UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Štrumbelj

Portal fitnes centra za delo s strankami in s povezavo do spletnega socialnega omrežja

DIPLOMSKO DELO
VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNISTVO IN INFORMATIKA

Mentor: doc. dr. Zoran Bosnić

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00543/2011
Datum: 01.04.2011

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

Kandidat: **MIHA ŠTRUMBELJ**

Naslov: **PORTAL FITNES CENTRA ZA DELO S STRANKAMI IN S POVEZAVO DO SPLETNEGA SOCIALNEGA OMREŽJA**

CUSTOMER SERVICE FITNESS CENTRE PORTAL WITH A CONNECTION TO A SOCIAL NETWORK

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija

Tematika naloge:

Pri poslovanju fitnes centrov v svetu predstavlja neobstoječa informacijska podpora za nadzorovanje ur vodenih vadb problem pri poslovanju. Evidentiranje prijav in obveščanje uporabnikov o posebnostih urnika vodenih vadb lahko doprinese tako k večjemu zadovoljstvu strank kot tudi omogoči upravi avtomatsko ocenjevati uspešnost obiskanosti in s tem poslovanja.
Kandidat naj v diplomski nalogi imlementira spletni portal, ki bo strankam fitnes centra omogočal prijavo na ure vodenih vadb. Portal naj upošteva spremenljivost urnikov in povezljivost do obstoječih uporabniških podatkov, ki pa naj jih dopolni še s podatki iz razširjenega socialnega omrežja. V nalogi naj kandidat predstavi razvoj in funkcionalnosti razvitega portala kot naj tudi ovrednosti njegovo skalabilnost.

Mentor:

doc. dr. Zoran Bosnić

Dekan:

prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Miha Štrumbelj,

z vpisno številko 63070451,

sem avtor diplomskega dela z naslovom:

Portal fitnes centra za delo
s strankami in s povezavo do spletnega
socialnega omrežja

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Zorana Bosnića
- 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 08.06.2011 Podpis avtorja:

Zahvala

Najlepše se zahvaljujem mentorju doc. dr. Zoranu Bosniću za mentorstvo in nasvete pri izdelavi diplomskega dela. Hvala tudi družini in prijateljem, ki so mi stali ob strani ter pretrpeli mojo odsotnost. Zahvaljujem se tudi podjetju Sokolgroup d.o.o., ki je nudilo vsebinsko pomoč pri izdelavi diplomskega dela ter za testiranje nastale programske rešitve.

Kazalo

Povzetek **1**

Abstract **2**

1 Uvod **3**

2 Informacijski sistem **5**
2.1 Vrste informacijskih sistemov 7
2.2 Opis izbranih vrst IS 7
2.2.1 CRM - Upravljanje odnosov s strankami 7
2.2.2 SCM - Upravljanje oskrbovalne verige 8
2.2.3 ERP - Poslovni informacijski sistemi 8
2.3 Metodologije razvoja, standardi in načela dobre prakse 9
2.3.1 Življenjski cikel programske opreme 10
2.3.2 Kaskadni način razvoja (Waterfall life-cycle) 11
2.3.3 Evolucijski razvoj 12
2.3.4 Agilne metodologije 13
2.4 Socialna omrežja 13
2.4.1 Facebook API 14
2.5 Tehnologije 14
2.5.1 Na strani strežnika 14
2.5.2 Na strani uporabnika 16
2.6 Razvojno okolje 20
2.6.1 Razvojno okolje Eclipse 20
2.6.2 MySQL Workbench 20

3 Razvoj portala fitnes centra za delo s strankami **22**
3.1 O podjetju 22
3.2 Opis postopka vodenja evidenc 24
3.3 Zajem zahtev 26

3.3.1 Funkcionalne zahteve 26
3.3.2 Nefunkcionalne zahteve 29
3.3.3 Identifikacija uporabnikov 29
3.4 Analiza projekta 30
3.4.1 Merila uspešnosti projekta 30
3.4.2 Osnutek predloga rešitve 31
3.4.3 Dogovor o poteku projekta 31
3.4.4 Profil uporabnikov informacijskega sistema 31
3.5 Analiza, načrtovanje in implementacija programske rešitve . . . 32
3.5.1 Dogovori poimenovanja in tehnike kodiranja 33
3.5.2 Arhitektura sistema 36
3.5.3 Primeri uporabe sistema 38
3.5.4 Podatkovni model 41
3.5.5 Uporabniški vmesnik 44

4 Opis rešitve 46
4.1 Portal 46
4.1.1 Jedro sistema 46
4.1.2 Prva stran 47
4.1.3 Administracija 48
4.1.4 Administracija urnikov 49
4.1.5 Uporabniški profil 49
4.1.6 Sporočila 50
4.1.7 Statistike 50
4.2 Integracija 52
4.2.1 Socialno omrežje Facebook 52
4.2.2 ARSO - podatki o vremenu 53
4.2.3 Informacijski sistem fitnes centra 53
4.3 Testiranje delovanja 53
4.3.1 Cilji testiranja 54
4.3.2 Zajem napak 54
4.3.3 Funkcionalno testiranje 54
4.3.4 Nefunkcionalno testiranje 55

5 Zaključek 61
5.1 Sklepne ugotovitve 61

A Podatkovni model sistema 63

Seznam slik 73

Seznam tabel **74**

Literatura **75**

Seznam uporabljenih kratic in simbolov

Seznam uporabljenih kratic in simbolov, ki morajo biti enotni v celotnem delu, ne glede na označevanje v uporabljenih virih.

IT Informacijske tehnologije.

IKT Informacijsko komunikacijske tehnologije.

WYSIWYG (angl.) What You See Is What You Get; Kar vidiš, to dobiš - paradigma programiranja.

RIA (angl.) Rich Internet Application; Interaktivne spletne aplikacije.

OOP Objektno Orientirano Programiranje.

RSS/ATOM (angl.) Really Simple Syndication; Standard, uporabljen za objavljanje vsebin.

JSON (angl.) JavaScript Object Notation; Odprt standard za izmenjavo podatkov.

SOAP (angl.) Simple Object Access Protocol; Protokol, namenjen izmenjavi strukturiranih podatkov med aplikacijami.

WSDL (angl.) Web Services Description Language; XML format zapisa za opis spletih storitev.

XML (angl.) Extensible Markup Language; Lahko berljiv strukturiran označevalni jezik

PHP (angl.) Hypertext Preprocessor; Skriptni programski jezik

GPL (angl.) General Public License; Licenca za prosto programiranje

DBMS (angl.) Database Management System: Sistem za upravljanje s podatkovno bazo

API (angl.) Application Programming Interface; Programski vmesnik

JDBC (angl.) Java Database Connectivity; Javin vmesnik za povezovanje s podatkovnimi bazami

DOM (angl.) Document Object Model; Objektni model podatkov v zvezi v XML zapisom

HTML (angl.) HyperText Markup Language; Jezik za označevanje nadbesedila

CSS (angl.) Cascading Style Sheets; predloge, ki določajo videz spletnih strani

Povzetek

Namen diplomskega dela je izdelava spletnega informacijskega sistema za vodenje evidence obiska skupinskih vadb. Elektronske evidence obiskov poenostavljajo delo osebja fitnes centra. Le-to lahko pridobljeni čas posveti dvigu kvalitete primarne storitve. Razvit informacijski sistem uporabnikom storitev olajša postopek prijavljanja ter odjavljanja udeležbe vodenih vadb, ponudniku storitev pa omogoča pridobivanje natančnejših podatkov o obiskanosti in priljubljenosti posamičnih vadb ter kakovosti inštruktorjev. V nalogi je opisan življenjski cikel ter metodologije razvoja programske opreme, prenesene na razvoj konkretnega programskega sistema. V delu je opisan razvoj spletnih aplikacij z odprtokodnimi tehnologijami PHP in Java programskim jezikom na spletnem strežniku Apache v kombinaciji s podatkovno bazo MySQL. Podprto je pridobivanje podatkov o trenutnih vremenskih razmerah in vremenskih napovedi iz XML servisa Agencije Republike Slovenije za okolje, sinhronizacija aplikacije s sistemom E-Check in povezava s socialnim omrežjem Facebook. Izdelana rešitev je opremljena z interaktivnim spletnim uporabniškim vmesnikom, ki od končnega uporabnika zahteva le sodobni spletni brskalnik brez nameščanja dodatnih vtičnikov oziroma druge programske opreme. Aplikacija je testirana v virtualnem okolju, ki je primerljivo s hipotetičnim končnim okoljem, za katerega je napisana. Rezultati so pokazali, da sprejemljive odzivne čase dobimo, tudi če aplikacijo sočasno uporablja trideset in več uporabnikov. Slovensko podjetje, ki ponuja storitve organizacije vodenih vadb, nam je med razvojem dalo na razpolago svoje vire in bo programske rešitve preizkusilo v praksi.

Ključne besede:

internet, informacijski sistem, vodene vadbe, planiranje virov, spletne tehnologije

Abstract

Purpose of this thesis is to develop an information system for online group workout booking system. It is designed to help fitness centre staff to optimise number of visitors on their classes and consequently to improve the quality of service, making it user-friendlier, as well. Information system simplifies the booking procedure and delivers more accurate participant data to the business. The thesis describes software development life cycle, gives brief information on some modern methodologies and tries to apply them into practice. There are plenty of technologies which promise fast development of high quality products. Some of the latest web server, database and user interface open source technologies were used during all phases of this project. The use of Java and PHP programming language on Apache web server in combination with MySQL database turned out to be the right mixture of technologies. Application core periodical retrieves data from XML weather service, synchronises client data with E-check system and uses advantages of Facebook Graph API. Designed solution is equipped with an interactive web interface which allows the use of any of the most common web browsers without the need of installing additional plug-ins or other software on the client side. Application has been tested in a virtual environment similar to the one for which it was designed. The tests showed satisfactory results even on higher load with thirty or more simultaneous users. The Slovenian fitness company which supported the project with its resources, has chosen to test the resulting software programme and consequently to implement it.

Key words:

internet, information systems, group workout, resource planning, world wide web technologies

Poglavje 1

Uvod

Današnji življenjski ritem narekuje hitro prilagajanje spremembam, pri tem pa pozabljamo na zadostno telesno aktivnost. Rešitev ponujajo podjetja, ki se ukvarjajo z nudenjem storitev organiziranja vodenih športnih aktivnosti. Sodoben človek si za šport pusti le malo časa, zato imajo taka podjetja veliko težav s privabljanjem novih strank.

Ena izmed bolj priljubljenih oblik športnih aktivnosti so vodene skupinske vadbe. Te so še posebej primerne za sodoben način življenja, saj ponujajo celostno vadbo, prilagojeno posamezniku v motivirani skupini. Ure vodenih vadb trajajo od petinštirideset minut do ene ure, zato so še toliko bolj primerne za natrpane urnike obiskovalcev.

Dobiček podjetij je pogojen z zadovoljstvom strank, zadovoljstvo strank pa je pogojeno s kvaliteto izvedbe vodenih vadb. Ta je odvisna od same vrste vadbe, inštruktorja, prostorov ter načina izvedbe. Zaradi zagotavljanja večje kvalitete izvedbe, fitnes centri od uporabnikov zahtevajo predhodno najavo udeležbe, kar jim omogoča bolj optimalno napolnjenost terminov.

Podjetja lahko zaradi določenih razlogov (nezadostno število prijavljenih, bolezen inštruktorja, ...) odpovejo termin, na katerega so bili uporabniki že prijavljeni, pogosto pa se zgodi, da stranke zaradi napolnjenosti napotijo v drug fitnes center ali na druge termine.

Večina podjetij problem rešuje z evidenco obiska v papirnati obliki, saj na trgu ne obstaja široko poznana elektronska rešitev za to problemsko domeno, namenski razvoj programske opreme pa veliko stane. Sistem z evidenco v papirnati obliki nima vgrajenih avtomatskih mehanizmov, ki bi v primerih sprememb ali odpovedi terminov poskrbela za obveščanje.

Cilj diplomske naloge je vzpostaviti programsko rešitev, ki bo podjetjem olajšala vodenje tovrstnih evidenc. Zaradi potreb po dosegljivosti in hitrosti

razvoja, smo se odločili za spletno rešitev. Spletna evidenca mora omogočati načrtovanje ter objavo urnika, prijavo ter odjavo na ure vodenih vadb.

V drugem poglavju je na kratko predstavljena teorija informacijskih sistemov, metodologije razvoja ter tehnologije in orodja, uporabljene pri nastanku portala fitnes centra.

Tretje poglavje opisuje zasnovo projekta razvoja portala. Razdelali smo profil uporabnikov, določili funkcionalne in nefunkcionalne zahteve ter načrtali podatkovni model. Naredili smo še analizo poteka projekta ter določili kriterije uspešnosti. Poglavje vsebuje tudi informacije o uporabljenih tehnologijah ter razvojnih okoljih.

Poglavje štiri predstavi razvito aplikacijo in njene module. Delovanje je ponazorjeno s posnetki bolj zanimivih zaslonov.

V petem poglavju predstavimo testiranje informacijskega sistema. Aplikacijo nameščeno v virtualno okolje, podobno končnemu, izpostavimo višjim obremenitvam in predstavimo rezultate testiranj.

Poglavje 2

Informacijski sistem

Količina informacij, ki se nahaja okoli nas je preprosto prevelika, da bi jo lahko uspešno obvladovali na klasičen način, ko imamo vse informacije 'v glavi'. Dandanes si ne moremo predstavljati življenja brez organizacije. Podjetja ne morejo poslovati brez učinkovitega informacijskega sistema.

Naivni bralec zna misliti, da je informacijski sistem nujno računalniško podprt sistem - torej neke vrste programska oprema, ki se izvaja na računalniku. Pojmovanje informacijskih sistemov je mnogo širše. Informacijski sistem [1] zajema množico medsebojno odvisnih komponent, ki zbirajo, procesirajo, hranijo in porazdeljujejo podatke in s tem podpirajo delovne procese v organizaciji z namenom doseganja predhodno definiranih ciljev. Informacijski sistem vsebuje podatke o organizaciji: notranja struktura podjetja, podatke o klientih, dobaviteljih in konkurenci. Poslovanja podjetij brez teh informacij bi lahko primerjali s prečkanjem prometnega križišča s prevezanimi očmi.

Poslovni procesi so dandanes povečini res računalniško podprti, vendar temu v preteklosti ni bilo tako. Primer klasičnega, računalniško ne-podprtega informacijskega sistema, bi bila javna uprava v rimskih časih. Hipotetičen primer dogodka, ki se zgodi v takem sistemu pa bi bil pobiranje davkov. Rimljani so morali hraniti podatke o premoženju posamičnega prebivalca imperija in podatke o plačanih prispevkih v davčno blagajno. Pri neplačanih terjatvah je moralo biti obveščeno sodišče, ki je sprožilo postopek izterjave. Z odlokom sodišča nato izterjevalec izterja dolg. Ko pride do primera, da v nekem naselju nihče noče plačati davkov, cesar nad kršitelje pošlje pristojen organ (v tem primeru vojsko).

Opisan je bil dovršen informacijski sistem, ki vključuje različne akterje (cesar, uradniki, sodišče, izterjevalci, vojska in prebivalstvo) ter opis medsebojne interakcije, ki ne vključujejo sodobne IKT (informacijsko-komunikacijske

tehnologije).

Sodobni informacijski sistemi so (večinoma) podprti s sodobnimi informacijskimi tehnologijami. Osrednji del vsakega informacijskega sistema je vedno človek in informacijski sistem je vedno bil bistven faktor preživetja združb. Laudon in Laudon, 2002, (citirano v [1]) ugotavljajta, da obstajajo 4 vidiki, ki spreminjajo poslovni svet in so naredili informacijske sisteme strateško bolj pomembne:

Globalna ekonomija: 25% ekonomskih aktivnosti v ZDA predstavlja mednarodni promet, tako morajo biti vse uspešne organizacije sposobne poslovati globalno. Podjetja morajo biti konkurenčna na novih trgih, pomembno je imeti poln nadzor nad dobavitelji, distribucijo proizvoda in biti v stiku s strankami 24 ur na dan, vse dni v letu. Uspešnost na globalnem trgu je pogojena z zmogljivimi informacijskimi sistemi.

Preobrazba industrije: industrijsko okolje čedalje bolj temelji na kvaliteti in znanju. Proizvodnja velikih količin produktov, ki v določenem trenutku niso primerni za trg, predstavlja podjetjem dodatno breme in zmanjšuje učinkovitost poslovanja. Storitve temeljijo na uporabi informacij in znanj, nekatere združbe se celo ukvarjajo le s tem (npr.: zdravstvo in šolstvo). Sodobne informacijsko podprte tehnologije so dandanes vključene v skoraj vsak produkt industrije (npr.: v avtomobil je vključeno veliko elektronike namenjene povečanju udobja ter varnosti v prometu).

Sprememba organizacijske strukture združb: tradicionalne združbe so organizirane hierarhično in so odvisne od vodstvenih in upravljavskih sposobnosti menedžmenta. V modernih organizacijah se organizacijska struktura spreminja v bolj fleksibilno kombinacijo virov. Pozornost podjetij se usmerja proti proizvodnji uporabnikom prilagojenih produktov in storitev.

E-podjetja: Podjetja, ki imajo dobro organizirano elektronsko poslovanje so veliko bolj sposobna prilagajanja hitrim spremembam okolja ter zahtevam uporabnikov storitev.

2.1 Vrste informacijskih sistemov

Iz prejšnjega poglavja vemo, da je pojem informacijskega sistema zelo širok. Pokaže se, da bi informacijske sisteme členili glede na formalnost (formalni, neformalni) in na stopnjo podprtosti z računalniškimi sistemi (nepodprto, podprta nekatera področja, pokritih večina področij ter internetna podjetja). V tem diplomskem delu se bomo bolj osredotočili na računalniško bolj podprte informacijske sisteme.

