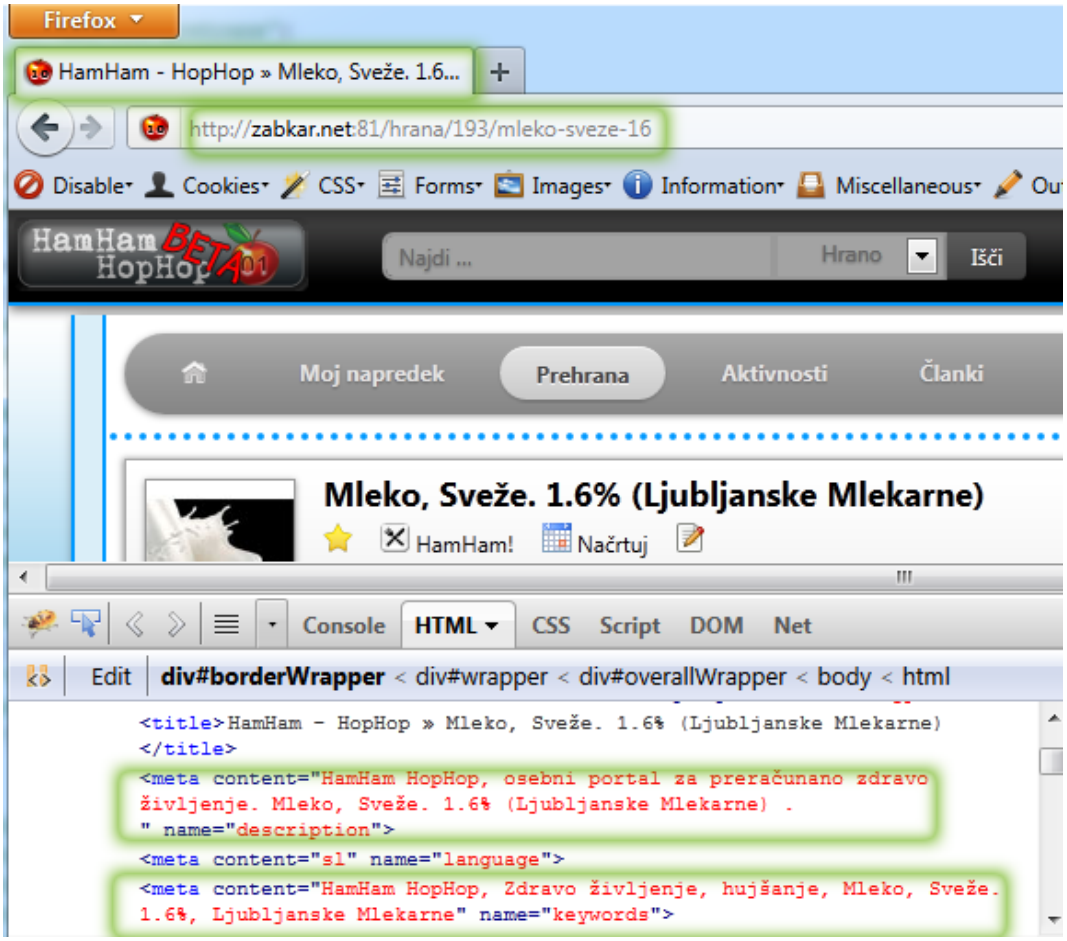
Izpis 3.5: Primer vnosov za optimizacijo SEO.

S pomočjo vnosov prikazanih v izpisu 3.5 v datoteki z viri za krajevno prilagajanje nato proizvedemo optimizirane strani, kot je prikazano na sliki 3.2.

Pri optimizaciji za iskalnike uporabljamo še mnogo drugih tehnik, kot so opisi slikovnega materiala, semantično pravilna uporaba gradnikov HMTL in tako naprej.

3.5 Objektno usmerjen razvoj s PHP-jem

Objektno programiranje je pri klasičnih programskih jezikih že dolgo razširjeno in omogoča lažji razvoj bolj kompleksnih aplikacij. Jezik PHP izvira iz iskanja



Slika 3.2: Od zgoraj navzdol si sledijo štiri tehnike optimizacije za iskalnike; optimizacija naslovov, optimizacija povezav in navigacije, optimizacija opisa in optimizacija ključnih besed.

nadomestka za skupek skript in čeprav PHP že od različice 3.0 podpira objektno programiranje je to postalo standard šele z različico 5.0. Pred tem je bil najbolj pogost postopkoven razvoj, kjer so se objekti uporabljali kvečjemu kot posebna oblika polj.

Z različico 5.0 je bilo obravnavanje objektov v PHP-ju spisano popolnoma na novo, razširjeno in optimizirano za boljšo zmogljivost. V starejših različicah so se objekti obravnavali kot vrednostni tipi in kot takšni tudi prenašali med klici metod. Sodobne različice PHP-ja imajo na primer abstraktne razrede, standarden model obravnavanja izjem, privatne in zaščitene metode in standarden način deklaracije konstruktorjev in destruktorjev [18].

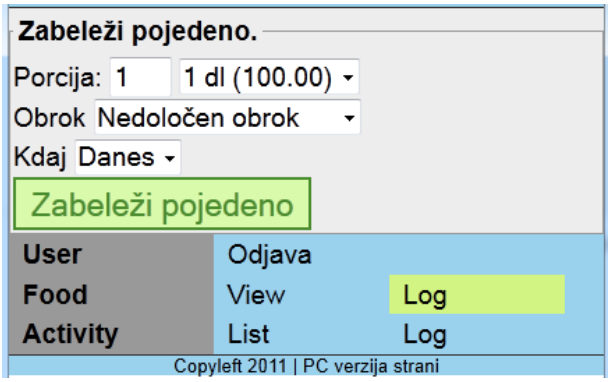
Za kompleksne spletne aplikacije kot je naša je objektno usmerjen razvoj bolj primeren od postopkovnega. Z objektnim pristopom in enkapsulacijo približamo realizacijo entitet podatkovnega modela realnemu svetu. Pri objektnem pristopu tudi zmanjšamo potrebo po ponavljanju kode in spodbujamo ponovno uporabo že spisanih metod. Je bolj primeren za skupinski razvoj, vodi do vzorcev, ki pripeljejo do bolj učinkovite kode in omogoča bolj pregledno konsolidacijo funkcionalnosti sistema. Razlog za izbiro objektnega pristopa je tudi usposobljenost razvijalcev, lažje vzdrževanje in razširljivost kode.

3.6 Uporaba predložnega stroja in obravnavanje mobilnih naprav

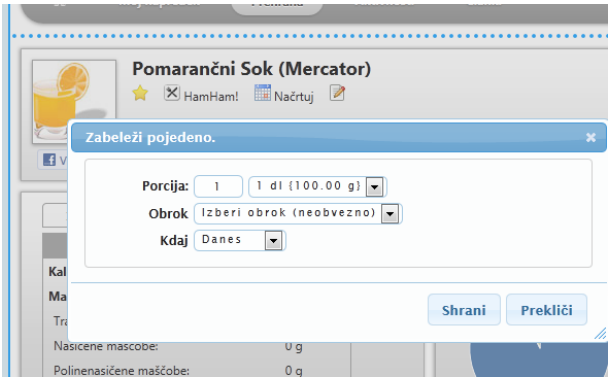
Z uporabo predložnega stroja kot je Smarty smo omogočili lažjo krajevno prilagajanje v primeru večjih kulturnih razlik. Na primer na arabskih in azijskih trgih besedilo pišejo od desne proti levi, pomen barv pa se lahko popolnoma spremeni. Z uporabo predlog si zagotovimo tudi možnost zunanjega razvoja različnih vmesnikov, obenem pa lahko uporabnikom ponudimo več vizualnih tem, ki jih lahko uporabljajo po želji in si prilagodijo spletno stran bližje svojim željam.

V sodobnem času razcvet doživljajo tudi pametne mobilne naprave za brskanje po spletu. S prihodom pametnih telefonov z operacijskimi sistemi iOS, Android, Windows Mobile 7, zasloni na dotik in močno strojno opremo, lahko uporabniki končno uporabljajo spletne strani z relativno lahkoto tudi na mobilnih napravah. Vendar pa je trg mobilnih naprav raznolik in cena prenosa podatkov je še vedno precej višja kot na domačih povezavah. Zato je dobro prirediti obraz in vmesnik spletne strani za takšne naprave in optimizirati tako vmesnik kot porabo pasovne širine [6]. S pomočjo knjižnice za zaznavanje mobilnih naprav, tem mobilnim brskalnikom že privzeto ponudimo dostop do prilagojenega mobilnega vmesnika, ki ga realiziramo z uporabo posebnih predlog s predložnim strojem. Uporabniku seveda ponudimo tudi možnost ročnega preklopa na standarden pogled.

Predloge za mobilne naprave smo na naši strani optimizirali za delo z zasloni na dotik. Močno smo zmanjšali tudi porabo pasovne širine in se izognili uporabi JavaScripta in tehnik AJAX, saj njihova uporaba na mobilnih brskalnikih še ni standardna. Na primeru vmesnika za beleženje prehrane lahko tako na sliki 3.3 vidimo mobilen vmesnik, na sliki 3.4 pa AJAX vmesnik za osebne računalnike.



Slika 3.3: Predloga za mobilne naprave.



Slika 3.4: Predloga za osebne računalnike.

3.7 Uporaba tehnik iz sveta oblikovanja igralnosti

Pri razvoju spletne aplikacije kot je naša, kjer je za uporabnike pomembna pozitivna vzpodbuda, lahko koristno uporabimo tudi tehnike, ki so ponavadi uporabljene pri razvoju iger. To so na primer vzpodbujanje raziskovanja z dosežki, napredovanje in razne nagrade. Tehnike iz sveta oblikovanja igralnosti lahko služijo kot zunanja vzpodbuda uporabnikom na poti do doseganja njihove zastavljene teže, prav tako pa pri uporabnikih zbujajo zanimanje in povečajo obiskanost strani. Tehnike kot so nabiranje točk, oblikovanje izstopanja posameznikov in nagrajevanje za pomoč drugim so tudi pomembni dejavniki za izoblikovanje skupnosti. Našo spletni stran smo že na začetku načrtovanja zastavil kot čimbolj odprto za razširitve. Tako je za implementa-

cijo različnih pristopov uporabljenih pri oblikovanju igralnosti v aplikaciji še ogromno prostora. Poglejmo pa si dve tehniki, ki ju že uporabljamo.

Sistem odklepanja dosežkov (angl. achievement unlocking) je zelo popularen pristop, ki ga uporablja veliko iger. Gre za različne akcije in naloge, ki sprožijo odklepanje nekega dosežka. Te dosežke nato uporabnik zbira in išče načine, kako jih dobiti še več. Odklepanje dosežkov smo implementirali zelo splošno, tako da je mogoče zelo preprosto in po ustaljenem načinu z uporabo objektnega pristopa dodajati nove dosežke tekom razvoja aplikacije. Razred `Achievements` ima metodo `triggerAchievement`, ki sprejme kot argument identifikator dosežka, morebitno uporabniško metodo in polje argumentov za klic te metode. Večina dosežkov ima definirano svojo metodo za preverjanje pogoja odklepa dosežka in klic te uporabniške metode izvedemo z uporabo PHP funkcije `call_user_func_array`, ki ji kot argument podamo ime uporabniške metode in polje argumentov. Poglejmo si primer implementacije dosežka za vnos živila. V datoteki za konstante določimo nekaj vnosov. Tu smo vnesli tri dosežke in sicer za vnos prvega živila, desetih in štiridesetih živil, vsakega s svojim identifikatorjem. V izpisu 3.6 lahko vidimo, da bo vsak od teh dosežkov uporabljal isto uporabniško metodo imenovano `ach_foodCreated`.

```
define("ACH_FIRST_FOOD_CREATED", '1');
define("ACH_FIRST_FOOD_CREATED_FUNC", 'ach_foodCreated');
define("ACH_10_FOOD_CREATED", '2');
define("ACH_10_FOOD_CREATED_FUNC", 'ach_foodCreated');
define("ACH_40_FOOD_CREATED", '3');
define("ACH_40_FOOD_CREATED_FUNC", 'ach_foodCreated');
```

Izpis 3.6: Definicije treh vnosov za dosežke.