Glede na namembnost ločimo naslednje vrste informacijskih sistemov:

- transakcijski informacijski sistem (Transaction Processing Systems, TPS),
- odločitveni sistemu (Decision Support Systems, DSS),
- ekspertni sistemi (Expert Information Systems, EIS),
- upravljavsko-ravnateljski informacijski sistem (Management Information Systems, MIS),
- sistemi za upravljanje delovnih procesov (Workflow System, WS).

Računalniške implementacije informacijskih sistemov zelo pogosto pokrivajo več področij. Sistemi tipa DSS ter MIS temeljijo na transakcijah. Za kvalitetne podatke je smiselno beležiti dejanske poslovne transakcije in to ob nastanku dogodka, le tako lahko zagotovimo da so vneseni vsi podatki in da so le ti čim bolj točni. Sistemi za podporo odločanju ter sistemi za podporo upravljanju brez točnosti in pravočasnosti podatkov iz transakcijskih sistemov ne morejo dati dovolj kvalitetnih izhodov. Še več, rezultati obdelav v takšnih sistemih so lahko napačni in škodujejo poslovanju ter razvoju poslovnega sistema.

2.2 Opis izbranih vrst IS

2.2.1 CRM - Upravljanje odnosov s strankami

Sistemi CRM (Customer Relationship Management) za upravljanje odnosov s strankami so namenjeni celostnemu upravljanju tako z obstoječimi kot tudi s potencialnimi strankami. Podjetja se morajo osredotočiti na svoje stranke, na njihove kupne navade ter biti stalno v stiku z njimi. Cilj sistema CRM je ustvariti stranke ter jih obdržati. Glavno področje, ki ga pokrijemo s rešitvami CRM je prodaja, vendar lahko z njimi pokrivamo tudi marketing, svetovanje

ter podporo.

Primeri računalniških rešitev CRM so:

- Microsoft Dynamics CRM - rešitev podjetja Microsoft,
- Oracle CRM,
- SAP CRM,
- Intrix CRM - primer slovenskega CRM sistema.

2.2.2 SCM - Upravljanje oskrbovalne verige

Sisteme SCM (Supply Chain Management) so razvili za podporo povezovanja različnih partnerjev v oskrbovalni verigi. Velika proizvodna podjetja so odvisna od večjega števila dobaviteljev. Proizvodnja lahko zaradi zamud pri dobavi surovin ali polizdelkov stoji, kar povzroča večje stroške ter neizkoriščenost virov. Npr. dobavitelji, ki nastopajo kasneje v proizvodnji, ne vedo, da je prišlo do zamud pri dobavi. Kapacitete teh dobaviteljev bi ta čas lahko oskrbovale drugega naročnika. Že majhna zamuda (ne glede na razlog) vpliva na vse partnerje, ki sodelujejo v oskrbovalni verigi. Sistemi SCM preprečujejo take dogodke ter omogočajo sodelovanje med podjetji, skupno načrtovanje, izvajanje ter koordinacijo v celotni logistični mreži.

Sistemi za upravljanje z oskrbovalno verigo so po svoji naravi močno integrirani v druge informacijske rešitve. Bolj poznane rešitve, ki so na voljo na trgu, so SAP SCM in Oracle E-Business Suite.

2.2.3 ERP - Poslovni informacijski sistemi

Sistem ERP (Enterprise Resource Planning) je orodje, ki pokrije vsa področja poslovnega sveta - od naročila, nabave, proizvodnje, prodaje ter marketinga. Zaradi kompleksnosti sistema ERP večina podjetij, ki take sisteme uporablja, navadno ne implementira vseh modulov, ki bi pokrivali vsa področja poslovanja. Najbolj pogosto uporabljeni moduli sistemov ERP so pregled zalog, sledenje naročilom, servisne storitve, finance in viri.

Primeri bolje poznanih ERP sistemov so: SAP, Microsoft Navision, Oracle E-Business Suite.

2.3 Metodologije razvoja, standardi in načela dobre prakse

Razlogi zakaj se podjetja odločajo za uporabo računalnikov v najširšem pomenu (od žepnih računalnikov do robotizirane proizvodnje) v svoje poslovne procese so različni. Stroji rešujejo probleme, ki so za človeka fizično zahtevni, monotoni ali pa zahtevajo veliko večjo natančnost, kot jo zmore človek. Stroji lahko dosežejo območja, ki so človeku nedosegljiva in delujejo v človeku sovražnih okoljih.

Dandanes že skoraj vsak, pa naj bo še tako preprost stroj, kot je hladilnik, vključuje komponento z mehko ožičeno logiko, katere vedenje se da določati s programsko kodo. Možnost programiranja strojev omogoča njihovo širšo uporabnost. Programski sistemi, tudi tako osnovnih naprav, so lahko zelo zahtevni in razvoj lahko traja več let.

Kompleksnost sistema, slabo načrtovanje, precenjevanje zmožnosti razvijalcev in drugi faktorji v veliki večini primerov povzročijo neuspešnost programskih projektov. Fitzgerald (citirano v [4]) poroča, da obstajajo dokazi krize v razvoju informacijskih sistemov in pravi da:

- večina programerskih projektov traja od 18 mesecev do 5 let,
- 68% projektov preseže rok izvedbe,
- 65% jih preseže proračun,
- 75% produktov je razvitih drugače kot so bili zasnovani,
- 35% podjetij ima vsaj en izgubljen projekt.

Razvoj informacijskih sistemov je očitno kompleksen proces, zato zahteva profesionalen pristop.

Podjetja, ki se ukvarjajo z razvojem informacijskih sistemov, že od samega začetka spreminjajo mnenja glede količine dokumentacije, ki naj bi nastala pred, med in po razvoju. Nekatera podjetja se bolj odločajo za planske metodologije, druga za bolj agilne. Na prvi pogled se zdi, da so agilne metodologije na splošno bolj primerne, vendar se pri zelo velikih projektih pokaže, da temu ni čisto tako.

Da je planiranje pomembno, pokaže dejstvo, da velik delež mladih podjetij propade zaradi nevarnosti, ki prežijo na njih. Nevarnosti lahko nastopijo zaradi slabega poznavanja problemske domene, s katero se soočajo, nepoznavanje tehnologij, slabe organizacije in zaradi drugih razlogov. S pravilnim

planiranjem lahko predvidimo težave in se nanje pripravimo ali se jim celo izognemo; s preveč planiranja pa se zna zgoditi, da večino časa porabimo za načrtovanje in analizo, za implementacijo in testiranje pa ostane zelo malo časa.

Metodologija (Avison in Fitzgerald, 2002) [1] je:

Skupek postopkov, tehnik, orodij in predlogov dokumentacije, ki pomagajo sistemskim inženirjem pri razvoju informacijskega sistema. Metodologijo sestavljajo stopnje, ki so razdeljene na pod-stopnje. Podrobnejša razdelitev omogoča načrtovalcem lažje načrtovanje, upravljanje, nadzor ter ocenjevanje projektov razvoja informacijskih sistemov.

2.3.1 Življenjski cikel programske opreme

Kot pri vsaki stvari na svetu ima tudi programska oprema svoj življenjski cikel. Informacijski sistem se po navadi rodi iz neke dejanske potrebe posameznika ali skupine ljudi. Potreba porodi idejo, ki jo nato lahko inženirska stroka z uporabo znanj in orodij uresniči. Razvoj programske opreme lahko v grobem razdelimo na notranji ter zunanji razvoj (za zunanjega naročnika).

Notranji razvoj programske opreme vključuje razvoj orodij in knjižnic in je lahko posledica generalizacije modulov namenjenih zunanjemu razvoju ali namenskemu razvoju ponovno uporabljivih programskih modulov.

Zunanji razvoj je tipično namenjen zunanjemu naročniku. Ta je lahko znan v naprej ali pa je znana le ciljna skupina potrošnikov, katerim je proizvod namenjen.

Razlika je v financiranju - pri zunanjemu razvoju se navadno ukvarjamo z reševanjem problema, kjer je tveganje za neuspeh manjše, to pomeni manjše stroške in posledično večjo možnost zaslužka. Notranji razvoj pa je namenjen razvoju komponent za katere *predvidevamo*, da nam bojo pri potencialnem naslednjem naročilu olajšala delo, znižale ceno razvoja ter s tem povečale konkurenčno prednost na trgu.

Življenjski cikel programske opreme bi v grobem lahko razdelili na naslednje faze:

1. poizvedba,

2. zajem zahtev,

3. analiza,
4. načrtovanje,
5. implementacija,
6. testiranje,
7. prenos v končno okolje,
8. vzdrževanje.

Ne glede na izbrano metodologijo, se zgoraj naštete faze ponekod pojavijo med samim razvojem, čeprav mogoče pri nekaterih metodologijah razvoja niso očitne.

Razvojne skupine že desetletja poizkušajo najti optimalen, predvidljiv in ponovljiv postopek razvoja, s katerim bi dosegli večjo produktivnost in kvaliteto programske opreme. Nekateri poizkušajo sistematizirati ter formalizirati postopek pisanja kode, drugi se oprijemajo bolj projektnih pristopov. Brez nadzora zlahka prekoračimo proračun ter dogovorjene roke za predajo izdelka.

Obstaja nekaj značilnih vzorcev razvoja informacijskih sistemov, ki jih bomo opisali v naslednjih razdelkih.

2.3.2 Kaskadni način razvoja (Waterfall life-cycle)

Razvoj informacijskega sistema je razdeljen v formalne faze, kjer so izhodni rezultati prejšnje faze vhodni parametri naslednje faze. Dobavljiv izdelek je na dnu kaskade. Značilnost življenjskega cikla je da faze, ko je enkrat zaključena ne ponavljamo.

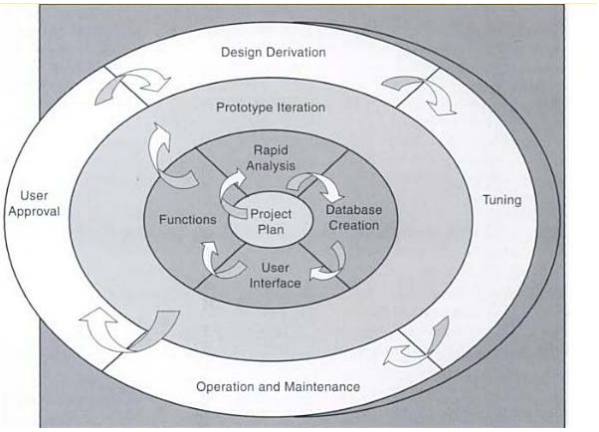
1. **Študija izvedljivosti:** preučimo obširnost projekta, namen novega sistema in vpliv na podjetje,
2. **Zajem zahtev**
3. **Načrtovanje:** prenos zahtev v programske specifikacije,
4. **Kodiranje:** razvijalci glede na specifikacije programskega sistema napišejo in dostavijo kodo,
5. **Testiranje:** po navadi vršimo v treh korakih (testiranje programskih modulov (Unit tests), sistemski testi, test obremenitve).

6. Prenos v delovanje

7. Vzdrževanje: odpravljamo nepravilnosti in beležimo možnosti izboljšav.

Kaskadni način razvoja izvira iz inženirskih okolij, kjer je veliko manj človeške interakcije in je primeren za izdelavo računalniških vezij ter konstrukcijo stavb. Velikokrat točne zahteve informacijskega sistema še niso znane ob sami zasnovi. Pri prenosu v realno okolje se lahko pokaže, da smo ves čas razvijali napačen IS. Napake v zasnovi se lahko pokažejo že med samim razvojem, vendar zaradi zasnove kaskadnega načina razvoja ne moremo prekiniti faze - sistem je tako obsojen na propad ali nezadovoljstvo končnih uporabnikov.

2.3.3 Evolucijski razvoj



Slika 2.1: Evolucijski življenjski cikel programske opreme.

Evolucijski pristop se zgleduje po naravi. Pri evolucijskemu pristopu sistem razvijemo kot prototip, ki ga prilagajamo in razvijamo glede na povratno informacijo, ki jo dobimo od naročnika. Primernost aplikacije je pri tem pristopu skoraj zagotovljena saj naročnik v razvoju sodeluje ves čas in ne samo na koncu, kot je bilo to pri kaskadnem načinu razvoja. Spremembe so vidne takoj in zelo hitro ugotovimo, da bi bil razvoj nekega modula na zastavljen način neprimeren. Slika 2.1 prikazuje potek projekta od projektnega plana (notranji krog), preko hitrih analiz, ustvarjanja podatkovnih zbirk, uporabniškega vmesnika in funkcionalnosti, do kreiranja delujočega prototipa. Prototip delujoče aplikacije pokažemo uporabniku in v primeru, da uporabnik še ni zadovoljen, cikel ponovimo. Ko je uporabnik zadovoljen posvetimo nekaj časa opti-

miziranju delovanja in prenesemo razvit sistem v končno okolje. Iteracije so po navadi dolge do tri mesece.

2.3.4 Agilne metodologije

Agilne metodologije razvoja programske opreme so skupina metod razvoja programske opreme, ki temeljijo na načelih agilnosti. Sestavlja jih niz najboljših inženirskih praks na področju vodenja projektov, samoorganizacijo in odgovornost. Agilne metodologije omogočajo hitro prilagajanje potrebam naročnika.

Najbolje poznani agilni metodologiji razvoja programske opreme sta *ekstremno programiranje* (angl. Extreme Programming) ali XP ter Scrum[1].

Metodologija XP [2] dovoljuje eliminacijo faz zajema zahtev, načrtovanja ter testiranja vključno s spremljajočo dokumentacijo. Te faze so vključene ves čas med procesom razvoja programske opreme. Iteracije so dolge en teden - znotraj ene iteracije je vključeno planiranje, načrtovanje, kodiranje ter testiranje. Programerji ocenjujejo količino dela glede na trenutno hitrost razvoja ter sodelujejo z naročniki, ki določajo prioritete funkcionalnosti. Rezultat vsake iteracije mora biti izvršljiv izdelek.

Metodologija XP je najbolj primerna za razvoj naše aplikacije, saj priporoča sprotne priprave in načrtovanje funkcionalnosti, implementacija in testiranje pa sledita v časovnem obdobju dneva ali dveh. To nam omogoča, da ideje realiziramo, ko so sveže in najbolje dodelane.

2.4 Socialna omrežja

Socialno omrežje je socialna struktura sestavljena iz posameznikov, ki predstavljajo vozlišča. Vozlišča so med seboj povezana preko relacij kot so prijateljstvo, sorodstvo, skupni interesi, finančne povezave in podobnih.

Modeliranje socialnih omrežij v informacijski tehnologiji ni velika novost. Poskusi spletnih socialnih omrežij segajo v dobo omrežij Usenet, Arpanet, Listserv ter BBS. Prve aplikacije socialnih omrežij je videti v omrežjih America Online, Prodigy in CompuServer. Socializiranje širše publike spletu se je začelo s skupnostnimi kot so Geocities (1994) in Tripod.com (1995). Podprta oblika komunikacije je bila izmenjava sporočil in povezovanje posameznikov v skupinske klepetalnice. Ponudniki storitev so uporabnikom dali na voljo spletni prostor, kjer so lahko podali svoje osebne informacije. Za lažje objavl-

[1]Scrum je starejša iterativna, inkrementalna agilna metodologija.

janje je bilo na voljo mnogo brezplačnih orodij, kar je omogočalo uporabo tudi računalniško manj vešči publiki.

V poznih devetdesetih so profili uporabnikov postali osrednja funkcionalnost ponudnikov storitev. Uporabnikom je bilo omogočeno generiranje seznama prijateljev ter iskanje oseb po skupnih interesih. Konec desetletja so bila socialna omrežja v polnem razmahu. Večji ponudniki storitev so bili SixDegrees.com (1997), Makeoutclub (2000), Friendster (2002), MySpace in LinkedIn (2003). Socialno omrežje MySpace je leta 2005 imelo večjo obiskanost kot spletni iskalnik Google. Leta 2004 je nastal Facebook, ki je trenutno največje spletno socialno omrežje.

Spletna socialna omrežja omogočajo povezovanje oseb s podobnimi interesi brez omejitev, kot jih prinaša realni svet. Zaradi svoje velikosti in zanimivosti relacij so postale zanimive za raziskovalce ter poslovni svet.

Omrežja kot je Facebook nudijo možnost integracije z zunanjimi aplikacijami preko API vmesnikov. Ti omogočajo objavljanje in izvoz vsebin v standardnih formatih.

2.4.1 Facebook API

Facebook API omogoča dostop do podatkov in objavljanje vsebin, kot so: osebne strani, dogodki, skupine, aplikacije, statusna sporočila, fotografije, videi, uporabniški zapiski in lokacijskih podatkov uporabnikov socialnega omrežja. API, poleg še naštetih funkcionalnosti, omogoča uporabo prijavnih podatkov iz socialnega omrežja v namene overovitve identitete uporabnika v zunanjih aplikacijah. Prednost takšnega načina prijave je v enostavnosti postopka registracije in prijave, saj odstranimo potrebo po vnašanju podatkov s strani uporabnika. Pravilnost vnesenih podatkov lahko preverimo pri ponudniku storitev.

2.5 Tehnologije

2.5.1 Na strani strežnika

Na strani strežnika smo izbrali širše uporabljene odprtokodne rešitve na področju spletnih strežnikov in podatkovnih baz. Kot spletna strežnika sta bila izbrana spletni strežnik Apache v kombinaciji s programskim jezikom PHP ter Apache Tomcat v kombinaciji s programskim jezikom Java, kot sistem za upravljanje s podatkovno bazo smo izbrali MySQL.

Podatkovna baza MySQL

Podatki so shranjeni v relacijskih tabelah v sistemih za upravljanje s podatkovno bazo (SUPB[2]) MySQL, ki ga trenutno vzdržuje in razvija podjetje Oracle. Hramba podatkov v sistemih SUPB omogoča standardiziran dostop do podatkov preko SQL poizvedb. MySQL omogoča poln nadzor (ACID[3]) nad podatki preko standardnega vmesnika. Uporaba podatkovnih baz doda nivo abstrakcije in omogoča prenosljivost napisane kode med različnimi sistemi, pri tem pa ne izgubljamo na zmogljivost ter odzivnosti.

Spletni strežnik Apache

Spletni strežnik je uporabljen Apache HTTP server (httpd), ki je eden izmed najbolj poznanih projektov Apache fundacije. V našem sistemu je spletni strežnik edina komponenta sistema, ki je dostopna preko svetovnega spleta (zunanje omrežje). Prednosti spletnih aplikacij so v prenosljivosti in dostopnosti. Uporabniki lahko do sistema dostopajo povsod, kjer imajo dostop do interneta, naj bo to osebni računalnik ali mobilni telefon.

Spletni strežnik Apache Tomcat in JasperSoft iReport

Spletni strežnik Apache Tomcat je namenjen izvajanju javanske kode in ponujanju rezultatov preko vgrajenega spletnega strežnika. Razlog za uvedbo še enega strežnika je v programskem jeziku. Programski jezik, uporabljen za razvoj aplikacij na Tomcat spletnem strežniku je Java.

Razlog za uporabo programskega jezika Java je izdelava poročil, za izdelavo katerih smo uporabili orodje iReport podjetja JasperSoft. Orodje ponuja razvojno okolje za grafično oblikovanje poročil in javanske knjižnice za izdelavo poročil v formatih PDF, DOC, DOCX, XLS in XLSX. Podatke je mogoče napolniti iz različnih virov - dva najbolj uporabna sta XML ter JDBC[4]. Pri razvoju spletne evidence smo uporabili direkten dostop preko JDBC.

Aplikacija poleg poročil preko Apache Tomcat strežnika izvaja pošiljanje e-pošte. Razlog za uporabo Jave je v možnosti boljšega nadzora nad pošiljanjem sporočil. Vgrajeni mehanizem Tomcat strežnika za beleženje dnevnikov omogoča zelo dobro zaznavanje nepravilnosti in proženje alarmov.

---

[2]SUPB - originalna kratica DBMS - Database Management System

[3]ACID (atomicity - atomarnost podatka pomeni, da imamo podatke v celoti ali pa nimamo zapisa, consistency - skladnost podatkov, isolation - izolacijo transakcij, durability - sposobnost okrevanja po podatkovnih nesrečah)

[4]omogoča direktno povezavo s podatkovno bazo

Windows razporejevalnik opravil

Sistemska storitev operacijskega sistema Microsoft Windows, ki nam omogoča poganjanje želenih programov v določenih časovnih okvirih. V našem primeru je storitev uporabljana za verifikacijo podatkov v zalednem sistemu, za osveževanje podatkov o trenutnem vremenu in osveževanje podatkov o vremenskih napovedih.

Programski jezik PHP

Programski jezik PHP (ime izvira iz Personal Home Page, sedaj Hypertext Preprocessor) je razširjen odprtokodni skriptni programski jezik razvit predvsem za razvoj dinamičnih spletnih strani [7]. Programski jezik odlikuje bogat nabor vgrajenih ukazov ter enostavno razširjanje funkcionalnosti. Skupina PHP Group programski jezik stalno razvija in s tem zagotavlja stabilnost delovanja ter ponuja dobro podporo skupnosti.

Pri sodobnemu programskemu jeziku se pojavlja problem slabo dokumentiranih vmesnikov, vendar pri PHP temu ni tako. Programski jezik se po namestitvi popolno integrira s spletnim strežnikom Apache, tako da je razvoj aplikacij zelo enostaven. Razvijalci so tolmač jezika prevedli za več operacijskih sistemov, kljub temu pa ohranili skoraj popolno prenosljivost napisane kode. Kodo lahko tako poganjamo na razvojem okolju Windows ter (skoraj) brez potrebnih prilagoditev prenesemo na produkcijsko okolje Linux.

PHP je pripadnik skriptnih jezikov, zato ima kar nekaj slabosti. Prva slaba lastnost skriptnih jezikov je počasnost zaradi sprotnega interpretiranja ukazov, vendar se izkaže da v primeru PHP počasnost ne povzroča težav niti pri zelo velikih obremenitvah. Primer zelo velike implementacije aplikacije, katere spletni del temelji na jeziku PHP, je dobro poznano socialno omrežje Facebook. Po njihovem mnenju je programski jezik "lahek za učenje, enostaven za pisanje, enostaven za branje in enostaven za razhroščevanje"[5].

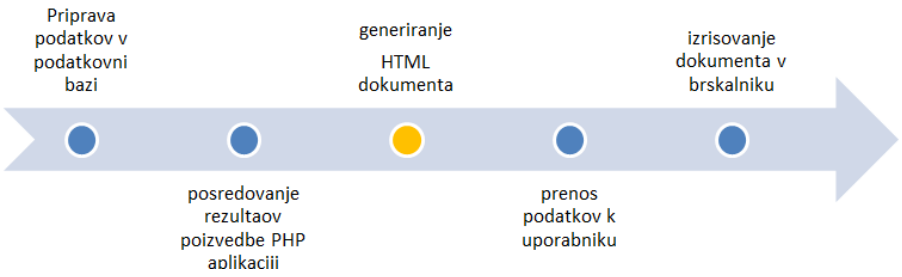
2.5.2 Na strani uporabnika

Na strani uporabnika so bile uporabljene sodobne spletne tehnologije, ki od uporabnika zahtevajo le uporabo novejšega spletnega brskalnika brez nameščanja vtičnikov ali druge programske opreme. Uporabljena je bila kombinacija tehnologij HTML, CSS ter Javascript. HTML in CSS skupaj določata videz aplikacije, uporaba Javascript programskega jezika pa doda interaktivnost.

[5]Citat s spletne strani podjetja Facebook, Inc.: Developer blog

S prenosom dela procesiranja podatkov, predvsem obdelave prikaza večje količine podatkov na odjemalca minimiziramo prenos podatkov in posledično zmanjšamo obremenitev strežnika. Uporaba knjižnice Javascipt[6] jQuery[7] ali podobne nam omogoča enostavno manipulacijo z DOM strukturo HTML dokumenta in asinhrono poizvedovanje na spletni strežnik.

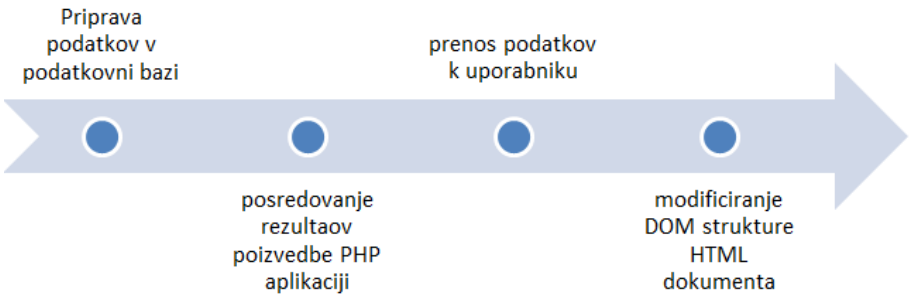
Poenostavljena slika 2.2 predstavlja klasično izgradnjo dokumenta od poizvedbe na podatkovni bazi do izrisa elementov na ekran uporabnika. Koraki, skozi katere mora priti dinamično generirana spletna vsebina, so: poizvedba v podatkovni bazi (model), priprava podatkov v aplikaciji PHP (kontrolnik) ter generiranje HTML kode (pogled). Generirani dokument se prenese do uporabnika, kjer brskalnik poskrbi za izris. Slika 2.3 predstavlja dinamično izgradnjo vsebine brez koraka izgradnje HTML dokumenta. Korak se nadomesti s serializacijo notranjih struktur (poljubnega objekta) jezika PHP v obliko JSON. Pretvorba se izvede v krajšem času kot sestavljanje HTML kode, na primer: sestavljanje seznama razpoložljivih terminov za prenos k uporabniku. JSON dokument se nato asinhrono prenese k uporabniku, kjer koda Javascript poskrbi za modifikacijo DOM strukture spletne strani ter tako izdela izpis. Kombinacija med seboj povezanih spletnih razvojnih tehnologij opisanih zgoraj se imenuje AJAX.



Slika 2.2: Klasična izgradnja spletne strani.

[6]Objektno orientiran, dinamičen, šibko tipiziran programski jezik. Javascript je primarno uporabljen kot skriptni programski jezik vgrajen v večino spletnih brskalnikov. Namenjen je izboljševanju spletnih uporabniških vmesnikov in za povečanje dinamičnosti spletnih strani.

[7]Priljubljena odprtokodna knjižnica za Javascript programski jezik [8]. Druge bolje poznane knjižnice so Dojo, MooTools, Prototype ter druge.



Slika 2.3: Izgradnja dokumenta v formatu JSON.

Primer kode pogleda za idelavo HTML tabele uporabnikov v jeziku PHP:

```
<table>
<?php
$uporabnikController = new UporabnikController();
$seznam = $uporabnikController->findAll();
for ($seznam as $u) {
  echo '\n<tr>
    <td>' . $u['id'] . '</td>
    <td>' . $u['ime'] . '</td>
    <td>' . $u['priimek'] . '</td>
  </tr>';
}
?>
</table>
```

Primer kode pogleda brez z generiranjem JSON objekta:

```
<?php
$uporabnikController = new UporabnikController();
echo json_encode($uporabnikController->findAll());
?>
```

Zgornja koda generira spodnji JSON objekt:

```
[{
  "id": 1,
  "ime": "Miha",  "priimek": "Štrumbelj"
}]
```

V prvem primeru smo z lepljenjem niza dobili pripravljeno tabelo, ki jo zna izrisati vsak brskalnik. V drugem primeru je rezultat skripte izpisan v JSON podatkovni obliki.

Uporabniški del aplikacije izvede asinhrono zahtevo na strežnik z naslednjim zaporedjem ukazov:

```
1 jQuery.post( "api.php?action=seznamObvestil", {}, function(data,
    textStatus, XMLHttpRequest) {
2   if (textStatus=="success")
3     alert("podatki uspešno naloženi");
4   }, "json" );
```

Programski jezik PHP z vgrajenim mehanizmom poskrbi za serializacijo objektov v JSON format v konstantnem času.

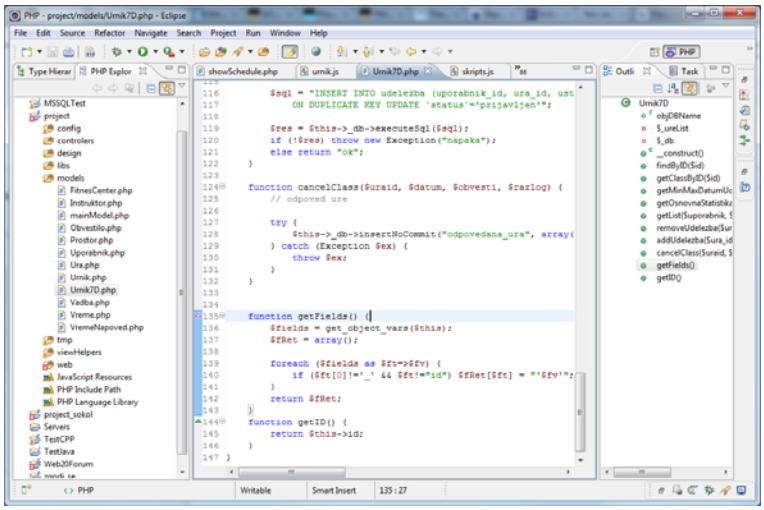
2.6 Razvojno okolje

Hiter razvoj informacijskih sistemov je močno odvisen od uporabljenih orodij. Uporaba orodij z nezadostno podporo tehnologiji ali slabo poznanih orodij lahko razvoj oteži ter posledično podaljša.

2.6.1 Razvojno okolje Eclipse

Pri razvoju aplikacije za vodenje evidenc so bila uporabljena široko uporabljena odprtokodna orodja z dobro podporo in možnostjo razširitev. Tako orodje je na primer razvojno okolje Eclipse (slika 2.4).

Orodje ponuja pregleden uporabniški vmesnik z možnostjo prilagoditve za vsakega posameznika. Orodje je zelo zmogljivo ter že ob namestitvi omogoča programiranje v najbolje razširjenih programskih jezikih (za prenos so na voljo verzije za: Javo, PHP, C/C++). Z dodatnimi vtičniki ga lahko poljubno razširjamo. Pri tem projektu smo Eclipse uporabili za razvoj in modeliranje UML (vtičnik UMLet).



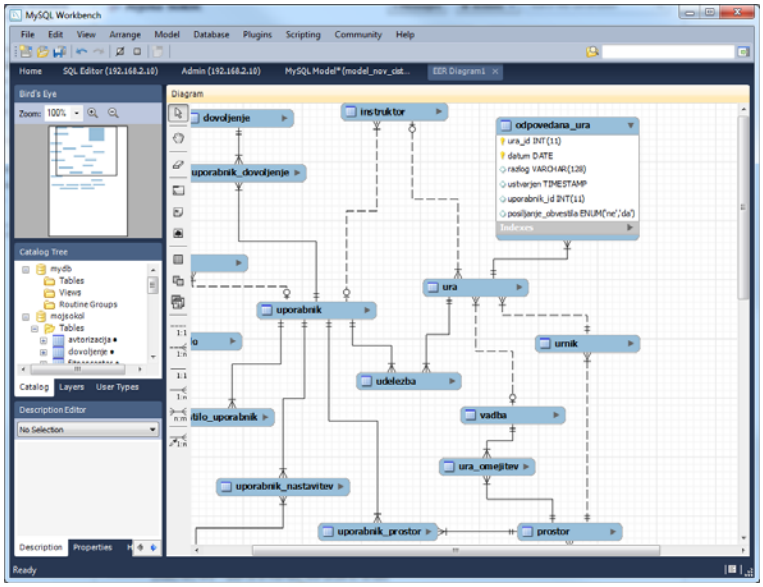
Slika 2.4: Razvojno okolje Eclipse.

2.6.2 MySQL Workbench

Razvojno okolje Eclipse nam mogoče načrtovanje podatkovnega modela, vendar te zmogljivosti nismo uporabili. Razlog je v obstoju boljših orodij za

podatkovno bazo MySQL. To je razvojno okolje MySQL Workbench podjetja Sun (sedaj pod okriljem podjetja Oracle). MySQL Workbench (na sliki 2.5) podpira poleg generiranja in poganjanja stavkov SQL DDL[8] in poizvedb DML[9] tudi generiranje podatkovnega modela iz obstoječe podatkovne zbirke (povratno inženirstvo).

V primeru, da smo pri načrtovanju uporabljali InnoDB[10] način zapisa tabel in za relacije uporabljali tuje ključe zna orodje grafično prikazati vse povezave med entitetami.



Slika 2.5: Razvojno okolje MySQL Workbench.

[8]DDL - Data Definition Language - stavki programskega jezika SQL namenjeni manipulaciji s shemo podatkov.

[9]DML - Data Manipulation Language - strukture SQL jezika namenjene poizvedovanju nad podatki.

[10]InnoDB je eden izmed tipov tabel pri podatkovni bazi MySQL. Poleg InnoDB ima MySQL sistem za upravljanje s podatkovnimi bazami še: MyISAM (zelo hiter a brez podpore transakcij), Memory (podatki shranjeni samo v delovnemu pomnilniku strežnika), NDB (namenjen shranjevanju podatkov v gruči - Clustru) ter drugi.

Poglavje 3

Razvoj portala fitnes centra za delo s strankami

3.1 O podjetju

Fitnes X je združenje podjetij s področja športa. V svojih prodajnih programih nudijo vse za šport: od proizvodnje fitnes naprav, izdajanja športnih publikacij, uvoza in distribucije športne prehrane, nudenja storitev fitnesa in aerobike do izdajanja lastne popustniško-zavarovalniške športne kartice. Ob prodaji nudijo tudi poprodajne aktivnosti, ki obsegajo uvajanje in izobraževanje ter ponujajo podporo za svoje storitve.

Podjetje je sestavljeno iz naslednjih poslovnih subjektov:

1. A: fitnes center z aerobiko, centrom borilnih veščin in pokritim plavalnim bazenom
2. B: wellness studio, ki vključuje fitnes, aerobiko, savne, solarij, masaže,
3. C: wellness studio, ki vključuje fitnes, aerobiko, savne, solarij, masaže,
4. X: proizvodnja fitnes naprav in športne opreme, servis

Vodenje storitve fitnes centra in vodenih vadb sestavlja velik del njihovega poslovnega modela, zato so se v podjetju odločili svoje poslovanje informacijsko bolje podpreti.

Športno udejstvovanje pod nadzorom usposobljenega kadra dosega vse večje zanimanje. V podjetju Fitnes X se tega še kako dobro zavedajo, saj se vsak dan ukvarjajo z problemom porazdelitve kapacitet.

Podjetje ima urejen sistem evidence v papirnati obliki. Evidenco vodi posamezni fitnes center v obliki lista papirja, ki se nahaja na recepciji centra.

Problemi s katerimi se soočajo pri sedanjem sistemu so:

- uporabnik, ki si želi rezervirati mesto na uri vodene vadbe, mora določen čas prej priti v podjetje ter se vpisati na seznam; če predhodni obisk ni mogoč, podjetje omogoča prijavo preko telefona,
- v primeru, da je na sezamu več prijavljenih, kot je kapaciteta dvorane, se uporabnik ne more več prijaviti,
- v primeru, da ura vodene vadbe odpade, je potrebno vsakega uporabnika poklicati ter ga obvestiti o dogodku,
- uporabnik, ki se želi odjaviti od določene ure mora ponovno izvršiti obisk ali klic izbranega fitnes centra.