Z metodo `ach_foodCreated` lahko z različnimi argumenti preverimo ali je uporabnik do sedaj vnesel eno, deset ali več različnih živil. V izpisu 3.7 je podana implementacija omenjene metode.

```
private function ach_foodCreated($minDefined = 1) {
 $sql = "SELECT COUNT(foodID) as countFood FROM "
     . DB_FOOD_TABLE .
     " WHERE food_user_id = {$this->userID}";

 $res = $this->db->Execute($sql);
 if(!$res) {
   return false;
 }
 if($res->fields['countFood'] > $minDefined - 1) {
    $res->Close();
    // Achievement unlocked
    return true;
  }
  $res->Close();
  return false;
```

```
17 | }
```

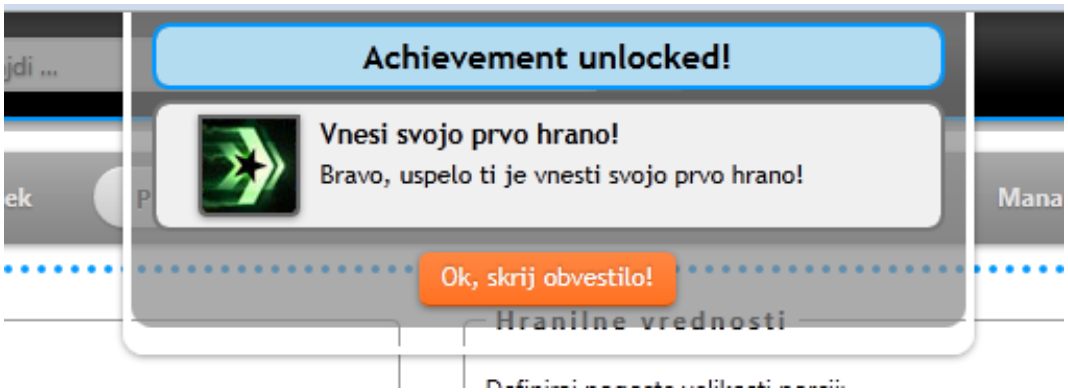
Izpis 3.7: S fleksibilno definicijo lahko eno metodo uporabimo za preverjanje različnih dosežkov.

Preverjanje pogoja za odklepanje dosežka preprosto kličemo po uspešnem vnosu živila:

```
1 | // Trigger achievements
2 | $achievements->triggerAchievement(ACH_FIRST_FOOD_CREATED,
3 |                         ACH_FIRST_FOOD_CREATED_FUNC, array(1));
4 | $achievements->triggerAchievement(ACH_10_FOOD_CREATED,
5 |                         ACH_10_FOOD_CREATED_FUNC, array(10));
6 | $achievements->triggerAchievement(ACH_40_FOOD_CREATED,
7 |                         ACH_40_FOOD_CREATED_FUNC, array(40));
```

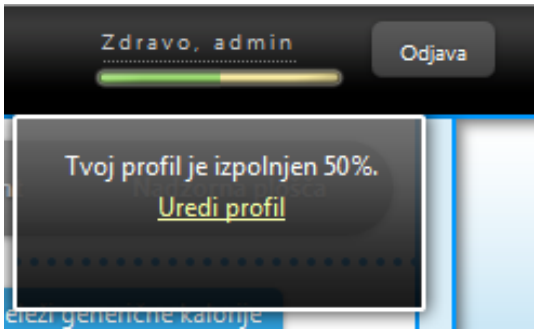
V primeru odklepa bo metoda `triggerAchievement` poskrbela za izpis obvestila v uporabnikovem brskalniku (slika 3.5) in v podatkovno bazo shranila čas odklepa dosežka.

Druga tehnika iz sveta razvoja iger, ki jo uporabljamo, je preprost indikator izpolnjenosti profila. Pri registraciji si želimo narediti izkušnjo za uporabnika čim bolj preprosto, zato odložimo možnost vnosa dodatnih osebnih podatkov na izbirne korake po registraciji. Da bi uporabnike kljub temu vzpodbudili k izpolnjevanju dodatnih nastavitev, jim v uporabniškem oknu predstavimo vrstico napredka, ki jih ob obisku z miško pozove k akciji (slika 3.6).



Slika 3.5: Obvestilo ob odklepu dosežka.

Seveda obstaja še mnogo možnosti uporabe tehnik iz oblikovanja igralnosti, kot sta nagrajevanje z naključnimi predmeti (angl. random drops) in skrivanje jajčk (angl. easter egg) [20]. Prav tako je možna tudi povezava strani z igrami narejenimi za Facebook, kar doda strani še dodatno družbeno komponento in razpoznavnost.



Slika 3.6: Vrstica napredka izpolnjenosti osebnih nastavitev.

3.8 Načrtovalni vzorci

Večino problemov s katerimi se srečamo kot programerji, so že mnogokrat obravnavali drugi programerji. Z uporabo načrtovalnih vzorcev lahko te izkušnje uporabimo. Načrtovalni vzorci prečistijo pogoste probleme, definirajo preizkušene rešitve in opišejo verjetne izide. Načrtovalni vzorec je torej analiziran problem s predlagano, dobro, praktično rešitvijo [18]. Načrtovalne vzorce je potrebno v aplikaciji pogosto prilagoditi specifični rešitvi [14].

Najpomembnejši načrtovalni vzorci, ki smo jih uporabili med razvojem naše spletne strani so: prilagojeni načrtovalni vzorec za implementacijo arhitekture MVC (poglavje 3.1) (angl. model-view-controller), vzorec ponavljalcev (angl. iterator) za dostop do zadetkov iz baze z uporabo knjižnice ADOdb in prilagojen vzorec nadzornika strani (angl. page controller) za osnovno strukturo posameznih strani [14]. Vsaka logična stran ima svojega nadzornika strani, ki skrbi za ustrezen dinamičen izpis vsebine glede na zahtevo. Primer uporabe je prikazan v poglavju 4.4.

Nekaj načrtovalnih vzorcev smo definirali sami. Primer takšnega je implementacija sistema odklepanja dosežkov (poglavje 3.7), kjer smo želeli rešitev zastaviti čim bolj splošno za morebitne razširitve. Definirali smo tudi vzorec za obdelovanje obrazcev in sicer kot razred `formValidator`, ki ga na več mestih uporabljamo pri delu z obrazci. Primer uporabe razreda `formValidator` je opisan v poglavju 4.4.1.

Vzorec ponavljalcev uporabljamo na primer pri paginaciji (angl. pagination), da shranimo rezultate poizvedbe za dodatno obravnavno (izpis 3.8). Za ta vzorec je tipična uporaba metode `moveNext` za nalaganje naslednjega rezultata poizvedbe.

```
1  /**
2  * Process query, populate results to items property
3  *
4  * @return boolean False if row count not retrieved
5  */
6  private function _fetchResults()
7  {
8   $query = str_ireplace("SELECT", "SELECT SQL_CALC_FOUND_ROWS", $this->↩
       query);
9   $query .= " LIMIT " . ($this->page - 1) * $this->pageSize . "," . $this↩
       ->pageSize;
10
11  // Execute query and store results
12  $rs = $this->db->Execute($query);
13  while(!$rs->EOF) {
14   $this->items[] = $rs->fields;
15   $rs->moveNext();
16  }
17  $rs->Close();
18  // Call result callback if defined
19  if(is_callable($this->resultManipulateCallback)) {
20   call_user_func($this->resultManipulateCallback, $this);
21  }
22
23  // Get num of rows
24  $rs = $this->db->Execute("SELECT FOUND_ROWS() as row_count");
25  if($rs && !$rs->EOF) {
26   $this->numOfPages = ceil( $rs->fields['row_count'] / $this->pageSize );
27  }
28  else {
29   return false;
30  }
31  return true;
32 }
```

Izpis 3.8: Uporaba vzorca ponavljalcev pri shranjevanju rezultatov poizvedbe.

Poglavje 4

Aplikacija NomNom HopHop

4.1 Podatkovni model

Za razvoj podatkovnega modela smo uporabili orodje MySQL Workbench, ki omogoča lažje in bolj pregledno delo z omejitvami in indeksi. Pri razvoju podatkovnega modela smo uporabili notacijo, ki nam omogoča lažje in bolj pregledno delo s poizvedbami SQL. Vsa polja tabel, razen primarnih ključev, imajo v imenu predpone, ki se začnejo z ustrezno enolično okrajšavo glede na celoten podatkovni model in jih od opisa polja ločimo s podčrtajem _. Pri razvoju podatkovnega modela smo bili posebej pozorni tudi na integritetne omejitve, neokrnjenost podatkov in uporabo indeksov.

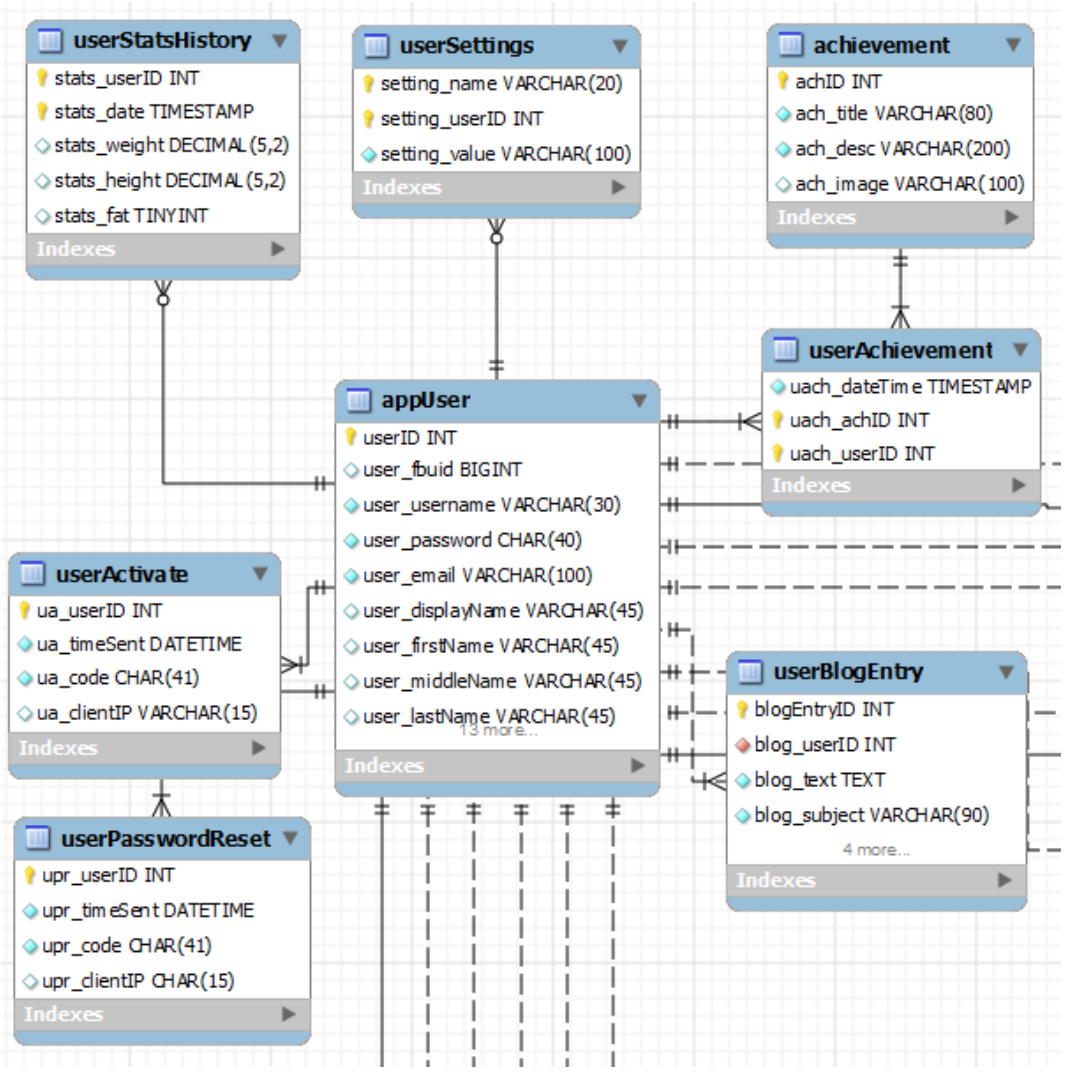
Na sliki 4.1 lahko vidimo povezave med nekaterimi tabelami, tesno navezane na uporabniške aplikacije, ki so predstavljene s tabelo `appUser`. Vsak uporabnik ima za primarni ključ enoličen identifikator `userID`, ki se uporablja skozi celotno aplikacijo in v podatkovnem modelu v mnogih tabelah nastopa kot tuji ključ. Uporabniki imajo preko indeksov tabele zagotovljene tudi enolične identifikatorje Facebook računa (`user_fbuid`), uporabniškega imena (`user_username`) in javnega imena (`user_displayName`).

Tabelo `userStatsHistory` uporabljamo za sledenje višine in teže uporabnikov, enolično določeno s časom vnosa in identifikatorjem uporabnika.

V tabelo `userSettings` shranjujemo poljubne nastavitve uporabnikov. Par tujega ključa in imena nastavitve zagotavlja enoličnost nastavitev.

Tabelo `userActivate` uporabljamo pri registraciji uporabniških računov, kjer v polje `ua_code` shranjujemo avtomatično proizvedeno kodo za aktiviranje uporabniškega računa.

Podobno strukturo kot tabela `userActivate` ima tudi tabela `userPasswordReset`, ki jo uporabljamo pri preverjanju veljavnosti zahtev za ponasta-

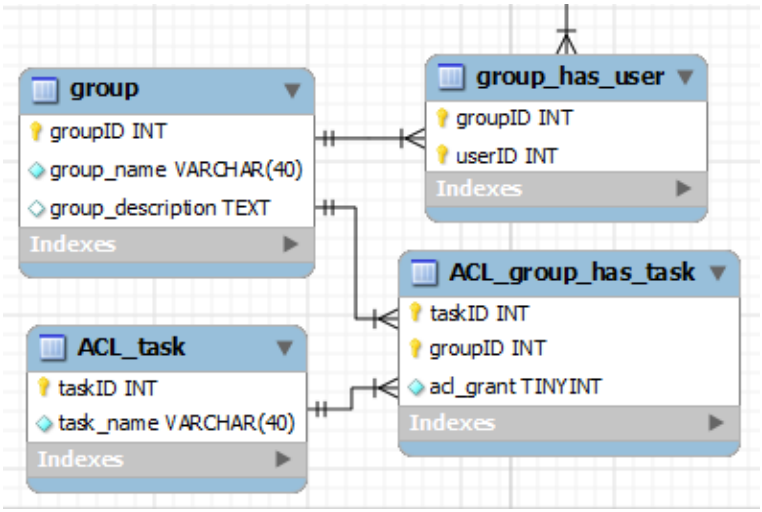


Slika 4.1: Tabele podatkovnega modela povezane z entiteto uporabnik.

vitev uporabniških gesel.

Za možnost storitve uporabniških spletnih dnevnikov imamo pripravljeno tudi tabelo `userBlogEntry`.

Z uporabo tabel `achievement` in `userAchievement` smo realizirali podatkovni del sistema dosežkov, ki uporabnike spodbuja k raziskovanju in uporabljanju aplikacije. V tabeli `achievement` shranjujemo tudi nize za imena in opise dosežkov, ki se lahko nato uporabljajo za lokalizacijo. Tabelo `userAchievement` pa uporabljamo za sledenje dosežkov uporabnikov.

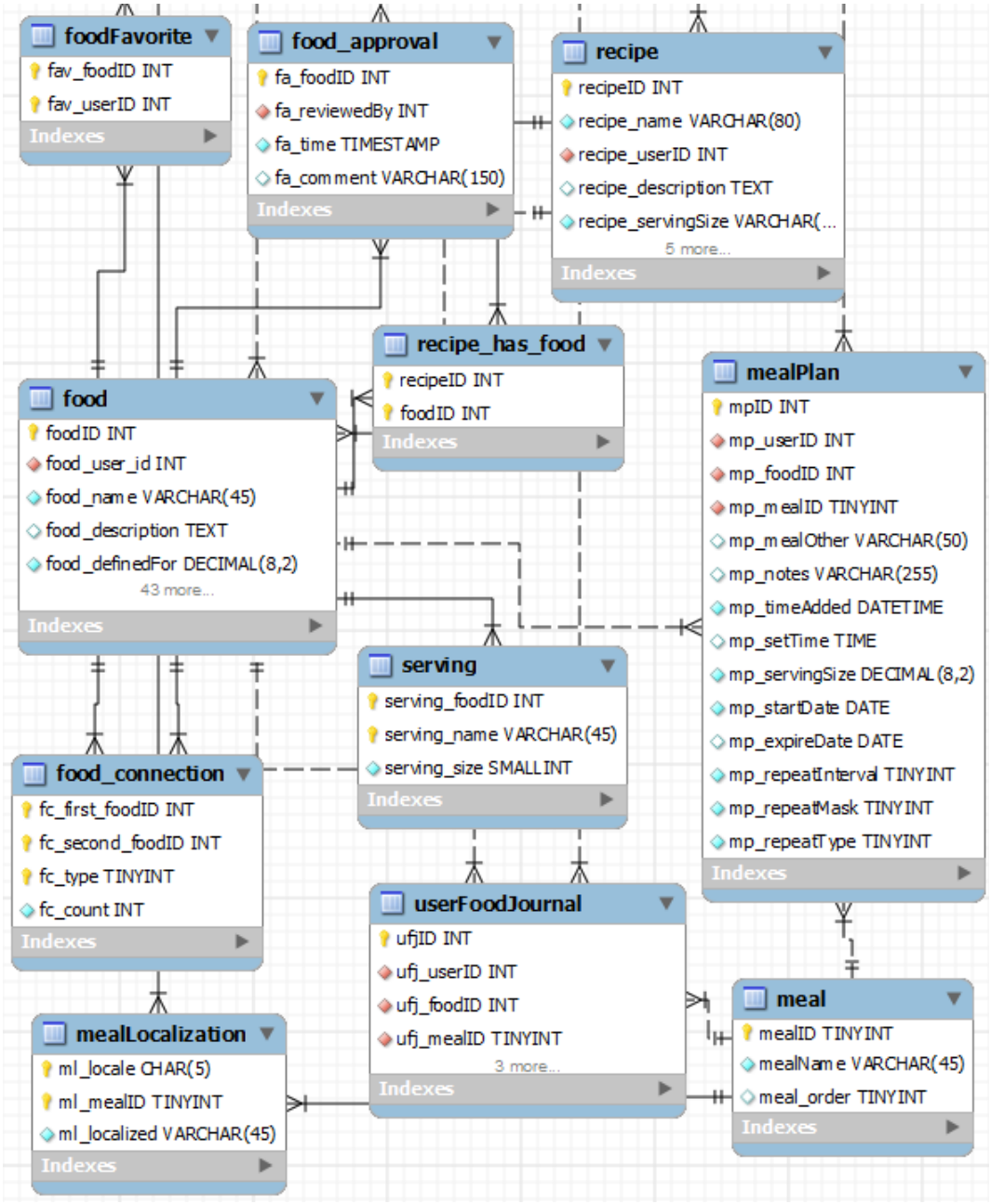


Slika 4.2: Tabele za realizacijo seznama za kontrolo dostopa.

Za realizacijo seznama za kontrolo dostopa uporabljamo tabele s slike 4.2. V tabelo `group` shranjujemo veljavne skupine, s tabelo `group_has_user` pa povezujemo uporabnike z vsemi skupinami, katerim pripadajo. Opravila seznama za kontrolo dostopa shranjujemo v tabelo `ACL_task`, pravice do izvajanja teh opravil za posamezne skupine pa v tabelo `ACL_group_has_task`. S spreminjanjem vrednosti polja `acl_grant` v tabeli `ACL_group_has_task`, lahko skupinam določamo različne načine dostopa do opravil.

Slika 4.3 nam predstavi tabele potrebne za realizacijo sledenja prehrane uporabnikov.

Srce sistema sledenja prehrane je tabela `food` v kateri je vsako živilo enolično določeno s ključem `foodID`. Na polje `food_name` nismo postavili enolične omejitve, saj želimo uporabnikom omogočiti čim večjo fleksibilnost pri vnosu lastnih živil, medtem ko za konsistentnost in enoličnost živil v javnem seznamu skrbimo z uporabo vmesnika za preverjanja vnosov. V tabeli `food` uporabljamo polji `food_public` in `food_approved` kot zastavici za objavo v javnem seznamu. Če je zastavica `food_public` nastavljena na vrednost nič, pomeni, da uporabnik tega živila ne želi deliti z drugimi. V nasprotnem primeru pa je stanje živila odvisno še od zastavice `food_approved`. Če je ta nastavljena na pozitivno vrednost, je živilo odobreno in objavljeno v javnem seznamu, če je nastavljena na negativno vrednost pa je bila zahteva za objavo zavrnjena, sicer pa se živilo nahaja v čakalni vrsti za odobritev. Vsak vnos v tabeli `food` vsebuje tudi obvezno numerično polje količine, za katero je živilo definirano, `food_definedFor`, ki se uporablja kot izhodišče



Slika 4.3: Tabele za sledenje prehrani.

za preračunavanje hranilnih vrednosti v naši aplikaciji. Tabela `food` vsebuje še kopico drugih polj, ki določajo hranilne vrednosti, znamko, uporabnika itd.

Za sledenje odobravanja in dodajanja živil v javni seznam uporabljamo tabelo `food_approval`, kamor shranjujemo tudi morebitne pripombe revizorjev.

V tabelo `serving` shranimo pogoste in priporočene velikosti porcij za živila, ki jih določijo vnašalci.

Tabela `foodFavorite` povezuje uporabnike z njihovimi priljubljenimi živili.

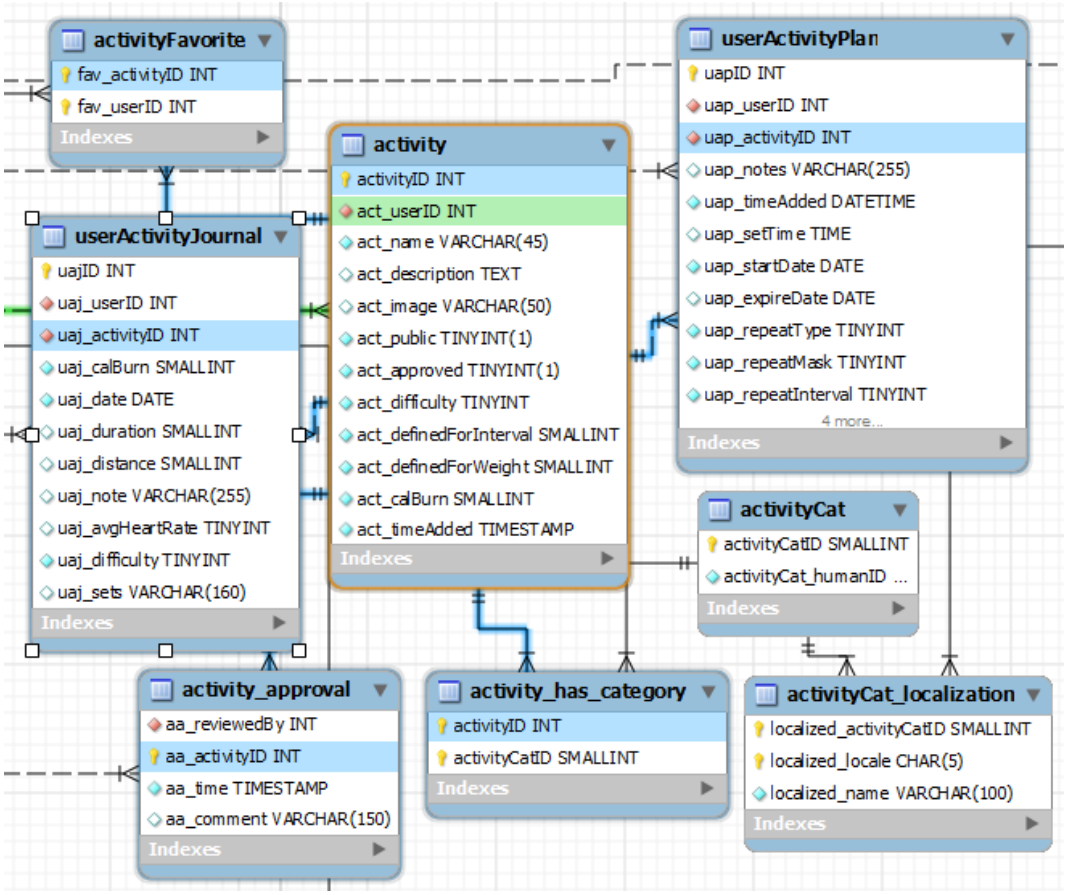
Z uporabo tabele `food_connection` lahko med živili vzpostavljamo različne povezave. V polje `fc_type` shranjujemo vrsto povezave, v polje `fc_count` pa število ponovitev. Primer takšne povezave je povezava živil, ki jih uporabniki ponavadi zaužijejo skupaj, kar lahko uporabnikom predstavimo kot seznam predlog za naslednjo živilo, ki ga lahko zabeležijo.

Za uporabnikovo sledenje prehrani podatke o zaužitih živilih shranjujemo v tabelo `userFoodJournal`. To so najbolj številčni vnosi v aplikaciji, zato je tabela minimalistična in vsebuje le potrebne povezave v obliki tujih ključev, datum in velikost obroka za preračunavanje. Tabela `meal` in `mealLocalization` nam omogočata bolj dinamično in fleksibilno definicijo različnih obrokov, s podporo večjezičnosti. Za načrtovanje obrokov in prehrane shranjujemo vnose v tabelo `mealPlan`, katere polja `mp_repeatInterval`, `mp_repeatMask` in `mp_repeatType` uporabljamo za čimbolj fleksibilno ponavljanje načrtov.