Podjetju je v interesu ponujati storitve, prilagojene čim večjemu številu strank. Današnji hiter tempo življenja narekuje stalne spremembe življenjskih navad. Vsak ponudnik storitev, namenjenih množicam, se mora čim bolj prilagajati posameznemu uporabniku.

Cilj projekta uvedbe elektronskega vodenja evidenc je predvsem olajšati delo recepcije fitnes centra, da se lahko namesto z vodenjem evidenc ukvarjajo z zagotavljanjem večje kakovosti storitev, zaradi katerih se uporabniki vračajo. Sistem elektronskih evidenc je prijazen uporabnikom, saj jih razbremeni razmišljanja o sami logistiki prijave - sodoben človek namreč nima časa stalno obiskovati fitnes centra, samo zato da bi se lahko prijavil za nekaj dni vnaprej na priljubljeno uro vodene vadbe.

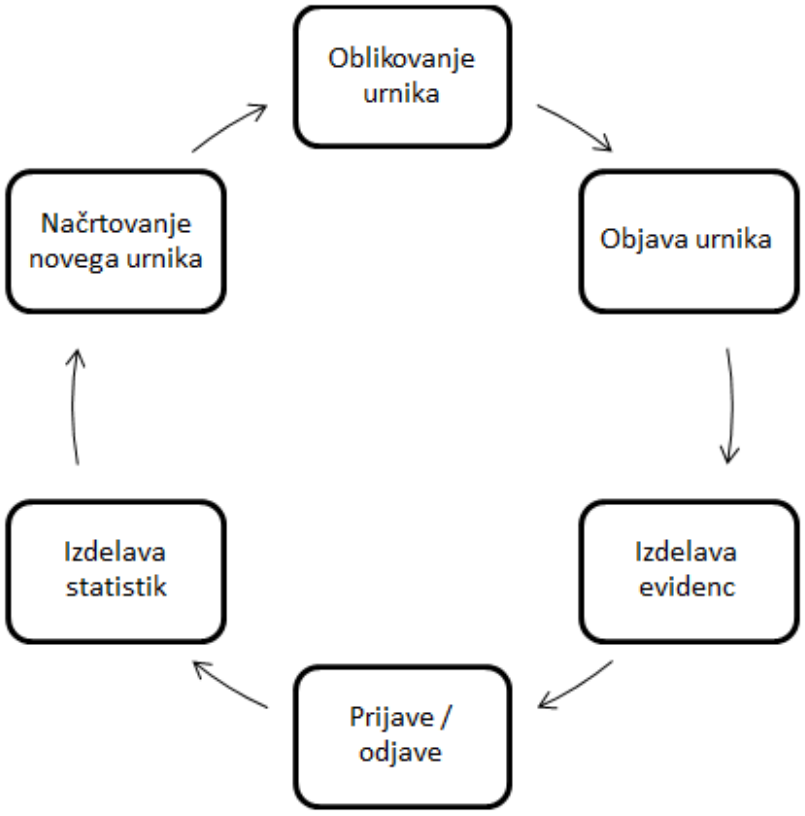
Med trajanjem projekta smo opravili intervjuje z vodstvom posameznih centrov, inštruktorji vodenih vadb in vadečimi in prišli do naslednjih ugotovitev:

- vodje ur aerobike niso zadovoljni z nizkim številom uporabnikov, ki predhodno najavijo udeležbo.
- izkušeni inštruktorji trdijo, da je za kvalitetno vadbo potrebno optimalno število udeležencev, trenuten sistem jim ne omogoča nadzora števila udeležencev,

- evidence obiskov trenutno niso najbolj zanesljive, kar lahko povzroči napačne odločitve glede ukinjanja oziroma uvajanja novih terminov,
- podjetje od uporabnika zahteva, da v svoje planiranje vključi še rekreacijo.

Cilj diplomske naloge je bil zasnovati informacijski sistem, ki bi avtomatiziral vodenje evidence ter izdelavo poročil.

3.2 Opis postopka vodenja evidenc



Slika 3.1: Življenjski cikel urnika od objave, do načrtovanja naslednje objave.

Skupina Fitnes X ima trenutno dobro utečen postopek oblikovanja urnikov. Celoten cikel poteka je sestavljen iz naslednjih korakov:

Oblikovanje urnika: Vodja aerobike sestavi urnik. Po navadi to stori tako, da obstoječi urnik prilagodi glede na rezultate analize obiska iz časa prejšnjih veljavnih urnikov.

Objava urnika: Vodja aerobike pošlje narejen urnik vsem zaposlenim, ki nato sporočijo svoje pripombe. Sestavljavec urnika popravi urnik v skladu s komentarji zaposlenih. Končni urnik se pošlje zunanjemu izvajalcu, ki poskrbi za objavo na spletni strani podjetja. Tiskane verzije urnikov dostavijo na recepcijo fitnes centra.

Izdelava evidence obiska : Iz urnika se za vsak teden izda prazna evidenca za prijave na ure vodenih vadb.

Prijave/odjave: Uporabniki se prijavijo tako, da za želeni dan, uro, dvorano in vadbo na seznam napišejo svoje ime in priimek - če želijo biti obveščeni o spremembah, morajo zraven napisati še svoje kontaktne podatke.

Izdelava statistik: Iz evidence prijav vodja fitnes centra oblikuje statistike, ki so mu v pomoč pri načrtovanju novih urnikov.

Načrtovanje urnika: Glede na statistike obiska preteklih kombinacij ure, vadbe, dneva in prostora se oblikujejo smernice za novi urnik.

Bralec lahko iz opisa sam zasluti kar nekaj potencialnih težav v trenutnem postopku. Prva težava je v kvaliteti zajetih podatkov, na podlagi katerih se oblikuje nov urnik. Težave z ne prijavljanjem rešujejo s štetjem udeležencev posamične vadbe.

Usklajevanje urnika lahko traja veliko časa, saj vsi inštruktorji niso vedno dosegljivi. Od zaključka oblikovanja urnika do same objave na spletni strani lahko ponovno, zaradi posredovanja zunanjega izvajalca, traja nekaj časa.

Uporabniki želijo biti vedno obveščeni o odpadlih vadbah. Da bi to dosegli, morajo ob vsaki prijavi napisati številko svojega telefona. Evidenca je vidna vsem obiskovalcem, kar bi lahko sprožilo vprašanja o zasebnosti. Naslednja težava je, da obiskovalci svoje prijave sporočajo preko telefona tik pred izvedbo posamične ure. Pogosto se zgodi, da je ura vodene vadbe zaradi potencialno premajhne udeležbe odpovedana, saj vodja centra določen čas pred izvedbo ure preveri evidenco in ukrepa glede na število prijavljenih.

Podjetje se je odločilo za uvedbo informacijskega sistema, s katerim bi odpravili pomanjkljivosti sedanjega sistema in izboljšali preglednost evidenc obiskov.

Ker točne zahteve ob ideji za uvedbo IS niso bile znane, zahtevana razen dokumentacije pa je bila nizka, smo se odločili za agilni pristop k razvoju.

3.3 Zajem zahtev

3.3.1 Funkcionalne zahteve

Informacijski sistem mora pokrivati vse faze postopka od objave urnika, do prijav na ure. Postopek prijave na ure vodenih vadb mora biti dosegljiv tudi preko spleta, tako da smo se odločili za spletno aplikacijo. Nekatere stranke fitnes centrov so računalniško manj usposobljene osebe, zato je bilo potrebno izdelati kar se da enostaven uporabniški vmesnik. Prav tako je bilo potrebno zelo poenostaviti postopke registracije ter same prijave v spletni vmesnik. Da bi bil postopek prijave bil kar se da prijazen, smo prijavo omogočili tudi preko najbolj razširjenega socialnega omrežja Facebook.

Zahtevane so bile naslednje funkcionalnosti:

1. **Registracija uporabnika**
   Ko se uporabnik registrira, mora sistem preveriti:
   - da se uporabnik s temi podatki predhodno še ni registriral,
   - da številka uporabnikove kartice obstaja v zaledni bazi,
   - če se uporabnik želi registrirati z računom Facebook, je potrebna povezava na strežnike Facebook ter preverjanje uporabniških podatkov. Uporabnik mora zaradi varnosti določiti še varnostno geslo, ki ga bo uporabljal za prijavo.
2. **Prijava uporabnika**
   Uporabniku se omogoči prijavo z:
   - uporabo računa Facebook,
   - uporabnikovega naslova elektronske pošte (tistega, ki ga je uporabil pri registraciji),
   - uporabo številke fitnes kartice, ki jo ima vsaka stranka skupine fitnes centra

   v kombinaciji z varnostnim geslom.
3. **Urejanje lastnega profila**
4. **Oblikovanje urnika**
   - kreiranje novega urnika,

- urejanje obstoječega urnika,
  - dodajanje vadbe,
  - odstranjevanje vadbe,
  - določanje inštruktorja na uri,
  - odstranjevanje inštruktorja na uri,
  - sprememba datuma začetka in konca veljavnosti,
  - sprožanje objave / prekinjanje objave urnika (omogoča izdelavo osnutka urnika)
- urejanje prekrivanja urnikov (vrstni red upoštevanja urnikov).

5. **Prijava na uro**
   uporabnik se lahko prijavi na uro pod pogoji:
   - da je ura razpisana in ni odpadla,
   - da so na določeni uri še prosta mesta.
6. **Odjava udeležbe**
7. **Odpoved ure zaradi določenega razloga**
   razlog se določi v polju pri odpovedi
8. **Urejanje seznama inštruktorjev**
   - dodajanje,
   - urejanje,
   - odstranjevanje (omogočeno, dokler se ne pojavlja v katerem koli kontekstu v aplikaciji).
9. **Urejanje seznama programov**
   - dodajanje,
   - urejanje,
   - odstranjevanje (omogočeno, dokler se ne pojavlja v katerem koli kontekstu v aplikaciji).
10. **Urejanje uporabnikov**
    - iskanje uporabnika,
    - prikaz profila uporabnika,

- urejanje polj profila,
- pregled prijav uporabnika,
- odjava od prijavljenih ur,
- odstranjevanje (omogočeno, dokler se ne pojavlja v katerem koli kontekstu v aplikaciji),
- urejanje dovoljenj,
- pregled dnevnika aktivnosti uporabnika - beležijo se akcije in napake, na katere uporabnik naleti.

11. **Urejanje seznama fitnes centrov**
12. **Urejanje prikaznih besedil**
    - urejanje predlog e-sporočil za avtomatsko obveščanje,
    - urejanje statičnih besedil.
13. **Obvestila**
    - kreiranje osnutka obvestila,
    - pošiljanje obvestil vsem uporabnikom,
    - pošiljanje obvestila določenim uporabnikom.

    Obvestilom se lahko določi veljavnost (od - do).
14. **Avtomatsko potrjevanje udeležbe - integracija z obstoječim sistemom**
15. **Modul za izdelavo statistik**

3.3.2 Nefunkcionalne zahteve

Varnost

Podjetje je zahtevalo visok nivo varnosti, zato se morajo podatki, ki jih vnesejo uporabniki, preveriti. Administrativni uporabniški vmesnik omogoča vpogled v profile vseh uporabnikov, zato mora biti močno zaščiten. Potrebno je beležiti vse akcije uporabnikov, tudi neuspešno izvedene zahteve, kar bo omogočalo odpravljanje slabih izkušenj s sistemom.

Sočasni dostop

Sistem mora biti več-uporabniški. Potrebno je upoštevati število strank podjetja, ki bodo aplikacijo uporabljale. Potrebno je zagotoviti, da ne bo prišlo do primera prijavljanje na vadbe, ki so že zasedene. Uporabnik mora v vsakem trenutku, ko uporablja aplikacijo, vedeti status prijave na vodeno vadbo in status razpoložljivih ur za prijave.

Dosegljivost sistema ter odzivnost

Zahteva se visoka dosegljivost sistema, saj drugače trpi zadovoljstvo uporabnikov. Študije kažejo, da so uporabniki pripravljeni na akcijo čakati največ 8 ± 2 sekund [10]. Iz pogovorov, ki smo jih opravili s strankami, pa je 8 sekund že nesprejemljiv odzivni čas. Potrebno bo zagotoviti odzivno strojno opremo ter optimizirati kodo ter sistemske nastavitve za kar najhitrejše izvajanje zahtev s strani uporabniškega vmesnika.

3.3.3 Identifikacija uporabnikov

Informacijski sistem bo uporabljalo veliko število uporabnikov. Glede nivoja dostopa pa ločimo le tri: administrator, receptor/inštruktor ter stranka. Izdelati je potrebno sistem za upravljanje privilegijev.

Administrator

Administrator je skupina uporabnikov z največjim nivojem pravic. Je edina skupina, ki ima omogočen dostop do vseh uporabniških profilov, modula za urejanje šifrantov ter modula za urejanje urnikov. Le administrator lahko drugim spreminja nivo privilegijev (sebi ne more, lahko pa drugemu administratorju). Administrator lahko podrobno določa privilegije uporabnikom.

Receptor / inštruktor

Receptor je predstavnik podjetja, ki ima največ stika s strankami. Receptorjeva delovna postaja je izpostavljena notranjim vdorom, tako da je potrebno poskrbeti, da ima uporabnik tega nivoja ravno pravšnjo količino privilegijev. Uporabniška skupina je namenjena vodenju evidenc vpisa za trenutni teden. Če receptor potrebuje dodatne privilegije, lahko za to zaprosi administratorja (vendar to ni del IS).

Stranka

Stranka je registriran uporabnik, ki mu je omogočen vpogled v lastne osebne podatke. Omogočene so funkcionalnosti prijave ter odjave na ure vodenih vadb. Sistem stranki ne sme omogočati vpogleda v osebne podatke drugih strank, prav tako ne sme dopuščati prijavljanja ter odjavljanja drugih uporabnikov.

3.4 Analiza projekta

3.4.1 Merila uspešnosti projekta

Projekt je uspešen, ko je stranka zadovoljna z rezultatom, in ko projekt prestane testiranje s strani uporabnikov z implementacijo vseh funkcionalnih in nefunkcionalnih zahtev. Zaradi narave procesa razvoja projekta je dokumentacija zgolj v pomoč razumevanju delovanja programske opreme in pripomoček za nadaljnji razvoj.

Programske komponente morajo biti razvite na način, ki omogoča enostavno dodajanje funkcionalnosti. Moduli naj bodo med seboj neodvisni in tako pripravljeni za ponovno uporabo. Uporabiti je potrebno dobro podprte odprtokodne tehnologije.

Kljub napredkom je internet še vedno nepredvidljiv. Spletna aplikacija mora poskrbeti, da se podatki res shranijo, v nasprotnem primeru pa obvestiti uporabnika ter dogodek zabeležiti.

Za vsakega uporabnika se mora beležiti dnevnik akcij (prijava v sistem, iskanje po uporabnikih, sledenje nepooblaščenih dostopov, prijave/odjave udeležbe).

3.4.2 Osnutek predloga rešitve

Izdela se interaktivna spletna aplikacija (RIA[1]) z uporabo odprtokodnih tehnologij, ki so podprte v večini brskalnikov brez posebnih vtičnikov.

Uporabnikom bo omogočen dostop preko svetovnega spleta. Za večjo enostavnost sistema se doda možnost uporabe uporabniških računov zunanjih ponudnikov upravljanja spletnih identitet (OAuth[2]).

3.4.3 Dogovor o poteku projekta

Za uspešen razvoj produkta je potrebno z naročnikom določiti: osnovni časovni plan razvoja, način sodelovanja (komunikacija) ter nivo izdelave dokumentacije. Določi se osebo v podjetju, ki je odgovorna za spremljanje postopka razvoja in ima dovolj informacij ter pooblastil za dostop do prostorov podjetja, kjer bo nameščena strojna oprema ter podatkov o količini ter tipu uporabnikov aplikacije.

Ker sistem vključuje integracijo z obstoječim sistemom (ta pa vsebuje občutljive podatke) je potreben podpis dogovora o ne-razkritju informacij (NDA).

3.4.4 Profil uporabnikov informacijskega sistema

Uporabniki sistema so stranke fitnes centrov. Zaradi boljšega poznavanja profila uporabnikov smo opravili intervjuje z naključnimi uporabniki. Zanimali so nas podatki o uporabi računalnika v vsakdanjem življenju, uporabi novejših tehnologij ter zaupanju v informacijsko tehnologijo.

Izkazalo se je da večina vprašanih uporablja računalnik vsak dan doma in na delu. Povprečna starost vprašanega je bila 28,41 let. Anketiranih je bilo 62,1% moških ter 37,9% žensk.

V vsakodnevnem življenju 58,6% anketirancev uporablja računalnik več kot 6 ur dnevno, 20,7% uporablja 3-6 ur dnevno, ostalih 12,2% do tri ure na dan. Vsi vprašani so uporabljali v internet povezane računalnike. Najbolj priljubljene storitve so:

1. **Prva skupina (skoraj vsi vprašani):**

[1]RIA - Rich Internet Application: Spletna aplikacija, ki ima glede interaktivnosti veliko podobnosti z namiznimi aplikacijami. Bolj znane tehnologije, ki omogočajo razvoj takšnih aplikacij so: AJAX, Adobe Flash, Java, Microsoft Silverlight in HTML5.

[2]OAuth - Open Authenication: Odprt protokol, ki omogoča varen in hkrati preprost ter standardiziran način avtentikacije tako za spletne kot tudi za namizne aplikacije.

- iskanje informacij (Google, Najdi.si, Bing),
- pregled novic (24ur.com, Siol.net, MMC RTV Slovenija),
- socialna omrežja (Facebook, Netlog),
- e-pošta (MS Outlook, Mozilla Thunderbird, Gmail),
- ogled večpredstavnostnih vsebin (Youtube).