Za interval ponavljanja si lahko izberemo preproste intervale, kot so vsakih nekaj dni, vsak isti dan v mesecu in podobno za leta. Pri tem se uporabita ustrezni vrednosti za polji `mp_repeatType` in `mp_repeatInterval`. V primeru tedenske ponovitve pa uporabimo bitno masko, realizirano z 1 bajtom v polju `mp_repeatMask`, kjer nastavimo ustrezne vrednosti posameznih bitov in nato z bitnimi operacijami preverjamo potrebe po ponovitvah.

Naša aplikacija omogoča tudi vnose receptov, za kar uporabljamo tabelo `recipe`. Uporabniki se lahko tudi za recepte odločijo, da jih objavijo v javni seznam, za kar uporabljamo podobno tehniko kot pri živilih. Vmesnik za pisanje receptov vzpodbuja uporabo živil na voljo v aplikaciji in ob uporabi teh shranjujemo v tabelo `recipe_has_food` povezave med recepti in živili, kar lahko kasneje uporabimo za priporočila in druge napredne prijeme.

Za sledenje aktivnosti uporabnikom omogočamo podobne funkcionalnosti kot pri prehrani z uporabo tabel na sliki 4.4. Tabela `activity` vsebuje aktivnosti, ki jih lahko vnašajo tudi uporabniki in podobno kot pri vnašanju živil polji `act_approved` in `act_public` uporabljamo za objavo v javni seznam. Vsaka aktivnost je definirana za določeno težo in časovni interval, kar omogoča ustrezno preračunavanje glede na uporabnika. Tabela `activity` je precej po-



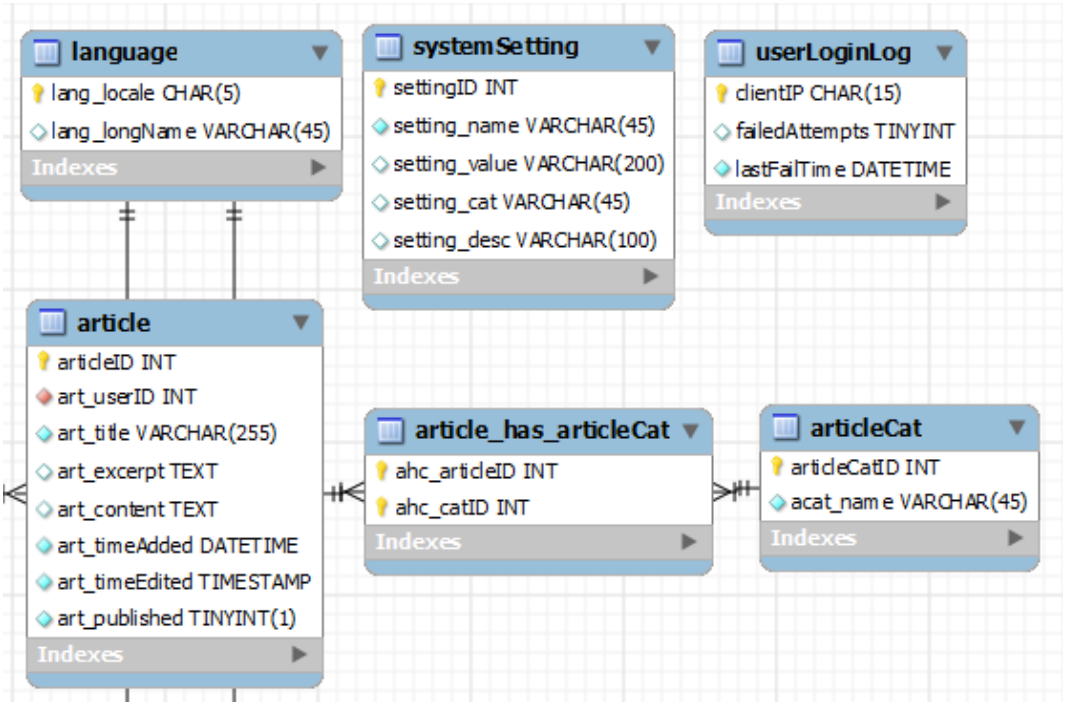
Slika 4.4: Tabele za sledenje aktivnosti.

dobna tabeli `food` in služi kot osnova za beleženje aktivnosti uporabnikov.

Vsaka aktivnost ima določene tudi kategorije. Omejeno število kategorij določimo v sistemu z vnosi v tabelo `activityCat` in ustreznimi prevodi za lokalizacijo v tabelo `activityCat_localization`. Za povezavo aktivnosti z ustreznimi kategorijami, uporabljamo vnose v tabeli `activity_has_category`.

Prav tako si lahko uporabniki tudi aktivnosti dodajo med priljubljene, kar beležimo v tabelo `activityFavorite`. Za nadzor dodajanja aktivnosti v javen seznam je tu spet tabela `activity_approval`. Za beleženje aktivnosti uporabnikov v tabelo `userActivityJournal` vpisujemo ustrezne povezave med uporabniki in aktivnostmi. Pri vnosu se lahko uporabnik odloči za ročen vnos porabe kalorij ali pa v aplikaciji uporabi kot izhodišče vrednosti porabe kalorij, ki jih nato aplikacija preračuna glede na uporabni-

kovo težo in čas opravljanja aktivnosti. Podobno kot pri načrtovanju obrokov lahko uporabniki načrtujejo svoje aktivnosti in v ta namen uporabljamo tabelo `userActivityPlan`. Za fleksibilno ponavljanje, tako kot pri živilih, tukaj spet uporabljamo polja `uap_repeatType`, `uap_repeatInterval` in `uap_repeatMask`.



Slika 4.5: Preostale tabele podatkovnega modela.

Preostale tabele so podane na sliki 4.5. Tabela `language` je šifrant jezikov. V tabelo `systemSetting` lahko shranjujemo poljubne sistemske nastavitve, ki jih lahko nato spreminjamo preko nadzorne plošče za moderatorje. Za varovanje pred napadi z grobo silo pri prijavi uporabljamo tabelo `userLoginLog`, kamor shranjujemo število neuspelih prijav glede na naslov IP uporabnika.

Za objavljanje člankov in novic, ki se obravnavajo kot ena izmed kategorij člankov, uporabljamo tabele `article`, `article_has_articleCat` in `articleCat`. Z zastavico `art_published` določimo ali je članek objavljen ali ne. Vsak članek je lahko objavljen tudi v več različnih kategorijah, kar povezujemo s tabelo `article_has_articleCat`.

Pri realizaciji podatkovnega modela je pomembna tudi uporaba ustreznega nabora znakov, za kar smo izbrali UTF-8. Za večino tabel smo uporabili pogon InnoDB, ki nam omogoča tudi uporabo transakcij.

4.2 Sistem uporabnikov in skupin

Naša spletna aplikacija je večuporabniška in za čimbolj naraven način grupiranja in določanja pravic smo se odločili za uporabo sistema skupin. Vsak registriran uporabnik lahko pripada večim skupinam, medtem ko so vsi neprijavljeni uporabniki združeni v skupino gosti. Za določanje skupin smo izdelali dinamičen vmesnik in omogočili enostavno dodajanje novih skupin in njihovih pravic v kasnejši fazi razvoja.

Nekaj skupin, ki jih uporabljamo: gostje, registrirani uporabniki, administratorji, moderatorji za prehrano in pisci. Z dinamičnim in odprtim sistemom skupin si odpremo tudi možnosti za dodajanje skupin po potrebi, kot na primer posebne skupine za pajke spletnih iskalnikov, zaupanja vredne uporabnike, uporabnike s prepovedmi ipd.

Kot goste obravnavamo vse neprijavljene uporabnike, na voljo pa imajo omejen spekter interakcij. Seveda pa so gostje zelo pomembni in jih obravnavamo kot bodoče člane ter jih na več mestih pozivamo k registraciji ali prijavi za uporabo več funkcionalnosti sistema. Prijavljeni uporabniki imajo dostop do funkcionalnosti kot so sledenje svoje prehrane, vnašanje živil, vnašanje receptov, mešanje živil, sledenje aktivnosti, sledenje spremembam lastne teže, prirejanje lastnih nastavitev v uporabniški nadzorni plošči ipd.

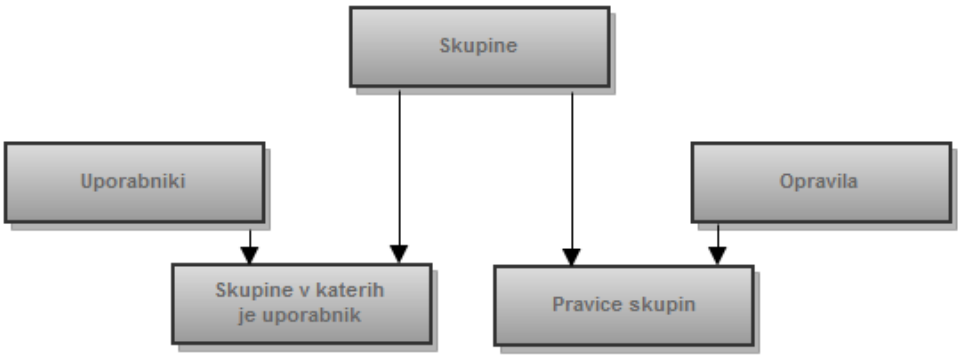
Najmanjše skupine uporabnikov pa so tiste, ki skrbijo za delovanje sistema. To so na primer moderatorji živil, ki urejajo in skrbijo za zahteve za vnos v javni seznam ter izboljšujejo kakovost podatkovne zbirke živil. Podobno nalogo imajo tudi moderatorji za aktivnosti. Administratorji skrbijo za nastavitve celotnega sistema, bdijo nad dejavnostmi moderatorjev in ob potrebi skrbijo za napredovanje uporabnikov. Posebna skupina so tudi pisci, ki imajo pravico objavljanja člankov in novic, njihovo urejanje in skrb za morebitne druge pisne vire.

Za realizacijo uporabnikov v sistemu uporabljamo razred `User`. Na sliki 4.6 so prikazane najpomembnejše metode razreda `User`, ki smo jih uporabili na več mestih med razvojem spletne strani. Za dostop do privatnih atributov razreda uporabljamo javne metode, ki se začno z `get`.

4.3 Seznam za kontrolo dostopa in meniji

Za varnostni sistem smo uporabili seznam za kontrolo dostopa, ki temelji na določanju vlog skupinam ali s kratico RBAC (angl. role based access control). V tem modelu vse uporabnike združimo v skupine, ki jim nato določimo pravice za izvajanje posameznih opravil. Na sliki 4.7 vidimo, da je lahko en uporab-

Slika 4.6: Najpomembnejše metode razreda za uporabnike `User`.



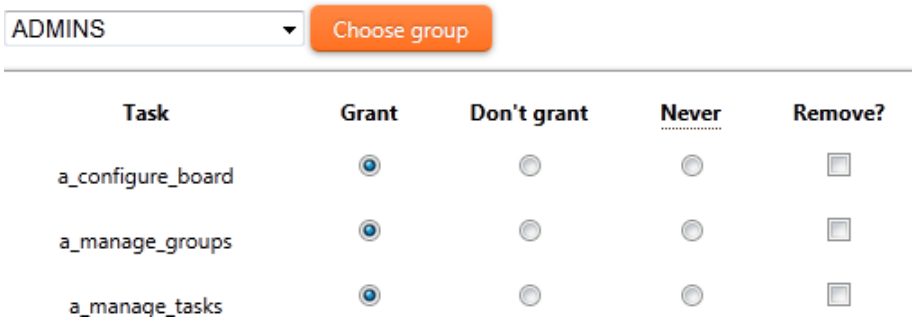
Slika 4.7: Razmerja med gradniki seznama za kontrolo dostopa.

nik vključen v več skupin, zato pri določanju pravic uporabljamo tri skupine dostopov. Skupina lahko uporabniku dostop do opravila omogoči, onemogoči, ali pa izklicno prepove, kar uporabniku onemogoči dostop do opravila, tudi če bi mu članstvo v neki drugi skupini dostop omogočilo. Pravice lahko administratorji skupinam dinamično določajo preko administracijskega vmesnika za pravice skupin v MCP, prikazanega na sliki 4.8. Za homogeno delo s pravicami dostopa s kodo v izpisu 4.1 določimo konstante za dostop v zaglavni datoteki

za konstante.

```
1 // ACL (Access control list) permissions
2 // 1 - Grant // 0 - Don't Grant // -1 - Never grant, override if task ↩
      permission added by other group
3 define("ACL_PERMISSION_GRANT", 1);
4 define("ACL_PERMISSION_NO", 0);
5 define("ACL_PERMISSION_NEVER", -1);
```

Izpis 4.1: Definicija konstant za delo s seznamom za kontrolo dostopa.



Slika 4.8: Administratorski vmesnik za določanje dostopa do opravil za skupine.