2. **Druga skupina (50% - 75%):**
   - takojšnje sporočanje (Google chat, Windows Live Messenger, Skype),
   - spletni nakupi (Mimovrste.com, Enaa.com, Amazon.com),
   - poslovne aplikacije (Office orodja, ERP, CRM ter sistemi za upravljanje vsebin - CMS).
3. **Tretja skupina (najredkeje uporabljene storitve - pod 20%):**
   - rezervacije in nakup počitniških aranžmajev ter letalskih kart (booking.com, Easyjet, Ryanair),
   - naročanje dostave hrane.

Zgornje vprašanje je bilo namenjeno odkrivanju navad uporabnikov. Predvsem nas je zanimal uporabniški vmesnik, na katerega uporabniki naletijo, ko se srečajo s storitvijo.

Uporabniki imajo radi preproste uporabniške vmesnike z jasno razvidno navigacijo ter čim manj nivojsko strukturo vsebine. Če se osredotočimo na spletne strani, so bolj priljubljene tiste, ki imajo preprosto ureditev elementov ter očem prijazen in nevsiljiv barvni nabor.

Izsledke preproste ankete smo uporabili pri načrtovanju uporabniškega vmesnika informacijskega sistema. Videz spletne aplikacije naj omogoča načine interakcije, ki so jih uporabniki vajeni, videz pa naj bo konsistenten ter očem prijazen.

Uporabnikom storitev se je zdelo pomembno vprašanje varnosti ter dostopnosti storitev, saj aplikacije hranijo veliko osebnih podatkov. Pričakuje se neprekinjen dostop, odzivni časi pa morajo biti zelo kratki.

3.5 Analiza, načrtovanje in implementacija programske rešitve

Za razvoj informacijskega sistema, namenjenega vodenju evidenc prijav na ure vodenih vadb smo uporabili sodobne tehnike, ki jih priporoča industrija.

Pomembni aspekti pri razvoju informacijskega sistema so:

- uporaba odprtokodnih tehnologij,
- prenosljivost med sistemi,
- ponovna uporabljivost komponent,
- podpora uporabljenim knjižnicam.

3.5.1 Dogovori poimenovanja in tehnike kodiranja

V računalništvu je dogovor poimenovanja skupek pravil, ki določa poimenovanje programov, modulov, funkcij, spremenljivk, procedur, tipov ter ostalih delov programske opreme [3]. Določanje teh pravil pred začetkom razvoja je zelo pomemben, saj le konsistentno poimenovanje (v kombinaciji z zamikanjem blokov) zagotavlja berljivost kode, ta pa je zelo pomembna pri razhroščevanju in ponovni rabi komponent.

Od začetka razvoja računalništva se pojavljajo različna priporočila uporabe tehnik, ki naj bi povečale verjetnost uspeha pri razvoju IS. Nekaj teh tehnik oziroma priporočil smo uporabili pri razvoju našega sistema.

OOP - Objektno orientirano programiranje

Proceduralno programiranje zahteva veliko discipline ter upoštevanje navodil lepega programiranja. Kompleksnost kode se povečuje skladno s kompleksnostjo projekta. Pri večjih projektih se zelo hitro zgodi, da zaradi bližanja rokov programerji pozabimo na kvaliteto kode ter pričnemo s tako imenovano "špageti-kodo" [4].

Objektno usmerjeno programiranje je paradigma, ki kot programsko enoto uporablja objekte. Objekti so podatkovni tipi sestavljeni iz podatkovnih polj, ki skupaj z metodami objekta omogočajo interakcijo z drugimi objekti. Tehnika posnema modeliranje realnih objektov. Objektno usmerjeno programiranje omogoča lažjo abstrakcijo problema, enkapsulacijo, polimorfizem in dedovanje. Uporaba naprednih konstruktov programske kode lahko zmanjša kompleksnost.

[3] Naming convention (programming) - dogovor o poimenovanju

[4] Spaghetti code - slengizem, ki označuje kodo z kompleksno strukturo. Tako oblikovano programsko kodo je težko brati. Tipičen primer je uporaba GOTO kontrolnega stavka.

MVC - Model-View-Controller paradigma

Paradigma MVC (Model-Pogled-Krmilnik) je tehnika organizacije programske kode, popularna predvsem pri spletnih projektih. Uporablja se jo tudi pri razvoju klasičnih informacijskih sistemov. Uporabo tehnike nam poenostavi kasnejše vzdrževanje, saj sta datotečna organizacija in napisana koda bolj pregledni, kar omogoča delo več programerjev na istem projektu. Tehnika je objektno orientirana in zaradi svoje narave ni omejena le na en programski jezik oziroma okolje kjer se izvaja.

Kot pove sam naziv, arhitektura programsko kodo logično razdeli na tri med seboj neodvisne komponente:

Model (model) - predstavlja tisti del aplikacije, ki skrbi za zapis ter branje podatkov iz podatkovne zbirke. Skrbi za realizacijo poslovnih pravil in logike. Model je po navadi preprost razred, ki predstavlja logično entiteto projekta. V našem primeru bi to bil "Urnik". Razred po potrebi vsebuje metode za osnovne operacije življenjskega cikla objekta (CRUD[5]): stvaritev, pregled, ažuriranje ter brisanje entitete.

View (pogled) - nivo abstrakcije skrbi za komunikacijo med aplikacijo ter odjemalcem. Ločitev tega nivoja je primerna za oblikovanje predstavitve podatkov za različne odjemalce. Tipična spletna aplikacija ima spletni vmesnik (HTML), poleg tega omogoča izvoz podatkov tudi v drugih oblikah (RSS/ATOM[6], JSON[7]). Aplikacija bi lahko podpirala tudi povezavo z zalednim sistemom, ki za komunikacijo uporablja SOAP[8].

Controller (kontrolnik) - komponenta, ki omogoča dodajanje nivoja med modelom in pogledom. Uvedba *kontrolnikov* omogoča uvedbe dodatne logike med poglede ter modele. Nekatere operacije zahtevajo interakcijo z več modeli: na primer brisanje uporabnika (brisati - oziroma označiti za brisane - je potrebno vse entitete, ki so v relaciji z uporabnikom).

[5]CRUD - akronim: Create (stvaritev), Read (branje), Update (ažuriranje), Delete (brisanje). Predstavlja osnovne operacije nad objektom.

[6]RSS (oziroma Really Simple Syndication) je družina formatov, primernih za splet. Gre za XML zapis podatkov, ki se dosti spreminjajo. Primer uporabe RSS: blog, spletne novice

[7]JSON (Javascript Object Notation) [9] je odprta tekstovna standardna oblika zapisa. JSON je derivat Javascript programskega jezika za predstavitev programskih struktur ter asociativnih polj podatkov.

[8]SOAP (Simple Object Access Protocol) - standardiziran protokol za izmenjavo strukturiranih podatkov med (različnimi) sistemi. Za izmenjavo se uporablja strukturiran zapis XML, katerega obliko narekuje dokument WSDL. WSDL (Web Services Description Language) je znova dokument XML v katerem je zapisan vmesnik s strukturo podatkov.

Struktura map evidence

Datoteke so urejene po mapah z namenom organizacije različnih modulov. Držimo se pravila da so vsi spremenljivi parametri aplikacije zapisani v datoteki z imenom, ki nakazuje na vsebino nastavitev (npr.: PageSettings.php) ali pa - v primeru, da so to uporabniške nastavitve - v podatkovni bazi.

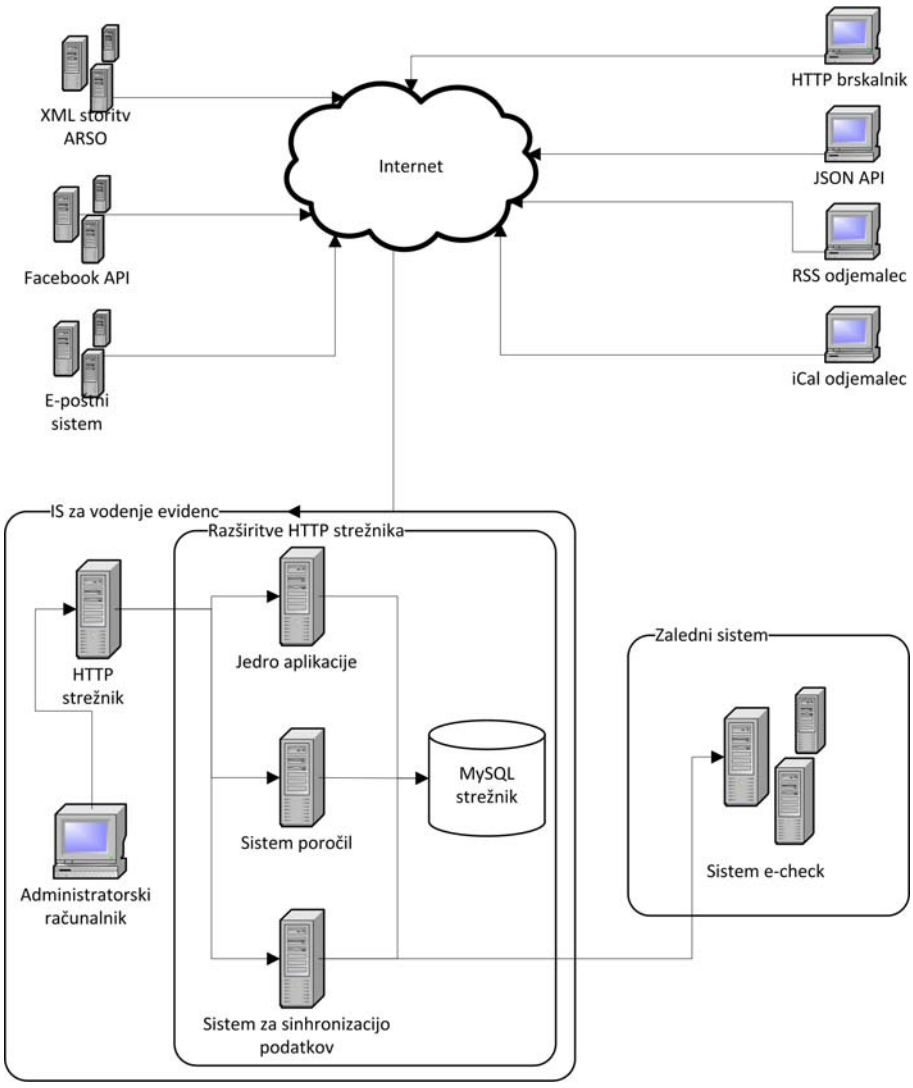
Mapa	*Opis mape*
/apl	korenska mapa
/apl/config	konfiguracijske datoteke ter predloge
/apl/config/html_templates	HTML predloge, ki določajo osnovni videz spletne strani
/apl/config/queries	parametrizirane predloge SQL poizvedb
/apl/controllers	kontrolniki, skozi katere pogledi dobijo podatke
/apl/models	vsebuje modele, ki pokrivajo vse operacije nad podatki
/apl/viewHelpers	razredi, ki so v pomoč izdelavi pogledov
/apl/viewHelpers/views	osnovni pogledi
. . . /views/adminUrnikView	pogledi administrativnega vmesnika za urnik
. . . /views/adminView	pogledi administrativnega vmesnika
. . . /views/inboxView	pogledi za operacije nad sporočili
. . . /views/indexView	pogledi za manipulacijo nad ostalimi podatki
. . . /views/statsView	pogledi za statistike
/libs	knjižnice
/web	datoteke objavljene na javnem delu strežnika (slikovne, css, js, . . . datoteke)

Tabela 3.1: Struktura map aplikacije.

3.5.2 Arhitektura sistema

Spletna aplikacija fitnes centra za podporo vodenju evidenc je večuporabniška spletna aplikacija. Za razvoj smo uporabili odprtokodne programske rešitve opisane v poglavju 2.5.

Informacijski sistem je sestavljen iz več sklopov, ki skupaj pokrivajo zastavljene funkcionalnosti. Slika 3.2 prikazuje komponente sistema in njihovo medsebojno povezavo.



Slika 3.2: Arhitektura informacijskega sistema.

Notranje komponente sistema so:

1. *HTTP strežnik* - spletni strežnik, ki je vstopna točka v naš portal. Aplikacija je nadaljnje sestavljena iz:
   (a) *jedro aplikacije*
   (b) *sistem poročil*
   (c) *sistem za sinhronizacijo podatkov*
2. *baza podatkov*

Zunanja komponenta, ki je del notranjega omrežja podjetja:

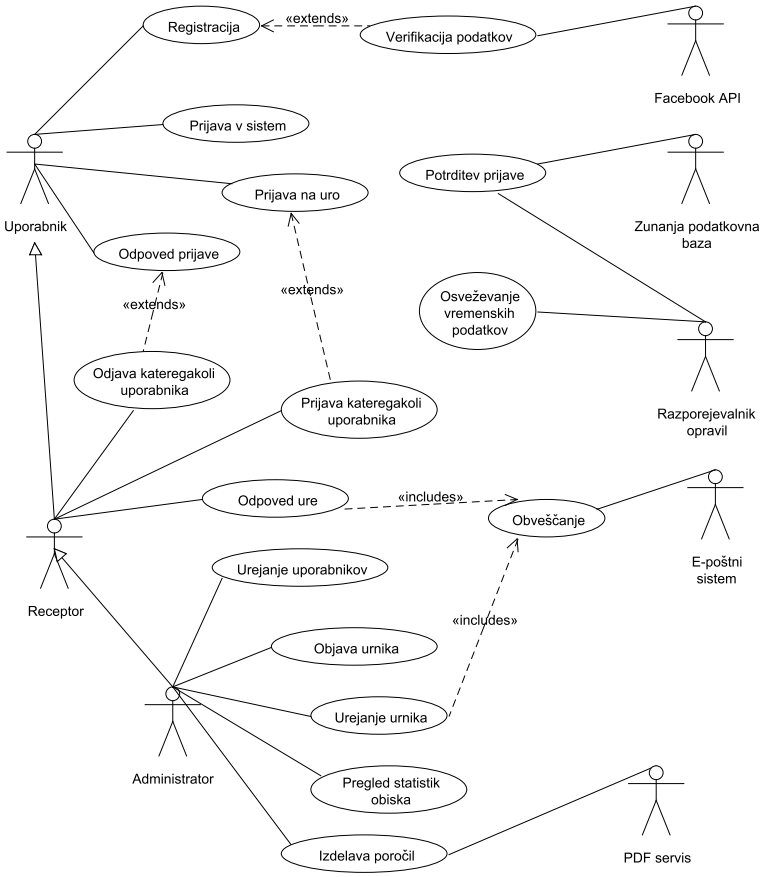
1. *sistem E-Check* - programski sistem za vodenje fitnes centra.

Ostale zunanje komponente (niso del notranjega omrežja podjetja):

1. *Facebook API* - opisan v poglavju 4.2.1,
2. *XML storitev ARSO* - storitev Agencije RS za okolje, ki ponuja tekoče vremenske podatke in vremenske napovedi v obliki XML,
3. *E-poštni sistem* - SMTP strežnik ponudnika internetnih storitev.

3.5.3 Primeri uporabe sistema

Delovanje programskega sistema je najlažje modelirati z diagramom primerov uporabe (angl. use case) saj omogoča grafični zapis funkcionalnosti sistema in nas vodi k reševanju težav že v sami zasnovi projekta [3]. Diagram primerov uporabe je tudi odlično orodje za preverjanje realizacije zahtevanih funkcionalnosti, kar je podlaga za končno oceno uspešnosti projekta.



Slika 3.3: Diagram primerov uporabe uporabljen za načrtovanje sistema.

Diagram na sliki 3.3 razkrije vpletenost dodatnih akterjev, ki jih pri samem zajemu zahtev nismo upoštevali. Poleg osnovnih akterjev (uporabnik, receptor in administrator) smo vpeljali še zunanje sisteme (zunanja podatkovna baza, sistemski razporejevalnik opravil, e-poštni sistem in PDF servis).

Uporabnik

Navadni uporabnik je glavni akter našega sistema. Na voljo mora imeti naslednje funkcionalnosti:

- registracija v sistem - uporabnik mora imeti možnost pričetka uporabe sistema, kot uporabniške podatke lahko uporabi podatke iz socialnega omrežja Facebook (če je to njegova želja),
- prijava v sistem - registriran uporabnik mora za dostop do svojih podatkov poznati uporabniško ime (številka članske kartice ali e-naslov) ter geslo, ki ga je podal ob registraciji; če uporabnik preferira prijavo z uporabo računa Facebook, lahko to stori preko iste prijavne maske,
- prijava na uro - prijavljen uporabnik mora imeti na voljo pregled urejen seznam vodenih vadb, na katere se lahko prijavi (prijava poteka preko istega vmesnika),
- odpoved prijave - uporabnik mora imeti na voljo seznam vodenih vadb, na katere je prijavljen; isti vmesnik mora omogočati odpoved prijave.

Receptor

Receptor je primerek navadnega uporabnika, ki ima poleg osnovnih funkcionalnosti na voljo še naslednje:

- odjava kateregakoli uporabnika - receptor sprejema odjave od ur vodenih vadb tudi preko telefona oziroma pri pultu - sistem mu mora omogočati, da uredi odjavo prijavljenih uporabnikov,
- prijava kateregakoli uporabnika - razširitev osnove funkcionalnosti prijave, ki omogoča prijavo poljubnega uporabnika. Receptorju je tudi dovoljeno da prijavi uporabnika, kljub temu da so kapacitete zasedene,
- odpoved ure - sistem mora omogočati uporabniku odpoved ure, pri tem mu mora omogočiti vnos razloga ter obveščanje uporabnikov, ki so bili prijavljeni na uro.