Ko uporabniškim skupinam določimo pravice za izvajanje določenih opravil, lahko v aplikaciji uporabljamo razred `Security`, z metodami katerega pogosto preverjamo, ali imajo uporabniki pravice izvajati določena opravila. Deli aplikacije kot so dodajanje živil, beleženje aktivnosti in podobno, od uporabnikov zahtevajo dostop do določenih opravil (ACL_TASK_U_FOOD_CREATE, ACL_TASK_U_ACTIVITY_LOG ipd.). Zahtevana opravila lahko v aplikaciji določimo že ob začetku obravnavanja zahteve in jih podamo konstruktorju razreda `Page` kot polje. Tako podana zahtevana opravila aplikacija preveri takoj in nepooblaščenim uporabnikom onemogoči dostop v celoti. Mogoče pa je uporabljati tudi statično metodo razreda `Security` imenovano `tasksGrantedToUser`, z uporabo katere lahko med izvajanjem programa preverimo, ali ima uporabnik ustrezne pravice za izvajanje danega opravila. Podan je primer preverjanja, ki se uporablja pri zahtevah AJAX za načrtovanje obroka:

```
1 if(!Security::tasksGrantedToUser($db, $userID, ACL_TASK_U_PLAN_MEAL)) {
2  throw new Exception("Unauthorized.", ERROR_UNAUTHORIZED);
3 };
```

Z uporabo dinamičnega sistema pravic, kot je RBAC, si močno olajšamo tudi razvoj dinamičnih menijev. V ta namen smo napisali razred `menuItem`,

ki ga uporabljamo za dinamično gradnjo menijev. V izpisu 4.2 konstruktor razreda `menuItem` zahteva tri obvezne parametre; enoličen identifikator objekta, skupino, kateri objekt pripada in polje potrebnih pravic za prikaz objekta menija uporabniku. Vsakemu objektu `menuItem` lahko hierarhično določimo tudi polje podrejenih objektov `menuItem`, ki jih nato rekurzivno obravnavamo med gradnjo menija.

```
$menuItems = array(
 new menuItem("MENU_USER_REGISTER", "ucp", ACL_TASK_U_CAN_REGISTER),
 new menuItem("MENU_USER_LOGIN", "login", ACL_TASK_U_CAN_LOGIN),
 new menuItem("MENU_USER_CP", "ucp", ACL_TASK_U_CONTROL_PANEL, array(
  new menuItem("MENU_USER_LOGOUT", "", ACL_TASK_U_CAN_LOGOUT)
)),
```

Izpis 4.2: Primer definicije dinamičnega menija.

Pri dinamični gradnji menija za del aplikacije, ki ga uporabnik trenutno zahteva, skrbimo tudi za označevanje ustrezne aktivne skupine, upoštevajoč tudi hierarhične menije. Ob izpisu in oblikovanju menijev je potrebno biti pozoren tudi na območje, ki ga uporabnik aktivira ob prehodu z miško in alternativno s spreminjanjem fokusa [10], zato smo v CSS datoteke vnesli ustrezne zapise `focus` in `hover` ter razširili aktivno območje na celotne okvirje menijev.

Z uporabo razreda `Security` med gradnjo menija za vsak objekt `menuItem` preverimo, če ima uporabnik pravico dostopa do opravila in se na podlagi tega odločimo za prikaz. Vsak objekt `menuItem` ima tudi atribut `showDisabled`, ki ga uporabljamo za prikaz funkcionalnosti gostom, da pritegnemo nove registrirane uporabnike. Če je atribut `showDisabled` nastavljen na logični `true`, se ta element menija prikaže tudi nepooblaščenim uporabnikom, vendar je neaktiven.

4.4 Razvoj sistema registracije in prijave

V tem razdelku bomo opisali razvoj dela aplikacije, ki ga uporabljamo za registracijo in prijavo uporabnikov. Ta del sistema smo izbrali, ker reprezentativno prikazuje uporabljene tehnike skozi razvoj celotne aplikacije.

Za rast, ustrezno uporabo in zbiranje vsebine s strani uporabnikov si želimo privabiti čim več aktivnih registriranih uporabnikov. Tako na primer pri ogledu živil izrišemo gumbe za beleženje prehrane, dodajanje med priljubljene in načrtovanje tudi gostom. Če gost poskusi uporabiti te funkcionalnosti, oziroma, ko na gumb postavi kazalec miške, ga povabimo k registraciji z vmes-

nikom na sliki 4.9. Za izpis tega vmesnika uporabljamo knjižnico za namige jQuery Tooltip, grafično prirejeno za naše namene.

Slika 4.9: Gradnik za privabljanje novih registriranih uporabnikov.

Gost lahko do vmesnika za registracijo dostopa preko omenjenih namigov, ustreznega elementa menija, kot dodatno možnost pri prijavi, ali pa kar s SEO prijaznim URL naslovom /registracija. Za izbiro ustrezne kode uporabljamo prilagojeni vzorec nadzornika strani (izpis 4.3).

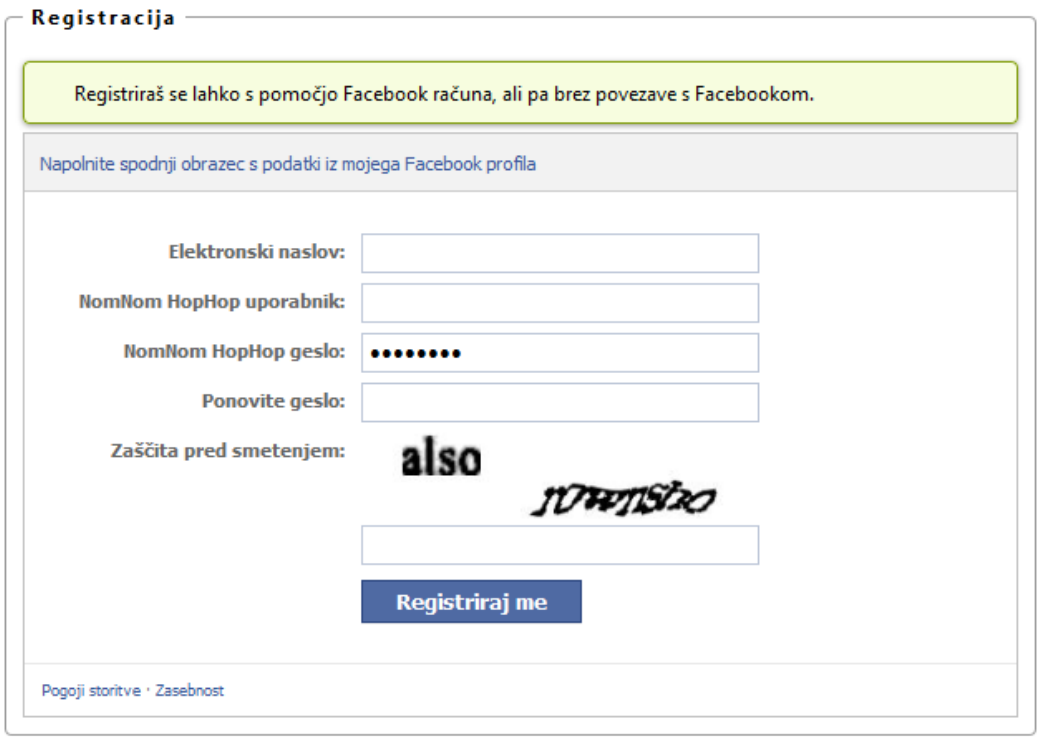
```
1  define("MODE_DEFAULT", 0);
2  define("MODE_REGISTER", 1);
3  define("MODE_ACTIVATE", 2);
4  define("MODE_PASSWORD_RESET", 3);
5  ...
6  if(!isset($_GET['mode'])) {
7      $mode = MODE_DEFAULT;
8  }
9  ...
10 else if("register" == $_GET['mode']) {
11     $mode = MODE_REGISTER;
12 }
13 ...
```

Izpis 4.3: Primer uporabe vzorca nadzornika strani.

Pri sistemu registracije smo implementirali dva povezana pristopa. Prvi je klasičen sistem spletne registracije, kjer od uporabnika zahtevamo najbolj osnovne podatke in potrditev preko spletne pošte. Drugi pristop pa je registracija z uporabo vmesnika za Facebook, ki uporabnikom Facebooka omogoča takojšno registracijo brez potrebe po potrjevanju preko spletne pošte.

4.4.1 Registracija brez Facebooka

Najprej si poglejmo postopek klasične registracije. Na sliki 4.10 je prikazan obrazec za klasično registracijo brez uporabe Facebook računa uporabnika. Od uporabnika zahtevamo vnos veljavnega enoličnega naslova spletne pošte, enoličnega uporabniškega imena, dvakratni vnos gesla in zaščito pred smetenjem s tehniko CAPTCHA.



Slika 4.10: Klasičen vmesnik za registracijo novih uporabnikov.

Pri poljih uporabniško ime in spletna pošta uporabljamo tehnike AJAX, za preverjanje enoličnosti vnosov ter tako uporabnika opozorimo na že uporabljene vrednosti še pred končno oddajo obrazca. Koda za preverjanje enoličnosti je hitra, preprosta, varna in razširljiva na preverjanje različnih vrednosti polj iz tabele uporabnikov.

```
1 @require("ajax_header.php");
2 try {
3  if("email" == $_POST['type']) {
4    $field = "user_email"; $q = $_POST['user_email'];
5  }
```

```
6  else if("username" == $_POST['type']) {
7    $field = "user_username";
8    $q = $_POST['user_username'];
9  }
10 ...
11 $sql = "SELECT COUNT(userID) FROM " . DB_USERS_TABLE . "
12        WHERE {$field} = {$db->qstr($q)}";
13 $res = $db->Execute($sql);
```

Izpis 4.4: Preverjanje vnosov z uporabo tehnik AJAX.

Izvajanje programa v izpisu 4.4 začnemo z vključitvijo datoteke *ajax_header.php*, skupne vsem programom PHP, ki jih uporabljamo v navezi s tehniko AJAX. V datoteki *ajax_header.php* izvedemo vse pogoste in potrebne vključitve datotek, kot so datoteka s konstantami, datoteka z nastavitvami in nekatere datoteke z razredi. Vzpostavimo tudi povezavo s SUPB-jem, uporabniško sejo in varnostno shemo. Ker je dobro vodilo pri varnosti spletnih aplikacij nezaupanje vsem uporabniškim vnosom, vrednosti polja `field` nastavimo eksplicitno med izvajanjem programa na poznane vrednosti. Prav tako za preverjanje obstoja vrednosti iskanega niza uporabljamo metodo `qstr()` razreda ADOConnection, ki zagotavlja ustrezno filtriranje pred napadi z vrivanjem. S klicem metode `Execute()` izvedemo poizvedbo SQL in v naslednjih korakih preprosto preverimo količino zadetkov. Program kot rezultat izpiše vrednosti v notaciji JSON, kar pričakujemo pri izvajanju JavaScripta v uporabnikovem brskalniku. Za kodiranje v notacijo JSON uporabljamo PHP funkcijo `json_encode()`.

Če je obrazec pravilno izpolnjen in Facebook potrdi pravilno rešitev izziva CAPTCHA, potem nadaljujemo z obravnavno registracije na strežniku v programu `user_register.php`. Podatke za registracijo prejmemo v obliki podpisane zahteve (angl. signed request) preko vmesnika za registracijo s Facebookom. Z uporabo naše knjižnice za delo s Facebookom, skrivnim in javnim nizom za Facebook aplikacijo iz podpisane zahteve razberemo vsebino. Uporaba metode `parseSignedRequest` je prikazana v izpisu 4.5

```
1 $fb = new FBInterface($_CONFIG['fb']['appID'], $_CONFIG['fb']
2             ['app_secret'], "", $_CONFIG['fb']['permissions'],
3             "", $_CONFIG['salt']);
4
5 if(null == $response = $fb->parseSignedRequest($_POST['signed_request']))
6 {
7  throw new Exception("Malformed facebook input.",
8                      USER_REG_TRANSACTION_ERROR);
9 }
```

Izpis 4.5: Uporaba metode `parseSignedRequest` za obdelavo podatkov prejetih s strani Facebooka.