Administrator

Privilegiran uporabnik, ki mora imeti pooblastila za vse operacije sistema. Poleg primerov uporabe navadnega uporabnika in receptorja ima na voljo še možnost:

- urejanja uporabnikov - funkcionalnost omogoča urejanje uporabniških podatkov vseh uporabnikov,
- objave urnika - primer uporabe vključuje kreiranje urnika,
- urejanja urnika - urejanje parametrov urnika ter urejanje terminov ur vodenih vadb; vsaka sprememba že obstoječih urnikov sproži obveščanje vpletenih uporabnikov,
- pregleda statistik obiska,
- izdelave poročil - statistike morajo imeti možnost izvoza v formata PDF ter XLS.

Razporejevalnik opravil

Razporejevalnik opravil (angl. *task scheduler*) je sistemski proces, ki periodično proži naslednje primere uporabe:

- osveževanje vremenskih podatkov - evidenca obiska mora vsebovati tudi podatke o vremenu; vremenske podatke osvežujemo iz spletne strani Agencije RS za okolje,
- potrditev prijave - iz zalednega strežnika dobi podatke o obiskovalcih fitnes centra ter samodejno potrdi udeležbo uporabnikom, prijavljenim na ure vodenih vadb v časovnem okviru izvedbe operacije.

Facebook API

Zunanji sitem, ki omogoča povezavo s socialnim omrežjem Facebook. Sistem uporabljamo za pridobivanje in verifikacijo podatkov o uporabnikih.

Zunanja podatkovna baza

Podatkovna baza sistema E-check. Ta vsebuje podatke o strankah fitnes centra in podatke o trenutnih obiskovalcih. Akter sodeluje pri potrjevanju udeležbe obiskovalcev.

E-poštni sistem

Zunanji sistem za pošiljanje elektronskih sporočil, namenjen obveščanju o spremembah ter pošiljanju sporočil med uporabniki.

PDF servis

Del sistema, odgovoren za izdelavo PDF ter XLS poročil o obisku skupinskih vadb.

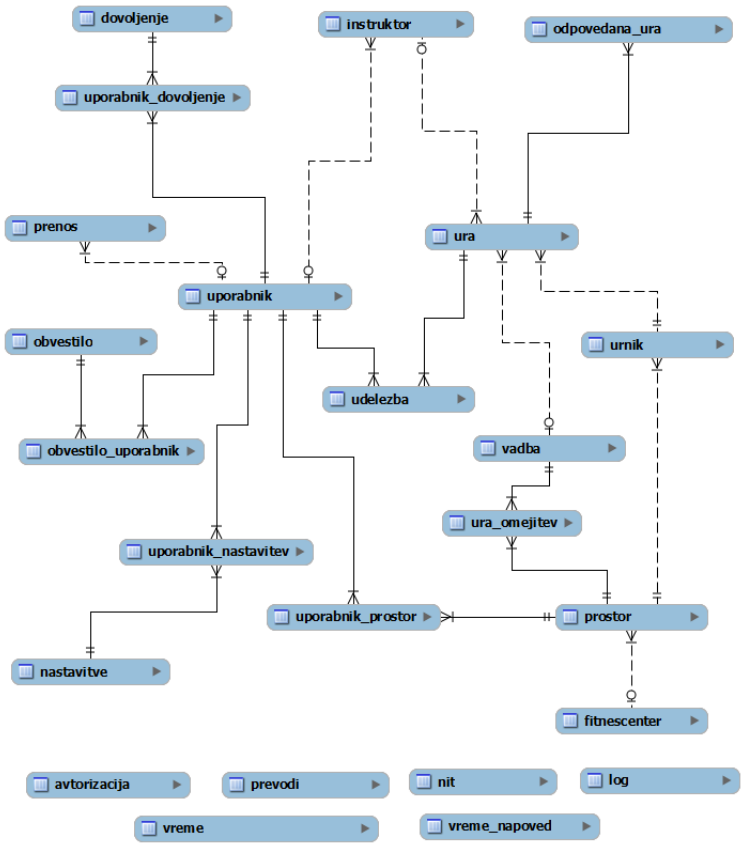
3.5.4 Podatkovni model

Osrednji del vsakega informacijskega sistema so prav podatki. Za potrebe elektronske evidence je smiselno ločiti entitetne tipe kot so: uporabnik, inštruktor, vadba, fitnes center in urnik. Seveda pa aplikacija potrebuje še druge enititetne tipe, ki zadevo smiselno povežejo med seboj. Poleg osnovnih entitetnih tipov je potrebno definirati še podporne tabele, kjer se shranjujejo podatki o obvestilih, dovoljenjih, prenosih, avtorizacijah, sistemskih dnevnikih in podobno. Naslednji podatkovni model (slika 3.4) prikazuje entitete v aplikaciji ter relacije med njimi.

Osnovne entitete podatkovnega modela

Slika 3.4 prikazuje Entitetno relacijski diagram podatkovne (ERD) baze. ERD vsebuje naslednje entitete:

- *dovoljenje* - šifrant hrani podatke o razpoložljivih dovoljenjih sistema (npr.: 'lahko ureja uporabnike'),
- *fitnescenter* - podatki o fitnes centrih,
- *instruktor* - podatki o inštruktorjih,
- *log* - dnevniški zapisi akcij uporabnikov,
- *obvestilo* - seznam vseh obvestil (vsebuje sporočila uporabnikov in obvestila sistema),
- *obvestilo_uporabnik* - seznam prejemnikov obvestil,
- *odpovedana_ura* - seznam ur, ki so bile odpovedane,
- *prenos* - seznam prenesenih datotek,
- *prostor* - seznam prostorov znotraj fitnes centra,
- *nastavitve* - možne nastavitve spletne strani,



Slika 3.4: Podatkovni model sistema.

- *udelezba* - udeležba na ure vodenih vadb - sem se zapisujejo prijave in odjave,
- *uporabnik* - podatki o uporabnikih sistema,
- *uporabnik_dovoljenje* - šifrant razpoložljivih dovoljenj (potrebno napolniti ob namestitvi),
- *uporabnik_nastavitev* - seznam nastavitev za uporabnika,
- *uporabnik_prostor* - seznam prostorov, ki so prikazani za uporabnika,
- *ura* - podatki o urniku - šifrant posamičnih vadb,
- *ura_omejitev* - posebne omejitve za posamično uro vodene vadbe,

- *urnik* - osnovni podatki o urnikih,
- *vadba* - šifrant vodenih vadb.

Podporne entitete

Poleg osnovnih objektov so za pravilno delovanje aplikacije potrebne še naslednje podporne entitete:

- *avtorizacija* - seznam avtorizacij iz zunanjih sistemov (izdelava poročil, pošiljanje e-pošte),
- *nit* - seznam trenutno tekočih niti v kolikor se gre za operacije, ki trajajo dalj časa in se lahko izvajajo samo v eni instanci,
- *prevodi* - seznam vseh *statičnih*[9] besedil informacijskega sistema,
- *vreme* - podatki iz avtomatskih meteoroloških postaj - tabela se polni iz spletnega servisa Agencije RS za okolje,
- *vreme_napoved* - meteorološke napovedi iz spletnega servisa Agencije RS za okolje.

Za natančen opis entitet glej dodatek A.

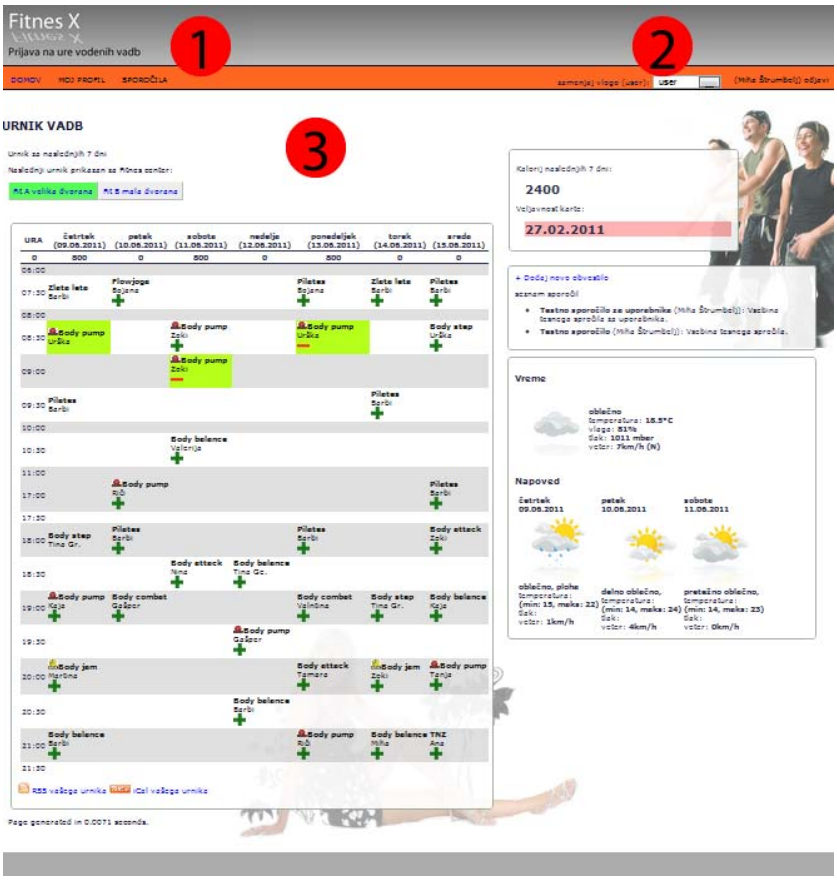
[9] omogoča večjezičnost

3.5.5 Uporabniški vmesnik

Grafični uporabniški vmesnik je za uporabnike najbolj pomemben del aplikacije. Pri spletnih aplikacijah je to tudi vse, kar vidijo. Uporabniška izkušnja je odvisna od videza, preglednosti vmesnika in odzivnosti aplikacije. Pri spletnih aplikacijah moramo biti pazljivi tudi na prenos podatkov med strežnikom ter odjemalcem, saj je lahko odjemalec v splet povezan z zelo počasno internetno povezavo.

Kombinacija tehnologij HTML, Javascript in CSS predlog[10] zadoščajo za izgradnjo dinamičnega interaktivnega uporabniškega vmesnika.

Videz uporabniškega vmesnika je na sliki 3.5.



Slika 3.5: Posnetek zaslona uporabniškega vmesnika.

Glavna področja portala so:

[10]CSS predloge določajo videz spletne strani

1. *zgodnji meni* - izpis opcij, ki so na voljo za prijavljenega uporabnika,
2. *hitra menjava nivoja uporabnika ter odjava* - uporabnik, ki ima dovoljenje menjave nivoja uporabnika lahko to vedno stori tukaj,
3. *osrednji vmesnik* - vsebnik trenutnega pogleda.

Poglavje 4

Opis rešitve

4.1 Portal

Končna aplikacija podpira vse funkcionalnosti, ki so potrebne za delo z urnikom, uporabniki ter podpira procese življenjskega cikla urnika.

Aplikacija je logično razdeljena na osem modulov:

- *jedro sistema*,
- *prva stran*,
- *administracija*,
- *administracija urnikov*,
- *uporabniški profil*,
- *urnik*,
- *sporočila*,
- *statistike*.

4.1.1 Jedro sistema

Modul aplikacije *Jedro sistema* pokriva temeljne storitve aplikacije. Te so dostop do podatkovne baze, pridobivanje podatkov iz Facebook Graph API, vremenskih servisov in pošiljanje elektronske pošte. Jedro sistema je tudi odgovorno za povezavo z zalednimi sistemi, kjer se vrši potrjevanje udeležbe

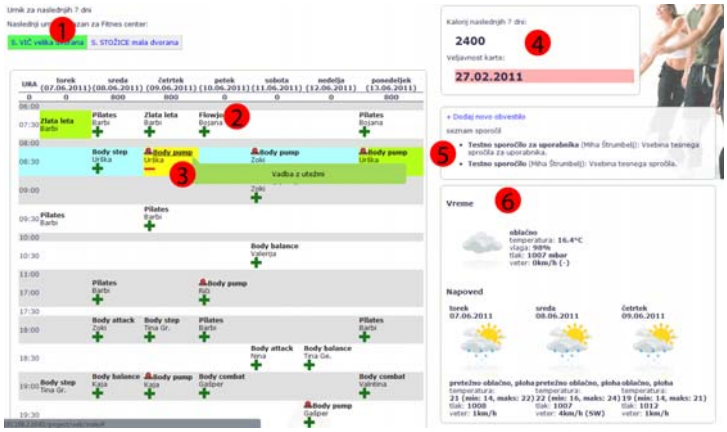
in sinhronizacija podatkov z zunanjim sistemom E-check. Do storitev jedra sistema lahko dostopamo preko API knjižnic JSON.

Jedro sistema vsebuje tudi modul za registracijo in modul za prijavo uporabnikov v sistem (posnetek zaslona na sliki 4.1).

Slika 4.1: Posnetek zaslona obrazca za prijavo v sistem.

4.1.2 Prva stran

Osrednji uporabniški modul portala je *prva stran* (slika 4.2). Informacije, ki jih ta pogled vsebuje, so: pregled urnika s termini vodenih vadb, izpis veljavnosti karte, predvideno število porabljenih kalorij za določen teden, sporočila in sistemska obvestila ter vreme in vremensko napoved.



Slika 4.2: Posnetek zaslona prve strani.

Uporabniku je omogočena izbira urnika po dvoranah (oznaka 1 na sliki: izbor prikazanih prostorov je na voljo v profilu uporabnika). Za izbrani prostor

se prikaže urnik za sedem dni naprej. Prijava ter odjava terminov se vršita z izbiro opcij *plus* (prijava - oznaka 2) in *minus* (odjava - oznaka 3) na želenem terminu. Zasedeni termini in termini, na katere prijava ni mogoča, so označeni z rdečo barvo. Termini, na katere je uporabnik prijavljen, pa so označeni s zeleno barvo. Prikazani so tudi termini današnjega dne, vendar prijava za nazaj ni mogoča, zato ti termini nimajo opcije prijave in odjave.

Za vsak dan v sedemdnevnemu obdobju se računa predvidena poraba kalorij, podatek se nahaja pod oznako dneva. Predvidena skupna poraba kalorij za naslednjih sedem dni in podatek o veljavnosti karte sta vidna na sliki 4.2 pod oznako 4.

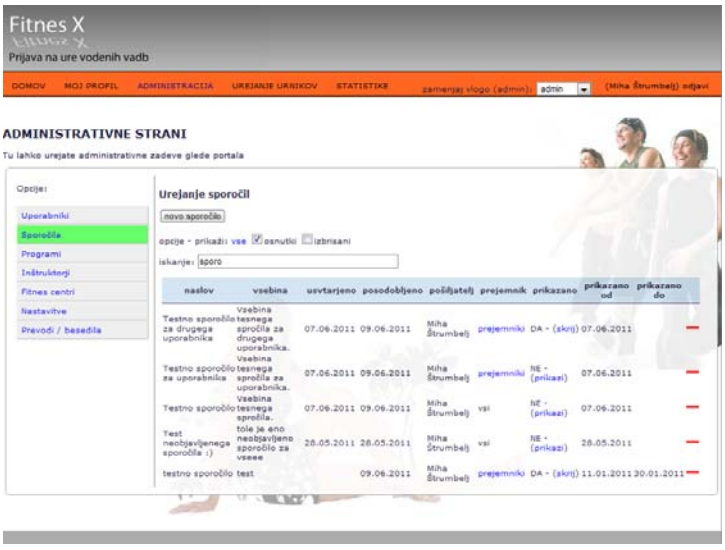
Na prvi strani se nahaja seznam sporočil ter sistemskih obvestil (oznaka 5) ter podatki o trenutnem vremenu in vremenski napovedi (oznaka 6).

4.1.3 Administracija

Vse potrebe po upravljanju z osnovnimi entitetami (razen urnika) se nahajajo v modulu *Administracija*.

Omogočeno je upravljanje z uporabniki in pregledovanje dnevnikov uporabe, pregled in urejanje fitnes centrov, inštruktorjev, šifranta vodenih vadb in sporočil.

Slika 4.3 prikazuje primer uporabe aplikacije za urejanje sporočil. Vmesnik omogoča pregled in filtriranje sporočil, dodajanje in brisanje ter spreminjanje vidnosti na prvi strani.



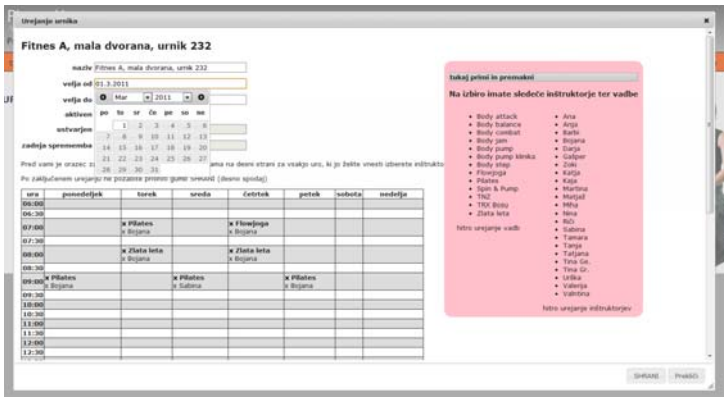
Slika 4.3: Posnetek zaslona administrativnega vmesnika - modul sporočila.

4.1.4 Administracija urnikov

Modul za urejanje urnikov omogoča grafično urejanje urnikov, spremembe se asinhrono shranijo na strežnik, ko uporabnik sproži akcijo *shrani*. Administrativni modul urejanja urnikov podpira vse akcije življenjskega cikla urnika.