Nato z uporabo razreda za preverjanje veljavnosti obrazcev preverimo veljavnost podatkov, ki jih je vnesel uporabnik. Razred za preverjanje veljavnosti obrazcev `formValidator` uporabimo tako, da konstruktorju podamo polje objektov `validatorItem`, ki predstavljajo polja obrazca. Prva dva argumenta konstruktorja objekta `validatorItem` skrbita za preslikavo imen spremenljivk v obrazcu v ustrezna polja v podatkovni bazi. Če je tretji argument postavljen na logični res, potem je to polje obvezno, sicer pa ne. Ostali argumenti niso obvezni, z njimi pa lahko določimo še naprimer podatkovno vrsto polja in privzete vrednosti. V izpisu 4.6 najprej izdelamo objekt `formValidator` in ga napolnimo z ustreznimi elementi, nato pa uporabimo metodo `validateInput`, ki kot argument sprejme polje uporabniških vnosov.

```
1  // Verify registration data
2  $fv = new formValidator(array(
3   new validatorItem("name", "user_displayName", false),
4   new validatorItem("email", "user_email",  true),
5   new validatorItem("username", "user_username",  true),
6   new validatorItem("password", "user_password",  true)
7  ));
8  if(!$fv->validateInput($response['registration'])) {
9   throw new Exception("Missing required input.", ↩
        USER_REG_TRANSACTION_ERROR);
10 }
```

Izpis 4.6: Preverjanje veljavnosti uporabniških vnosov z razredom formValidator.

Če so vnešeni podatki v ustreznem formatu in vsa zahtevana polja ustrezno izpolnjena, potem še enkrat preverimo enoličnost uporabniškega imena in e-poštnega naslova s poizvedbo SQL, saj preverjanje v brskalniku s tehnikami AJAX ni zagotovilo za ustreznost podatkov, ker lahko napadalec na primer enostavno onemogoči izvajanje JavaScripta. V naslednjem koraku zašifriramo uporabnikovo geslo z uporabo statične metode `encodePassword` razreda `User`:

```
1  $encodedPass = User::encodePassword($response['registration']['password'↩
       ], $_CONFIG['salt']);
2  $fv->items['password']->setValue($encodedPass);
```

Sledi vnos novega uporabnika v sistem, generacija kode za aktiviranje računa in zapis te kode v tabelo za aktiviranje uporabnikov `userActivate`. Ker zahtevamo, da se izvedeta obe operaciji vnosov v podatkovno bazo ali nobena, uporabimo transakcije. V izpisu 4.7 za vnos novega uporabnika proizvedemo ustrezen stavek SQL z uporabo metode `getInsertSQL()` razreda `formValidator`, ki glede na prej podana polja proizvede čist in ustrezen

stavek SQL.

```
$db->StartTrans();
// Insert new user
$db->Execute($fv->getInsertSQL($db, DB_USERS_TABLE));

// Generate activation code
$code = sha1(mt_rand(10000,999999).time() . $_CONFIG['salt'] .
             $response['registration']['email']);

$sql = "INSERT INTO " . DB_USER_ACTIVATE . "
      (ua_userID, ua_timeSent, ua_code, ua_clientIP)
    VALUES (LAST_INSERT_ID(), now(),
      '{$code}', '" . getClientIP() . "')";

$db->Execute($sql);

// Complete transaction or rollback
if($db->HasFailedTrans() ) {
 $transFail = true;
}
$db->CompleteTrans();
```

Izpis 4.7: Transakcija za aktiviranje novega uporabniškega računa.

Po uspešnem vnosu v podatkovno bazo uporabniku pošljemo potrditveno e-pošto z uporabo metode `mailUserActivation` razreda `User`, prikazano v izpisu 4.8. Metoda `mailUserActivation` pripravi ustrezno besedilo in naslovnika ter pošiljanje pošte prepusti metodi `sendAsyncEmail`. Z metodo `sendAsyncEmail` z uporabo PHP funkcij `curl_` pošljemo asinhrono zahtevo za pošiljanje pošte programu `async_mailer.php`, ki izvede pošiljanje pošte z uporabo knjižnice `PHPMailer`. Asinhrono pošiljanje dosežemo z nastavitvijo nizke vrednosti za parameter `CURLOPT_TIMEOUT_MS` funkcije `curl_setopt`, kot je prikazano v izpisu 4.8. Takšen preprost asinhron sistem pošiljanja je nadomestek za bolj kompleksen sistem pošiljanja pošte z uporabo vrst, vendar občutno pohitri obravnavanje postopka registracije v uporabnikovem brskalniku, saj uporabniku izpišemo odgovor, še preden se izvede pošiljanja pošte.

```
// Send activation code via email
/* ... */
$user::mailUserActivation($response['registration']['email'], $code))
/* ... */
return User::sendAsyncEmail($sendTo, $subject, $body);
/* ... */
$ch = curl_init();
/* ... */
curl_setopt($ch, CURLOPT_TIMEOUT_MS, 100);
$response = curl_exec($ch)
```

Izpis 4.8: Nekaj izsekov kode za pošiljanje e-pošte.

Če je med obravnavo registracije prišlo do napake ali izjeme, uporabnika o tem obvestimo z lokaliziranimi sporočili:

```
catch (Exception $e) {
 $smarty->assign("FORM_ERROR_TRIGGERED", true);
 $smarty->assign("FORM_REGISTER_MSG",
  $page->localization->localize("reg_error_msg_code_" .
                  $e->getCode(), "ucp"));
}
```

Uporabnik nato prebere e-pošto, ki mu jo sistem pošlje v obliki HTML. Potrditvena pošta vsebuje zahvalo, neposredno povezavo za aktiviranje računa z uporabo parametra `code` v obliki GET spremenljivke in kot alternativno možnost še povezavo do strani za aktiviranje računa z ustrezno šifro.

Če uporabnik pošte ni prejel, jo je izgubil ali pa je prišlo do kakšne napake pri primerjavi šifer, uporabniku ponudimo tudi preprost vmesnik za pošiljanje nove šifre za aktiviranje računa. Pri tem od uporabnika zahtevamo e-poštni naslov, ki ga je uporabil pri registraciji. Za zaščito pred zlorabo je obrazec zaščiten s tehniko CAPTCHA, ob obravnavanju zahteve za izdajo nove šifre na strežniku pa obravnavamo le zahteve za uporabniške račune, ki imajo ustrezen vnos v tabeli za aktiviranje računov s kodo v izpisu 4.9.

```
// Check if account not activated
$sql = "SELECT userID FROM " . DB_USERS_TABLE .", " . DB_USER_ACTIVATE
. " WHERE
    userID = ua_userID AND
    user_email = {$db->qstr($_POST['email'])}";

$res = $db->Execute($sql);
if(!$res || $res->NumRows() < 1) {
  throw new Exception("No account to activate with email.",
              NO_ACCOUNT_ACTIVATION_FOUND);
}
```

Izpis 4.9: Preverjanje veljavnosti postopka za aktiviranje uporabniškega računa.

Uporabniki lahko torej svoj račun aktivirajo s pošiljanjem parametra `code` v obliki GET zahteve, kot v primeru aktiviranja preko povezave v e-pošti, ali pa s pošiljanjem POST zahteve v primeru uporabe obrazca za aktiviranje računa. V obeh primerih se pred zlorabo s poskušanjem aktiviranja zaščitimo z uporabo izziva CAPTCHA, ki ga prikažemo glede na neuspelo število poskusov iz istega naslova IP. Implementacija zaščite je prikazana v izpisu 4.10.

```
$recaptchaResponse = recaptcha_check_answer(
 $_CONFIG['recaptcha']['privateKey'],
 getClientIP(),
 $_POST["recaptcha_challenge_field"],
```

```
5   $_POST["recaptcha_response_field"]);
6
7  if (!$recaptchaResponse->is_valid) {
8   $recaptchaError = $recaptchaResponse->error;
9   throw new Exception("Captcha mismatch", USER_REG_CAPTCHA_MISMATCH);
10 }
```

Izpis 4.10: Zagotavljanje veljavnosti uporabniške zahteve s preverjanjem rešitve izziva CAPTCHA.

Če se šifra ujema s tisto shranjeno v tabeli `userActivate`, lahko nadaljujemo z aktiviranjem računa. Iz tabele `userActivate` odstranimo vnos za aktiviranje računa, nato pa uporabnika dodamo v skupino registriranih uporabnikov:

```
1 $sql = "INSERT INTO " . DB_GROUP_HAS_USER_TABLE . " (groupID, userID) ↩
      VALUES (" . GROUP_REGISTERED_USERS . ", {$userID})";
2 $db->Execute($sql);
```

Proces zaključimo z obvestilom uporabniku in sprožimo odklepanje dosežka za aktiviranje računa:

```
1 $achievements->triggerAchievement(ACH_ACCOUNT_ACTIVATED);
```

Na koncu uporabnika preusmerimo na vmesnik za prijavo v sistem.

4.4.2 Registracija z uporabo Facebooka

Hitrejši postopek registriranja za uporabnika je uporaba vmesnika za registracijo na spletnih straneh z uporabo omrežja Facebook. Facebook vmesnik za registracijo na spletnih straneh omogoča hitro in prepoznavno prijavo na več spletnih straneh z uporabo Facebook računov. Uporabnik Facebooka mora ob registraciji, Facebook aplikaciji spletne strani, na katero se prijavlja, omogočiti dostop do osnovnih podatkov in Facebooku dovoliti izvedbo registracije. V večini primerov mora uporabnik vpisati le še enolično uporabniško ime in ustrezno geslo, klikniti gumb za registracijo in potrditi pogoje uporabe. Na spletno stran se lahko uporabnik nato prijavi tudi s svojim Facebook računom in mu ni potrebno vpisovati še enega uporabniškega imena in gesla.

Za uporabo sistema za registracijo preko omrežja Facebook smo z aplikacijo za Facebook razvijalce, najprej registrirali novo Facebook aplikacijo, ki predstavlja našo spletno stran. Za komuniciranje s Facebookom uporabljamo enolični identifikator aplikacije in skrivno šifro, ki ju za našo aplikacijo proizvede Facebook. Več o uporabi Facebooka in varnosti takšne komunikacije lahko najdete v poglavju 4.5. Za realizacijo obrazca za registracijo in prijavo

smo uporabili paket Facebook JavaScript SDK, Facebook PHP SDK in naš razred `FBInterface` za delo s Facebookom. Uporabili smo tudi imenski prostor (angl. namespace) *xmlns:fb="http://www.facebook.com/2008/fbml"*, ki nam omogoča uporabo jezika FBML. Tako obrazec za registracijo vstavimo v predloge z uporabo značk `<fb:registration>` prikazano v izpisu 4.11.

```
<fb:registration redirect_uri="{$SERVER_URI}/{$L_G_seo_register}"
 fields='[
 {"name":"name","view":"prefilled"},
 {"name":"email"},
 {"name":"gender","view":"prefilled"},
 {"name":"birthday","view":"prefilled"},
 {"name":"username","description":"NomNom username", "type":"text"},
 {"name":"password"},
 {"name":"captcha","view":"not_prefilled"}
]'
onvalidate="validate"></fb:registration>
```

Izpis 4.11: Uporaba značke `fb:registration`.

Slika 4.11: Facebook vmesnik za registracijo novih uporabnikov.

Uporabniku vmesnik za registracijo na sliki 4.11 preko omrežja Facebook ponudi delno izpolnjen obrazec ter mu obenem prikaže prijatelje, ki so se že registrirali na našo spletno stran z uporabo Facebooka. Spet s tehnikami AJAX preverimo enoličnost podatkov in v primeru ustrezno izpolnjenega obrazca prejmemo podpisano zahtevo za registracijo od Faceboka.

V programu *user_register.php* spet raztolmačimo podpisano zahtevo, preverimo enoličnost uporabniškega imena in e-pošte ter zakodiramo geslo. Zaradi načina komunikacije med Facebookom in našo aplikacijo bi lahko napadalec vnesel lažna polja, zato s kodo v izpisu 4.12 preverimo veljavnost podatkov z uporabo spremenljivke `registration_metadata`, ki jo podpisani zahtevi doda Facebook.

```
$fields = json_decode($response['registration_metadata']['fields']);
// Only verify our email metadata is set correctly.
if(!property_exists($fields[1], "name") ||
  $fields[1]->name !== "email" ||
  1 !== count((array)$fields[1])) {
 throw new Exception("Metadata match fail.", USER_REG_TRANSACTION_ERROR);
}
```

Izpis 4.12: Uporaba podatkov `registration_metadata` za preverjanje pristnosti vnosov.

Preverimo tudi, če je uporabljeni Facebook račun že povezan s katerim uporabniškim računom in veljavnost rojstnega datuma. Nato izvedemo vnos uporabnika podobno kot pri normalnem toku registracije. Dodatno pa v tabelo za nastavitve uporabnikov `userSetting` vnesemo tudi podatke o spolu in rojstni datum. Uporabnik je tako registriran brez potrebe po potrjevanju preko e-pošte, poleg tega pa že vnesemo nekaj podatkov, ki bi jih moral uporabnik vnesti tudi sicer med uporabo naše spletne aplikacije za namene preračunavanja porabe kalorij. Uporabnik ima sedaj svoj uporabniški račun povezan s svojim Facebook računom in se lahko v spletno stran prijavi že z enim klikom, brez vnašanja uporabniškega imena in gesla.

V tem poglavju smo se osredotočili na razvoj PHP logike, seveda pa smo morali pri razvoju registracije pripraviti tudi ustrezne predloge, JavaScript programe, oblikovne datoteke, prevode in nize za SEO.

4.5 Povezava s Facebookom

4.5.1 Kako in zakaj?

Omrežje Facebook je daleč najbolj popularno socialno omrežje na svetu in se ponaša z 800 milijonov aktivnih uporabnikov. Preveden je v več kot 70 jezikov in razvijalcem ponuja razvojno platformo z dostopom do velike količine podatkov, ki jih uporabniki delijo s svetom ali z razvijalci. Z uporabo omrežja Facebook si lahko občutno povečamo vidljivost in obisk naše spletne strani. Prav tako lahko za uporabnike Facebooka močno poenostavimo komentiranje, registracijo in prijavo. Še bolj pomemben pa je prikaz aktivnosti prijateljev uporabnika, ki jo prinese povezava s Facebookom. Tako na primer uporabniku pri procesu registracije izpišemo seznam nekaterih njegovih prijateljev na Facebooku, ki so se že registrirali na naši strani. Podobno izpišemo seznam prijateljev in število ljudi, ki jim je všeč določeno živilo pri uporabi gumba *všeč mi je*.

Proces povezave naše spletne strani s Facebookom smo začeli s prijavo nove Facebook aplikacije na razvojnem omrežju Facebooka. Z uporabo Facebookove aplikacije za razvijalce smo nastavili nekaj osnovnih nastavitev in prejeli identifikator aplikacije ter skrivno šifro, ki jo potrebujemo za delo s Facebook API-jem. Uporabljamo več različnih pristopov in orodij glede na funkcionalnost, ki jo potrebujemo. Imenski prostor FBML uporabljamo za registracijo, prijavo in vključitev meta podatkov; knjižnico Facebookov PHP SDK največ za avtorizacijo in dešifriranje podatkov; Facebookov Javascript SDK pa za dodatne funkcionalnosti v brskalniku. Razvili smo tudi svojo PHP knjižnico za delo s Facebookom imenovano `FBInterface`, s pomočjo katere pohitrimo programiranje pogostih opravil, kot sta na primer pridobivanje informacij od Facebooka in dešifriranje teh sporočil.

4.5.2 Primeri uporabe FBML

Za uporabo označevalnega jezika za Facebook FBML (angl. Facebook Markup Language) smo v glavo dokumenta HTML dodali ustrezen URL naslov:

```
1 <html xmlns="http://www.w3.org/1999/xhtml"
2       xmlns:fb="http://www.facebook.com/2008/fbml">
```

Tako smo lahko na primer v glave HTML odgovorov na zahteve za prikaz informacij o živilih dodali tudi meta značko `og:image`, ki jo omrežje Facebook uporabi za objavo ustrezne slike pri objavljanju zgodb iz podanega URL

naslova. Primer takšne objave je zgodba, da je uporabnik aktiviral gumb *všeč mi je*. Meta značke `og:image` vstavimo v glavo dokumenta s kodo:

```
<meta property="og:image" content="{\$META_OGIMAGE}" />
```

Značka FBML, ki smo jo uporabili za prijavo na stran s pomočjo omrežja Facebook se imenuje `fb:login-button`:

```
<fb:login-button on-login="FBLogin(arguments)">
{\$L_G_login_with_facebook} </fb:login-button>
```

Med razvojem strani smo uporabili tudi znački `fb:like` in `fb:registration` (poglavje 4.4.2).

4.5.3 Facebook komentarji

Pomemben gradnik sodobnih spletnih strani so komentarji uporabnikov, ki prinesejo strani dodatno vrednost in povečajo zanimanje uporabnikov. Najbolj osnovni način komentiranja je povezava časovno urejenih komentarjev uporabnikov z določenim delom spletne strani. Za najbolj osnovno implementacijo komentarjev ne potrebujemo drugega kot tabelo, v kateri hranimo vsebino, datum, identifikator strani in identifikator uporabnika. Seveda je potrebno poskrbeti tudi za nadzor nad neprimernimi komentarji, varovanje pred napadi XSS, CSRF, vrivanjem SQL in podobno.

Za implementacijo sistema komentarjev smo se odločili uporabiti sistem komentiranja za spletne strani omrežja Facebook. Prednosti uporabe takšnega sistema so prijaznost do uporabnikov, prepoznaven vmesnik za komentiranje, vidljivost na družabnem omrežju Facebook, večje število komentarjev ter preskušeno varen in hiter razvoj. Za pomembno prednost pred bolj anonimnimi metodami komentiranja se izkaže tudi večja zrelost in odgovornost komentatorjev, saj je vsak komentar enolično povezan z njihovim Facebook računom.

Seveda ima takšen pristop tudi slabe strani, kot so manj fleksibilne možnosti pri razvoju in nadzoru sistema, potreba po uporabi Facebook računov za komentiranje in izpostavljenost napakam omrežja Facebook. Kljub temu menimo, da je uporaba Facebook komentarjev na sodobnem spletu smiselna in pozitivna.

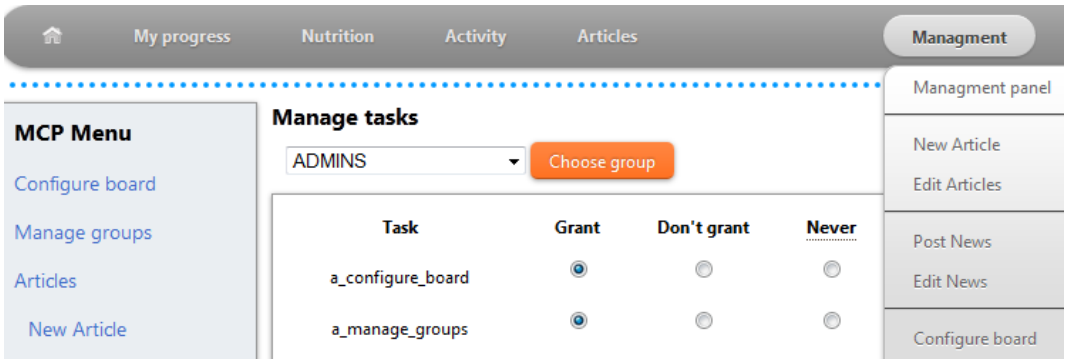
Implementacija sistema komentarjev je preprosta in spet povezana z našo aplikacijo na omrežju Facebook. Uporabimo značko `fb:comments`, nastavimo nekaj osnovnih parametrov in vmesnik za komentarje je pripravljen:

```
<fb:comments href="{$FB_PAGE_URI}"
  num_posts="{$FB_COMMENT_NUM_POSTS}" width="880">
</fb:comments>
```

Za upravljanje komentarjev lahko uporabljamo Facebook račune, ki jim dodelimo ustrezne pravice. Dostop do dodatnih funkcionalnosti, kot so preverjanje števila komentarjev, omogoča uporaba jezika FQL (angl. Facebook Query Language), s katerim lahko izvajamo poizvedba nad odprtim grafom (angl. OpenGraph).

4.6 Administracija

Za administracijo naše aplikacije smo napisali nadzorno ploščo za moderatorje MCP (angl. Moderator Control Panel) prikazano na sliki 4.12. Za določanje pravic dostopa do opravil v MCP uporabnike vključimo v ustrezne skupine. Primeri skupin, ki imajo dostop do opravil v MCP so administratorji, moderatorji za živila, moderatorji za aktivnosti in moderatorji vsebine. Člani skupin kot so moderatorji za živila imajo dostop le do določenih opravil v MCP, seveda pa je lahko uporabnik član več skupin naenkrat in tako dostopa do razširjenega spektra opravil.

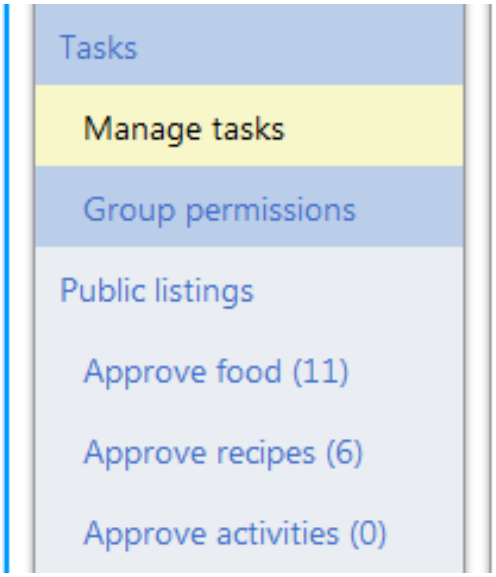


Slika 4.12: Vmesnik za administracijo.

Za izpis ustreznih elementov glavnega menija smo uporabili naš dinamičen sistem menijev, poleg tega pa smo napisali še razširjen meni posebej prirejen za MCP, prikazan na sliki 4.13.

Z MCP-jem lahko spreminjamo globalne nastavitve spletne strani, urejamo skupine, pravice skupin, urejamo članstvo uporabnikov v skupinah, objavljamo in urejamo članke, urejamo opravila in skrbimo za objavljanje v javnih seznamih.

Za objavljanje živil v javni seznam uporabljamo vmesnik v MCP, do katerega imajo dostop administratorji in moderatorji za živila. Na sliki 4.14 lahko



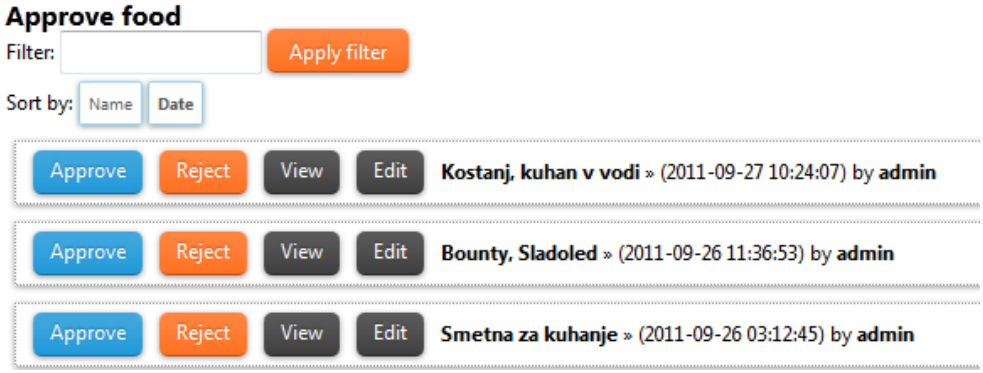
Slika 4.13: Razširjeni meni v nadzorni plošči za moderatorje.

vidimo vmesnik, ki ga uporabljamo za odobritev in zavrnitev objav. Moderator lahko uporabi filter za iskanje določenih živil in zadetke ureja po imenu ali datumu objave. Na voljo je tudi vmesnik za urejanje živil, da lahko moderator popravi morebitne pomanjkljivosti.

Če moderator meni, da je živilo ustrezno izpolnjeno in ga v javnem seznamu še ni, potem lahko izbere gumb za odobravanje in z uporabo tehnike AJAX se bo seznam osvežil ter živilo dodal v javni seznam. V ozadju v tabeli `food` spremenimo zastavice `food_public` in `food_approved` na pozitivne vrednosti. Če se moderator odloči zahtevo za objavo zavrniti, mu ponudimo tudi okno za vpis razloga, ki ga shranimo v tabelo `foodApprove`.

Za pisanje in urejanje člankov ter novic uporabljamo obogateni uporabniški vmesnik CKEditor, ki moderatorjem omogoča WYSIWYG (angl. what you see is what you get) urejanje vsebine. Napisali smo tudi vmesnik za dodajanje kategorij, ki z uporabo AJAX tehnike dinamično izpisuje vse kategorije v sistemu, obenem pa omogoča vpis novih.

Pri razvoju nadzorne plošče za moderatorje smo v precej primerih uporabili tehnike AJAX za pohitritev dela. Posebno pozornost smo seveda namenili tudi varnosti in vestnemu, večkratnemu preverjanju uporabniških pravic za izvajanje posameznih opravil.