Urnik je mogoče urejati tudi po tem, ko je že objavljen (v primeru, da pride do sprememb). V tem primeru se v podatkovni bazi obstoječ zapis časovno zaključi in odpre se nov zapis. Udeležbe za termine, ki se še niso izvedli, se preusmerijo na nove termine. Uporabniki so o spremembi obveščeni preko elektronske pošte.

Del administracije urnika je implementiran tudi na prvi strani. Receptorju in administratorju je omogočen vpogled v število prijavljenih za določen termin ter prijava in odjava udeležbe za poljubnega uporabnika. Preko tega vmesnika se izvaja tudi akcija odpovedi temina.



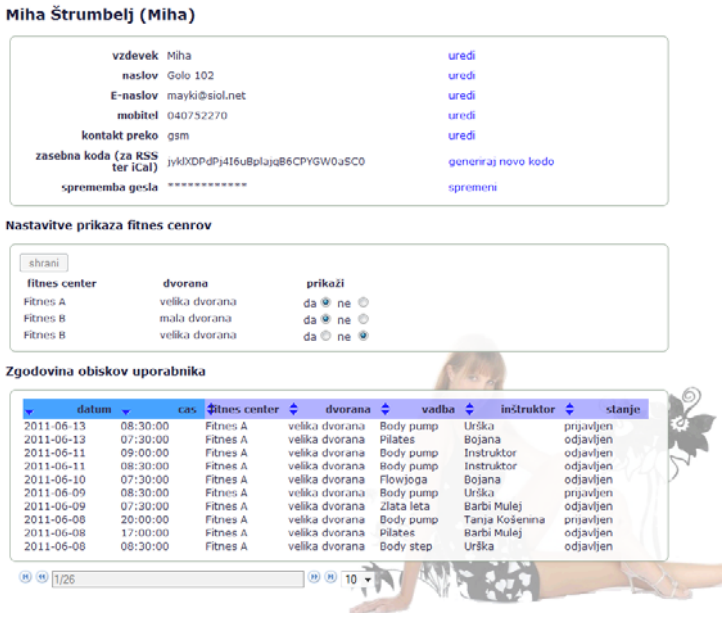
Slika 4.4: Posnetek zaslona za sestavljanje urnika.

Slika 4.4 prikazuje grafično urejanje urnika. Administrator ima na levi strani tabelo s tedenskim urnikom, ki ga lahko popravlja s pomočjo akcije *primi in spusti* (angl. *drag and drop*). Vmesnik omogoča tudi poenostavljeno urejanje šifranta inštruktorjev ter skupinskih vadb.

4.1.5 Uporabniški profil

Uporabniški profil (posnetek zaslona na sliki 4.5) je modul, ki uporabniku omogoča vpogled ter urejanje svojih zasebnih podatkov. Modul omogoča tudi pregled zgodovine prijavljanja ter odjavljanja terminov vodenih vadb. Uporabniku ima možnost izbire prikaza različnih urnikov na njegovi prvi strani.

Uporabnikova RSS koda je ključ do RSS in iCal vmesnika. Ta omogoča izvoz *živih zaznamkov* s termini vadb, na katere je uporabnik prijavljen za prihodnjih sedem dni v formatu RSS ali iCal (za uvoz v aplikacijo koledar). Vmesnik ne zahteva prijave v sistem, saj nekatere naprave za zajem *živih zaznamkov* ne omogočajo avtentikacije. V primeru, da uporabnik sumi zlorabo, lahko svoj RSS ključ kadarkoli zamenja preko tega vmesnika.



Slika 4.5: Posnetek zaslona obrazca za urejanja profila uporabnika.

4.1.6 Sporočila

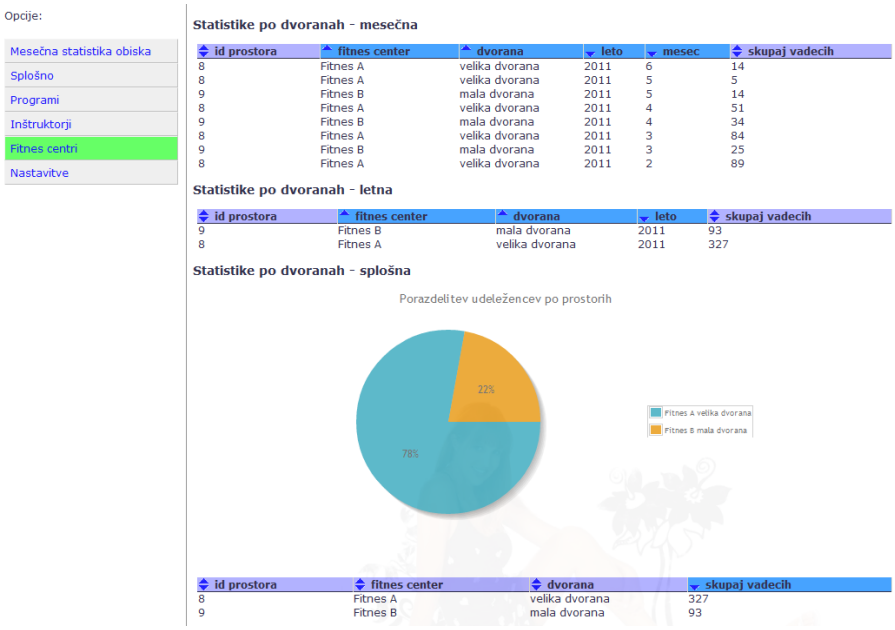
Modul sporočila (posnetek zaslona na sliki 4.6) omogoča vpogled v vsa sporočila na ločeni strani. Uporabniki, ki imajo dovoljenja za objavljanje sporočil, imajo tukaj na voljo tudi ta pogled. Pogled omogoča razvrščanje sporočil po vseh stolpcih.

4.1.7 Statistike

Modul statistik (slika 4.7) omogoča preglede statističnih podatkov o obiskanosti ur vodenih vadb. Sestavljen je iz petih podmodulov, ki omogočajo preglede po posamičnih prerezih. Statistike so grafično prikazane tam, kjer je to smiselno, in doda informacijo. Vse statistike je mogoče razvrščati po vseh stolpcih.

Slika 4.6: Posnetek zaslona modula sporočil.

Grafične prikaze je mogoče raziskovati po delih s pomočjo funkcionalnosti približevanja in oddaljevanja nivojev. Na grafičnih prikazih osnovnih statistik obiska ter mesečnih povzetkov je mogoče izbiranje vadb, ki so v prikaz zajete.



Slika 4.7: Posnetek izreza zaslona aplikacije za pregled statistik.

4.2 Integracija

4.2.1 Socialno omrežje Facebook

Proces prijavljanja v sistem smo poenostavili z omogočanjem prijave z uporabo računa Facebook.

Preko knjižnic Facebook API lahko dostopamo do naslednjih entitet socialnega omrežja: uporabniki, spletne strani, dogodki, skupine, aplikacije, sporočanje, fotografije in albumi, videi, zapiski, lokacije uporabnikov ter druge.

Vsi objekti v omrežju Facebook imajo svojo identiteto, na primer: do uporabnika 'Bret Taylor' pridemo tako da pošljemo HTTP GET zahtevek na https://graph.facebook.com/btaylor (kjer je https://graph.facebook.com/ spletni naslov storitve, 'btaylor' pa ID uporabnika).

Prijava v sistem poteka v dveh korakih:

1. **Javascript klient od ponudnika storitev zahteva dostopni žeton (ta je veljaven dokler je uporabnik prijavljen v isti seji)**

```
FB.init({
        appId: '177062205651059', cookie:true,
        status:true, xfbml:false
});
FB.login(function(response) {
  if (response.session) {
    doFBLogin();
  } else {
    // uporabnik je preklical prijavo
  }
}, {perms:'email,publish_stream'});
```

   Klic funkcije *FB.init()* je namenjen inicializaciji FB API knjižnice. Klic *FB.login()* povzroči odprtje dodatnega okna s strani ponudnika, kjer uporabnik izvrši prijavo v sistem. Ko uporabnik opravi s prijavo (uspešno ali neuspešno), se izvrši koda podane funkcije. Pri klicu funkcije zaprosimo za dovoljenja po objavljanju ter dostopanju do podatka o e-poštnem naslovu uporabnika. Uporabnik je obveščen o nivoju informacij, do katerega dovoli dostop.

2. **Preverjanje podatkov na strežniški aplikaciji**

```
$fbstr = get_url_https('https://graph.facebook.com/me?
    access_token=' . $_GET['access_token'];
$fb = json_decode($fbstr);
validate($fb);
```

Najprej pridobimo podatke o uporabniku preko knjižnice Facebook Graph API. Ker so ti v obliki JSON jih s PHP proceduro *json_decode()* pretvorimo v PHP objekt. Funkcija *validate()* nato opravi potrebno validacijo podatkov.

S preprostim postopkom (nekaj več logike je skrito v funkciji *validate()* ter *doFBLogin()*) smo postopek prijave poenostavili, hkrati pa pridobili dovoljenje za obveščanje uporabnika preko storitev Facebook.

4.2.2 ARSO - podatki o vremenu

Podatki o trenutnih vremenskih razmerah ter vremenskih napovedih so pridobljeni iz spletne storitve Agencije Republike Slovenije za okolje (ARSO). Storitev ponuja podatke rednih opazovanj, osvežene vsaj štirikrat dnevno in večdnevne napovedi, obnovljene vsaj dvakrat na dan.

Za namen naše spletne aplikacije pridobivamo podatke o trenutnih vremenskih razmerah iz avtomatske meteorološke postaje Ljubljana. V ta namen je spisana skripta, ki se vsako uro poveže na strežnike ARSO (http://meteo.arso.gov.si/), naloži najnovejši dokument XML, ga razčleni in zapiše v podatkovno bazo.

Podatki o vremenu so prikazani na *prvi strani* portala ter v *statistikah obiska*.

4.2.3 Informacijski sistem fitnes centra

Fitnes centri imajo svoj informacijski sistem, v katerem beležijo vse podatke o svojih strankah. Fitnes center, ki je sodeloval pri razvoju, ima v ta namen nameščen sistem E-Check. Razvijalci sistema so za namen integracije z našo aplikacijo dovolili dostop do podatkov preko pogledov SQL. To nam omogoča, da ob registraciji uporabnika preverimo obstoj številke člana in pravilnost vnesenih podatkov. Poleg preverjanja podatkov ob registraciji se preko istega vmesnika vrši še potrjevanje udeležbe in pridobivanje podatkov o veljavnosti kart.

4.3 Testiranje delovanja

Pomembnosti testiranja programske opreme ne gre podcenjevati. Testiranje je bistvena komponenta pri zagotavljanju kvalitete in vključuje pregled specifikacij, načrtovanja ter kodiranja. Podjetja, ki se ukvarjajo z razvojem programskih rešitev porabijo za testiranje skoraj toliko časa kot pri načrtovanju in implementaciji.

4.3.1 Cilji testiranja

Testiranje ni namenjeno preverjanju delovanja aplikacije, marveč odkrivanju primerov nedelovanja. Dejstvo je, da s testiranji ne moremo pokriti vseh funkcionalnosti sistema, saj je množica kombinacij vhodnih parametrov aplikacije neskončno velika. Dobro testiranje odkrije slabosti sistema.

4.3.2 Zajem napak

V aplikacijo smo vgradili modul beleženja napak ter ga povezali z uporabniškim modulom, kar nam omogoča odkrivanje ter sledenje napakam od same akcije, sprožene s strani uporabnika. Administrator ima nato pregled nad akcijami (bodisi uspešno bodisi neuspešno zaključenimi) uporabnika v administrativnem uporabniškem vmesniku pod rubriko *uporabnik*.

Programski jezik PHP omogoča enostavno prestrezanje napak. Spodnji primer kode predstavlja mehanizem prestrezanja ter beleženja napak, uporabljen pri razvoju naše aplikacije.

```
1 function main_error_handler($errno, $errstr, $errfile, $errline) {
2   if ($errno!=8) // 8 - opozorila
3     pisi_log($errno, $errstr, $errfile, $errline);
4 }
5
6 set_error_handler("main_error_handler", E_ALL);
```

Glavni del skripta je klic procedure `set_error_handler(<funkcija>, <nivo_lovljenja_napak>);` s parametri:

1. `<funkcija>` - naziv funkcije, ki se kliče ob napaki
2. `<nivo_lovljenja_napak>` - nivo beleženja napak. Možnosti so E_ERROR (samo napake), E_WARNING (opozorila), E_PARSE (napake pri interpretiranju), E_NOTICE (lažja opozorila) ter E_ALL (kombinacija napak, opozoril ter lažjih opozoril). Poleg prestrezanja PHP omogoča PHP tudi veliko drugih načinov ravnanja z napakami (npr.: direkten zapis v dnevnik) - le te pa moramo nastaviti v konfiguracijski datoteki php.ini.

4.3.3 Funkcionalno testiranje

Skozi razvojni cikel informacijskega sistema smo bili v kontaktu s potencialnimi bodočimi uporabniki. Glavni fokus testiranja aplikacije je v pokrivanju funkcionalnosti ter pravilni izvedbi le-teh. Potrebe po varnosti zahtevajo preverjanje aplikacije s testnimi scenariji nepooblaščenega dostopa do informacij.

4.3.4 Nefunkcionalno testiranje

V fazi načrtovanja smo si zadali nalogo zadostiti naslednjim nefunkcionalnim zahtevam:

1. varnost,
2. sočasni dostop,
3. dosegljivost sistema,
4. odzivnost.

Testiranja obremenitve in vzdržljivosti sistema smo opravili z grafičnim odprtokodnim orodjem Apache JMeter. Orodje omogoča izvajanje tako funkcionalnih kot nefunkcionalnih testov. Orodje originalno razvito za testiranje spletnih aplikacij, sedaj podpira tudi testiranje ostalih tipov aplikacij.

Testiranje je bilo opravljeno na virtualnem strežniku Microsoft Windows Server 2003 s programsko opremo:

- spletni strežnik Apache 2.2 s PHP 5.3.6,
- podatkovno bazo MySQL 5.5.11,
- strežnik Apache Tomcat 7.0,
- strežnik Memcache 2.2.6.

Ure vodenih vadb pri nas še niso zelo popularne, zato smo kot testni scenarij predvideli množico 30 osebkov, ki so določeno nalogo ponovili desetkrat. Cilj je simulirati najbolj pogost scenarij prijave uporabnika na ure vodenih vadb.

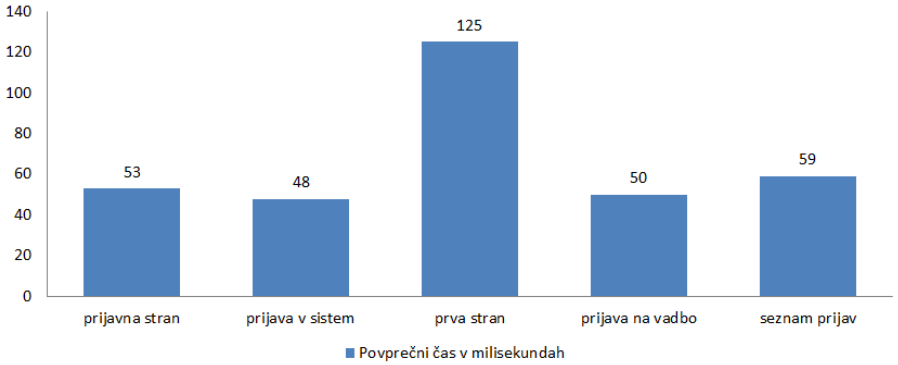
Zaporedje spletnih strani, ki jih od strežnika zahteva brskalnik z namenom uresničitve akcije, je:

1. prijava v sistem (login.php),
2. izpis domače strani (index.php),
3. asinhrono populiranje urnika (api.php),
4. prijava na uro vadbe (api.php).

Rezultati meritev testa obremenitve - eden uporabnik zahtevo ponovi desetkrat

Sistem s sočasno strežbo enemu uporabniku je trenuten sistem v vseh fitnes centrih, največje število ur, na katere so se uporabniki prijavljali, pa je 10 tedensko. Test obremenitve z enim sočasnim uporabnikom tako zadosti osnovnim potrebam po odzivnosti, vendar pa sistem z enim mestom v čakalni vrsti povzroča (v konicah) čakanje zahtevkov.

Tabela 4.1 prikazuje rezultate testiranja obremenitve z enim samim uporabnikom. Vsi časi so navedeni v milisekundah. Rezultati meritev povprečnih časov dostopa so grafično prikazani na sliki 4.8. Uporabnikov brskalnik je na odgovor povprečno čakal 67 milisekund, najdaljši čas je bil 591 milisekund, najkrajši pa 28 milisekund. Povprečen čas, potreben za izvedbo celotnega postopka od prijave v sistem do potrditve prijave na vadbo, je 335 milisekund. Kot vidimo na sliki 4.8 je v povprečju najdaljša zahteva bila zahteva po prvi strani.



Slika 4.8: Graf meritev izvajanja prijave pod visoko obremenitvijo (1 uporabnik ponovi postopek desetkrat).

	avg.	*mean.*	*90%*	*min.*	*maks.*	*propustnost*	*KB/sek*
prijavna stran	53	41	51	28	150	2,992	51,706
prijava v sistem	48	47	53	34	67	3,088	0,177
prva stran	125	76	83	54	591	3,053	377,626
prijava na vadbo	50	46	59	39	70	3,086	0,168
seznam prijav	59	58	66	53	69	3,084	34,664
	67	53	77	28	591	13,982	425,904

Tabela 4.1: Rezultati testa obremenitve z enim uporabnikom in 10 zahtev. Stoplci prikazujejo: avg - povprečno vrednost, mean. - mediano, 90% - devetdeseti percentil; 90% meritev časa pade pod ta čas, min - najkrajši čas odgovora, maks - najdaljši čas odgovora, propustnost - število obdelanih zahtev v sekundi, KB/s - količina prenesenih podatkov.