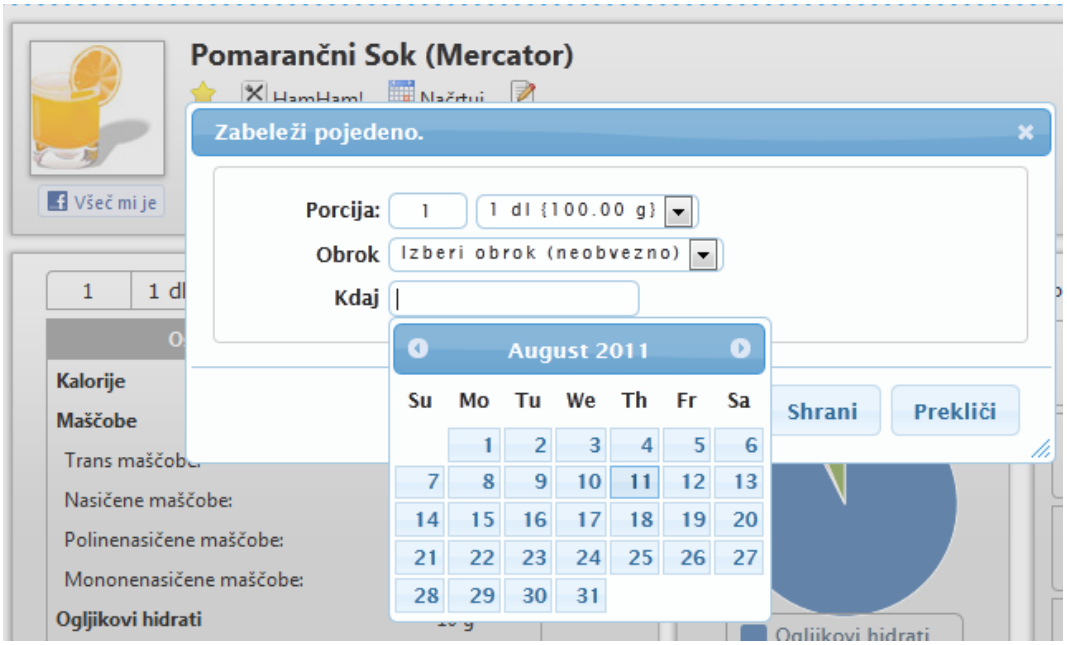
Slika 4.14: MCP vmesnik za odobritev objave živil v javni seznam.

4.7 Funkcionalnosti grafičnega vmesnika

Za konec si poglejmo še nekaj vmesnikov za delo z našo spletno aplikacijo.

Slika 4.15: Vmesnik za dodajanje živil, dostopen vsem registriranim uporabnikom.

Na sliki 4.15 je podan vmesnik za dodajanje živil, ki ga lahko uporabljajo vsi registrirani uporabniki. Z zvezdicami označujemo obvezna polja. Ob oddaji obrazci z JavaScript kodo preverimo veljavnost numeričnih polj, obveznih polj in podobno ter izpišemo ustrezna opozorila z rdečim besedilo, kjer je potrebno.



Slika 4.16: Vmesnik za beleženje prehrane, dostopen vsem registriranim uporabnikom.

Vmesnik za beleženje prehrane na sliki 4.16 uporabljamo v navezi s tehnikami AJAX, za prikaz dialogov pa uporabljamo funkcionalnosti knjižnice jQuery UI.

Pri izdelavi vmesnika za komentarje na sliki 4.17 smo uporabili FBML značko `fb:comments`, vmesnik pa je tudi ustrezno lokaliziran. Pri obrazcu za prijavo na sliki 4.18 uporabniku ponudimo možnost prijave s Facebookom ali pa z vpisom uporabniškega imena in gesla. Uporabnikom ponudimo tudi povezavo do obrazca za registracijo in obrazca za ponastavitev gesla.

Vmesnik za mešanje hrane, prikazan na sliki 4.19, uporabljamo za mešanje živil in kot pomoč pri pisanju receptov. Z mešalnikom lahko bolj preprosto vnašamo nova živila, ali ustvarjamo recepte ter določamo hranilne vrednosti tako ustvarjenih živil. Pri tem vmesniku za boljšo uporabniško izkušnjo uporabljamo tehnike AJAX in knjižnico jQuery.

Slika 4.17: Vmesnik za komentiranje z uporabo Facebooka.

Slika 4.18: Vmesnik za prijavo uporabnikov.

Slika 4.19: Vmesnik za mešanje živil in pisanje receptov.

Poglavje 5

Zaključek

V diplomski nalogi sem na primeru spletne strani *NomNom HopHop* predstavil orodja in tehnike za razvoj sodobnih spletnih strani.

Predstavil sem razvoj orodij in tehnik, ki so razvijalcem spletnih strani na voljo in standarde, ki se jih je dobro držati pri razvoju. Razvoj spletnih standardov, brskalnikov, pohitritev povprečnega internetnega dostopa in bolj zmogljivi osebni računalniki razvijalcem omogočajo uporabo vedno bolj bogatih, dinamičnih in zanesljivih prijemov. Mnogi spletni razvijalci so v začetku stoletja neradi uporabljali JavaScript, saj je bila podpora v brskalnikih tistega časa zelo nekonsistentna, računalniki pa manj zmogljivi. Danes JavaScript s pomočjo knjižnic kot je jQuery uporabljamo kot standarden jezik, ki ga razumejo vsi pomembni spletni brskalniki. Omogoča nam razvoj spletnih strani in aplikacij z vmesniki podobnimi klasičnim aplikacijam in uporabo tehnik AJAX.

Z uporabo tehnik AJAX močno povečamo odzivnost in funkcionalnost spletnih strani, saj uporabnikom ni potrebno čakati na ponovno nalaganje strani za vsako zahtevo, ki potrebuje odgovor strežnika. Seveda pa je potrebno tehnike AJAX uporabljati previdno in paziti na varnost, obnašanje in odzivnost nekonvencionalnih naprav, zgodovino brskanja in streženje vsebine pajkom spletnih iskalnikov.

Pri razvoju smo uporabljali preizkušene in odprte knjižnice, kjer je bilo to mogoče. Odprtost knjižnic nam omogoča lastne modifikacije po potrebi in boljši pregled nad varnostjo aplikacije. Odprta koda sama po sebi ne pomeni manj varne aplikacije, saj je varnost skozi skrivanje redko dobra izbira. Primer spletnega orodja, ki uporablja skrivanje kode je Flash, ki je mnogokrat tarča napadov in zlorab.

Varnost igra pomembno vlogo pri razvoju spletnih strani, saj smo na spletu

izpostavljeni napadalcem iz celega sveta noč in dan. Prikazal sem nekaj pogostih tehnik in napadov, ki se jih poslužujejo napadalci kot so napadi z vrivanjem SQL, napadi XSS, napadi CSRF in poskušanja z grobo silo. Pri razvoju kompleksne spletne strani moramo biti v vsaki fazi razvoja pozorni na možnost zlorabe. Še posebej pomembna je ustrezna obravnava in dvom v pristnost vseh podatkov, ki jih vnašajo uporabniki. Pri razvoju spletnih strani se je potrebno zavedati tudi, da napadalec največkrat ne bo uporabljal spletnega brskalnika in se tako na varnost, ki jo zagotavljajo brskalniki ne moremo zanašati.

Pokazal sem tudi, da sodobne spletne strani niso le kup statične vsebine, ampak dinamičen sistem, ki ga uporabniki s sodelovanjem in uporabo spreminjajo, dopolnjujejo in izpopolnjujejo. Srce ideje spleta 2.0 je sodelovanje uporabnikov pri nastajanju vsebine in izmenjavi informacij, kar med razvojem naše spletne strani s pridom uporabljamo. Pomemben korak v razvoju spletnih strani je tudi uporaba dodatnih semantičnih oznak in povezovanje storitev, ki jih nudijo druge spletne aplikacije. Tako smo našo spletno stran močno povezali z družabnim omrežjem Facebook in v HTML kodo dodali tudi posebne semantične oznake, ki so namenjene avtomatičnim objavam na omrežje Facebook.

Bistveno gonilo pri razvoju naše spletne strani je bila tudi odprtost za razširitev in čim bolj splošno zasnovane rešitve. Tako smo na primer uporabili predložni stroj in tako lažje implementirali prirejene vmesnike za mobilne naprave. Podobno bi lahko seveda uporabili tudi prirejene vmesnike za druge naprave ali uporabnike s posebnimi potrebami. Seveda obstaja še mnogo pristopov, ki jih lahko uporabimo v prihodnosti. Eden takšnih je uporaba tehnik strojnega učenja za grupiranje ljudi v skupine kot so diabetiki in vegetarijanci ter ustreznega prirejanja rezultatov iskanja in priporočil.

Naša spletna stran je zasnovana kot platforma, ki bi jo lahko uporabljali tudi z namensko razvitimi aplikacijami za mobilne operacijske sisteme Android, iOS, Windows Mobile ipd. Trg mobilnih aplikacij je poln možnosti in uporaba tehnik kot so določanje lokacije, uporaba bralnikov črtnih kod in povezave z drugimi uporabniki, lahko odprejo čisto nove dimenzije razvoja storitev.

Spletna stran še ni zaživela, želimo pa si, da bi bila ta storitev na voljo slovenskem trgu in se bomo za to potrudili.

Slike

Literatura

[1] R. Avsec, *Sodobni spletni iskalniki in optimizacija spletnih strani za učinkovito iskanje*, diplomsko delo, Fakulteta za računalništvo in informatiko, UL, 2008.

[2] B. Bibeault, Y. Katz, *jQuery in Action, Second Edition*, Manning Publications, 2010.

[3] A. Budd, S. Collision, C. Moll, *CSS Mastery: Advanced Web Standards Solutions, Second Edition*, Apress, 2009.

[4] K. Chellapilla, P. Simard, *Using Machine Learning to Break Visual Human Interaction Proofs (HIPs)*, Postopki o NIPS, 2004

[5] P. J. Deitel, H. M. Deitel, *AJAX, Rich Internet Applications and Web Development for Programmers*, Pearson Education, 2008.

[6] M. Firtman, *Programming the Mobile Web*, O'Reilly Media, 2010.

[7] D. Flanagan, *JavaScript: The Definitive Guide, Fourth Edition*, O'Reilly Media, 2001.

[8] J. Governor, D. Hinchcliffe, D. Nickull, *Web 2.0 Architectures*, O'Reilly Media, 2009, pogl. 3.

[9] B. Hoffman, B. Sullivan, *Ajax Security*, Addison-Wesley, 2008.

[10] B. Hunt, *Save the Pixel: The Art of Simple Web Design*, Samozaložba, 2008.

[11] J. Jerkovic, *SEO Warrior*, O'Reilly Media, 2009.

[12] J. Keith, *HTML5 for Web Designers*, A Book Apart, 2010.

[13] R. Murray, V. Rodwell, D. Bender, K. M. Botham, P. A. Weil, P. J. Kennelly, *Harper's Illustrated Biochemistry, Twenty-Eighth Edition*, McGraw-Hill Medical, 2009, pogl. 16.

[14] D. Reiersøl, M. Baker, C. Shiflett, *PHP in Action*, Manning Publications, 2007.

[15] J. N. Robbins, *Web Design in a Nutshell, Third Edition*, O'Reilly Media, 2006.

[16] B. Schwartz, P. Zaitsev, V. Tkachenko, J. Zawodny D., A. Lentz, D. J. Balling, *High Performance MySQL: Optimization, Backups, Replication, and More, Second Edition*, O'Reilly Media, 2008.

[17] C. Shiflett, *Essential PHP Security*, O'Reilly Media, 2006.

[18] M. Zandstra, *PHP Objects, Patterns, and Practice, Third Edition*, Apress, 2010.

[19] (2011) A history of the dynamic web. Dostopno na: http://royal.pingdom.com/2007/12/07/a-history-of-the-dynamic-web/.

[20] (2011) Easter egg. Dostopno na: http://en.wikipedia.org/wiki/Easter_egg_(media).

[21] (2011) Google Gruyere Web Exploits and Defences. Dostopno na: http://google-gruyere.appspot.com/.

[22] (2011) History - HTML WG Wiki. Dostopno na: http://www.w3.org/html/wg/wiki/History.

[23] (2011) IE6 Countdown. Dostopno na: http://mashable.com/2011/03/04/ie6-countdown/.

[24] (2011) ISO - FAQs - Language Codes. Dostopno na: http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/language_codes.htm.

[25] (2011) PHP Manual. Dostopno na: http://php.net/manual/en/index.php.

[26] (2011) Software Vulnerability Disclosure: The Chilling Effect. Dostopno na: http://www.csoonline.com/article/221113.

[27] (2011) SQL Injection – The Web Flaw That Keeps on Giving. Dostopno na: http://www.thesecurityblog.com/2011/09/sql-injection-the-web-flaw-that-keeps-on-giving/.

[28] (2011) Web Squared: Web 2.0 Five Years On. Dostopno na: http://www.web2summit.com/web2009/public/schedule/detail/10194.

[29] (2011) Wikipedia - AJAX. Dostopno na: http://en.wikipedia.org/wiki/Ajax_(programming).

[30] (2011) Wikipedia - HTML. Dostopno na: http://en.wikipedia.org/wiki/HTML.

[31] (2011) Wikipedia - jQuery. Dostopno na: http://en.wikipedia.org/wiki/JQuery.

[32] (2011) Wikipedia - MySQL. Dostopno na: http://en.wikipedia.org/wiki/MySQL.

[33] (2011) Wikipedia - PHP. Dostopno na: http://en.wikipedia.org/wiki/PHP.