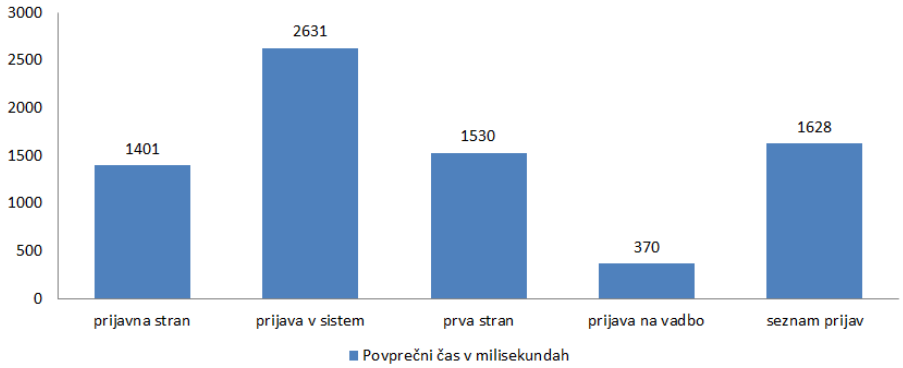
Rezultati meritev testa obremenitve trideset uporabnikov, vsak desetkrat ponovi proces

Testiranje obremenitve z enim uporabnikom ni realno, zato smo izvedli še testiranje s 30 ter 60 sočasnimi uporabniki. Iz meritev za enega uporabnika lahko razberemo, da naš sistem zmore strežbo približno s 14 zahtevami v sekundi, propustnost aplikacije pa je približno 426 KB/s.

Obremenitev sistema z enim uporabnikom je pokazala, da en cikel traja v povprečju 335 milisekund. Povprečen čas, potreben za cikel pri povečani obremenitvi bi moral biti 30 krat daljši, torej 10050 milisekund. Rezultati testiranja v tabeli 4.2 pokažejo, da je povprečni čas res večji. Povprečen čas za izvedbo celotnega cikla je pri drugi meritvi 7560 milisekund, kar je 2490 milisekund (25%) manj kot je bilo pričakovano.

Povprečen čas za nalaganje spletne strani je bil pri taki obremenitvi 1,5 sekunde, najkrajši čas je bil le milisekundo daljši od najmanjšega časa prejšnjega testiranja z enim uporabnikom. Najdaljši čas nalaganja pa je bil več kot 10 sekund, vendar je 90% zahtevkov bilo postreženih v manj kot 3,6 sekunde kar je sprejemljiv rezultat.

Slika 4.9 prikazuje povprečne nalaganja posamičnih strani v milisekundah. Vidimo, da je prijava na uro še vedno najhitrejša akcija. Pri tej obremenitvi se je izkazalo da je v povprečju najpočasnejša akcija prijava v sistem. Ta je imela tudi najdaljši čas nalaganja - 10353 milisekund.



Slika 4.9: Graf meritev izvajanja prijave pod visoko obremenitvijo (30 sočasnih uporabnikov ponovijo postopek desetkrat).

	avg.	*mean.*	*90%*	*min.*	*maks.*	*propustnost*	*KB/sek*
prijavna stran	1401	1277	2301	58	6539	3,620	62,562
prijava v sistem	2631	2044	6351	33	10353	3,626	0,209
prva stran	1530	1172	2804	66	7279	3,628	448,783
prijava na vadbo	370	89	1019	29	3887	3,668	0,201
seznam prijav	1628	1188	3690	43	7174	3,669	41,238
	1512	1139	3579	29	10353	18,040	549,521

Tabela 4.2: Rezultati testa obremenitve s 30 sočasnih uporabnikov (vsak 10 zahtev). Stoplci prikazujejo: avg - povprečno vrednost, mean. - mediano, 90% - devetdeseti percentil; 90% meritev časa pade pod ta čas, min - najkrajši čas odgovora, maks - najdaljši čas odgovora, propustnost - število obdelanih zahtev v sekundi, KB/s - količina prenesenih podatkov.

Rezultati meritev testa obremenitve ko šestdeset uporabnikov vsak postopek ponovi desetkrat

Literatura [6] omenja, da so uporabniki pripravljeni čakati na povratno informacijo največ 8 do 12 sekund brez indikatorja napredka. Najdaljši čas nalaganja spletne strani pri testiranju z 30 uporabniki je bil 10 sekund, kar še vedno sodi v to kategorijo.

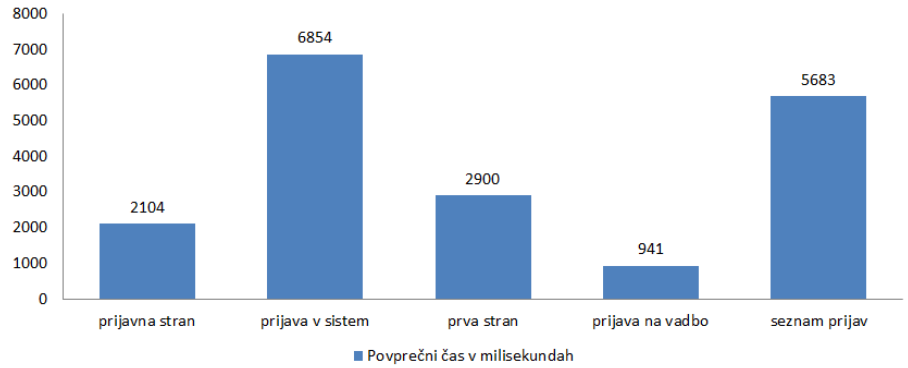
Poizkus smo izvedli še z dvakrat večjo obremenitvijo. Testni scenarij je predvidel enak postopek, ponovljen desetkrat, kot v prejšnjih primerih le da sedaj do aplikacije hkrati dostopa 60 uporabnikov.

Najdaljši čas nalaganja strani, kot vidimo v tabeli 4.3, se je sedaj povečal

na dobrih 20 sekund, kar pade iz meja sprejemljivih časov. Kljub zelo visoki obremenitvi, so vse zahteve bile postrežene, 90% zahtev je bilo postreženih v manj kot 9 sekundah. Povprečni skupni čas za izvedbo vseh akcij je bil okoli 3,7 sekunde.

Kot vidimo na sliki 4.10 je prijava v sistem še vedno najpočasnejša akcija. Čas nalaganja te strani je skoraj dvakrat večji od povprečnega. Druga najpočasnejša akcija je seznam prijav.

Kljub zelo visokim obremenitvam je sistem stabilen, najdaljši povprečni čas (kot vidimo na sliki 4.10) je 6854 milisekund, kar je sprejemljiva številka. Najhitrejša akcija se, tudi pri visoki obremenitvi, izvrši v manj kot sekundi.



Slika 4.10: Graf meritev izvajanja prijave pod visoko obremenitvijo (60 sočasnih uporabnikov ponovijo postopek desetkrat).

	avg.	*mean.*	*90%*	*min.*	*maks.*	*propustnost*	*KB/sek*
prijavna stran	2104	1530	4570	46	11793	3,059	52,866
prijava v sistem	6854	6964	11511	50	20118	3,074	0,177
prva stran	2900	2112	6580	63	10388	3,061	378,559
prijava na vadbo	941	443	2356	29	7179	3,097	0,169
seznam prijav	5683	5576	10694	47	15951	3,096	34,793
	3696	2376	8943	29	20118	15,179	462,404

Tabela 4.3: Rezultati testa obremenitve s 60 sočasnih uporabnikov (vsak 10 zahtev). Stoplci prikazujejo: avg - povprečno vrednost, mean. - mediano, 90% - devetdeseti percentil; 90% meritev časa pade pod ta čas, min - najkrajši čas odgovora, maks - najdaljši čas odgovora, propustnost - število obdelanih zahtev v sekundi, KB/s - količina prenesenih podatkov.

Testiranje take obremenitve nakazuje, da je sistem primeren tudi za večje namestitve ob predpostavki, da so procesorske strežniške zmogljivosti večje. Slika 4.11 prikazuje porabo procesorske moči po vpletenih procesih pri zadnjemu testu obremenitve z 60 sočasnimi uporabniki.

httpd.exe - proces spletnega strežnika Apache

memcached.exe - proces začasnega shranjevanja rezultatov v pomnilniku

mysqld.exe - proces podatkovne baze MySQL

Kot začasno shrambo podatkovnih struktur smo uporabili odprtokodno rešitev Memcached. Iz pregleda porabe procesorskega čas ter zasedenega pomnilnika, ki jo Memcache zahteva, se poraja vprašanje, ali je uporaba dodatnega nivoja začasne hrambe učinkovita. Da bi odgovorili na to vprašanje smo ponovili test brez uporabe rešitve Memcache. Test je pokazal, da je povprečni čas dostopa skoraj enak, največji čas pa se poveča na več kot 26 sekund. Zaradi večje potrebe po sočasnih povezavah na podatkovno bazo prihaja do primerov zavračanja povezave (sicer pri tej obremenitvi samo 0,6%), do katerega prej ni prišlo, saj je bila potreba po poizvedovanju po podatkih direktno v podatkovno bazo manjša.

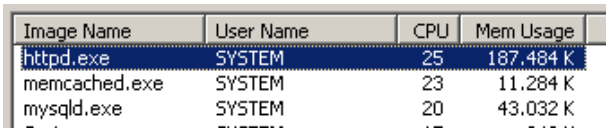


Image Name	User Name	CPU	Mem Usage
httpd.exe	SYSTEM	25	187.484 K
memcached.exe	SYSTEM	23	11.284 K
mysqld.exe	SYSTEM	20	43.032 K

Slika 4.11: Izrez posnetka zaslona Upravitelja opravil prikazuje porabo procesorskega časa.

Poglavje 5

Zaključek

5.1 Sklepne ugotovitve

Pri izdelavi diplomske naloge smo načrtovali in implementirali programsko rešitev za vodenje evidenc obiska vodenih vadb. Razvita programska oprema pokriva celoten proces vodenja evidenc od snovanja urnika do analize obiskanosti ter izdelave poročil. Naredili smo pregled teorije informacijskih sistemov ter ga poskusili uporabiti v praksi.

V diplomskem delu smo predstavili orodja in tehnologije, uporabljene pri razvoju, ter na koncu izvedli testiranje aplikacije na končnem okolju.

Glavni prispevki so bili v vzpostavitvi celotnega informacijskega sistema, vključno s sestavljanjem strojne opreme ter testiranjem na enem od slovenskih fitnes centrov. Tako vodstvo, kot uporabniki, ki so informacijski sistem testirali so bili z rešitvijo problema zadovoljni. Fitnes center, ki je programsko rešitev testiral, bo narejeno rešitev verjetno uporabil za optimizacijo svojega poslovanja.

Kljub uspehom je bilo pri razvoju videti več težav. Ena od funkcionalnosti informacijskega sistema je omogočanje prijave preko socialnega omrežja Facebook. Ta funkcionalnost je bila uspešno izvedena, a je vmes prenehala delovati zaradi spremenjenega vmesnika s strani ponudnika storitve. Kontrolnik, ki skrbi za povezovanje z omrežjem, smo popravili in funkcionalnost ponovno deluje.

Prvi rezultati testiranja kažejo, da bodo stranke fitnes centra dobro sprejele novost in jo tudi uporabljale. Fitnes center, ki rešitev uvaja je pripravljeno poskrbeti za promocijo uporabe storitve in se na ta korak že pripravlja.

Na željo podjetja smo dodali tudi sprotni izračun kalorij, ki jih bo uporabnik porabil, v primeru da se bo udeležil vseh vodenih vadb, na katere je pri-

javljen. Funkcionalnost na uporabnike deluje motivacijsko, saj dandanes na vsakem zavitku lahko preberemo, koliko kalorij bomo vnesli v sistem v primeru zaužitja določenega živila.

Nadaljnje možnosti razširitve vidimo predvsem v dodajanju inteligentnega modula za predlaganje sprememb urnika z namenom boljše obiskanosti. Fitnes center, ki rešitev testira, svojim uporabnikom v istem časovnem terminu ponuja izbiro več vodenih vadb na več lokacijah. To nam daje možnost razširitve z modulom za predlaganje izbire vadbe v stilu "morda bi vas zanimalo ...".

Dodatek A

Podatkovni model sistema

Seznam tabel z opisi polj:

AVTORIZACIJA

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
uporabnik_id	int(11)	D	FK
ustvarjen	timestamp	D	
uporabljena	timestamp	D	
velja_sekund	smallint(6)	D	
auth_string	varchar(20)	D	
ip	varchar(15)	D	
link	varchar(128)	D	

DOVOLJENJE

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naziv	varchar(128)	D	
modul	varchar(60)	D	
dovoljenje	varchar(256)	D	
rwu	varchar(1)	D	

FITNESCENTER

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naziv	varchar(128)	D	
kratkoIme	varchar(45)	D	
naslov	varchar(255)	D	
veljaOD	timestamp	D	
veljaDO	timestamp	D	
veljaven	int(1)	D	
ustvarjen	timestamp	D	
posodobljen	timestamp	D	

INSTRUKTOR

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
dolgoIme	varchar(128)	D	
kratkoIme	varchar(45)	D	
veljaOD	timestamp	D	
veljaDO	timestamp	D	
veljaven	int(1)	D	
ustvarjen	timestamp	D	
posodobljen	timestamp	D	
uporabnik_id	int(11)	D	FK
opis	varchar(256)	D	

LOG

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
stnapake	int(11)	D	
tekst	text	D	
cas	timestamp	D	
url	varchar(128)	D	
datoteka	varchar(128)	D	
errline	int(11)	D	
getparams	mediumtext	D	
postparams	mediumtext	D	
seja	mediumtext	D	
uporabnik	varchar(128)	D	
ip	varchar(256)	D	

NASTAVITVE

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	tinyint(4)	N	PK
nastavitev	varchar(256)	D	
vrednost	varchar(256)	D	
opcije	varchar(1024)	D	

NIT

ime stolpca	*tip podatka*	*prazno*	*ključ*
naziv_niti	varchar(30)	N	PK
zagnana	datetime	D	
ugasnjena	datetime	D	
zastavica	tinyint(4)	D	

OBVESTILO

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naslov	varchar(128)	D	
tekst	text	D	
ustvarjen	timestamp	D	
posodobljen	timestamp	D	
od	int(11)	D	FK
za	enum('uporabnik','vsi')	D	
prikazan	enum('ne','da')	D	
veljaOD	timestamp	D	
veljaDO	timestamp	D	
brisan	varchar(1)	D	

OBVESTILO_UPORABNIK

ime stolpca	*tip podatka*	*prazno*	*ključ*
obvestilo_id	int(11)	N	PK
uporabnik_id	int(11)	N	PK
ustvarjen	timestamp	D	

ODPOVEDANA_URA

ime stolpca	*tip podatka*	*prazno*	*ključ*
ura_id	int(11)	N	PK
datum	date	N	PK
razlog	varchar(128)	D	
ustvarjen	timestamp	D	
uporabnik_id	int(11)	D	FK
posiljanje_obvestila	enum('ne','da')	D	

PRENOS

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
original_fn	varchar(256)	D	
uporabnik_id	int(11)	D	FK
ustvarjen	timestamp	D	
ip	varchar(15)	D	

PREVODI

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
tag	varchar(128)	D	
lang	varchar(10)	D	
translation	varchar(1024)	D	

PROSTOR

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naziv	varchar(128)	D	
veljaOD	timestamp	D	
veljaDO	timestamp	D	
veljaven	int(1)	D	
ustvarjen	timestamp	D	
posodobljen	timestamp	D	
kapaciteta	int(1)	D	
fitnescenter_id	int(11)	D	FK

UDELEZBA

ime stolpca	*tip podatka*	*prazno*	*ključ*
ura_id	int(11)	N	PK
uporabnik_id	int(11)	N	PK
ustvarjen	timestamp	D	
posodobljen	timestamp	D	
status	enum('nev','pri','pot','zav','odj','pro')	N	
datum	date	N	PK

UPORABNIK

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
geslo	varchar(128)	N	
ime	varchar(30)	D	
priimek	varchar(60)	D	
spol	varchar(15)	D	
lokacija	varchar(80)	D	
naziv	varchar(256)	D	
naslov	varchar(256)	D	
sokol_id	varchar(45)	D	
gsm	varchar(45)	D	
email	varchar(128)	D	
privzeta_vloga	enum('user','manager','admin')	D	
kontakt_preko	varchar(45)	D	
ustvarjen	timestamp	N	
zadnja_prijava	datetime	D	
veljaOD	date	D	
veljaDO	timestamp	D	
posodobljen	datetime	D	
veljaven	varchar(1)	D	
rsskoda	varchar(32)	D	
fbid	int(11)	D	
veljavnost_karte	date	D	
auth_code	varchar(30)	D	
vloga	enum('admin','manager','user')	D	
instruktor_id	int(11)	D	FK

UPORABNIK_DOVOLJENJE

ime stolpca	*tip podatka*	*prazno*	*ključ*
uporabnik_id	int(11)	N	PK
dovoljenje_id	int(11)	N	PK
rwu	varchar(1)	D	
posodobljeno	timestamp	D	

UPORABNIK_NASTAVITEV

ime stolpca	*tip podatka*	*prazno*	*ključ*
uporabnik_id	int(11)	N	PK
nastavitev_id	tinyint(4)	N	PK
vrednost	varchar(256)	D	

UPORABNIK_PROSTOR

ime stolpca	*tip podatka*	*prazno*	*ključ*
uporabnik_id	int(11)	N	PK
prostor_id	int(11)	N	PK
prikazi	varchar(1)	D	
posodobljeno	timestamp	D	

URA

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naziv	varchar(45)	D	
ura	time	D	
dan	int(1)	D	
omejitev	tinyint(4)	D	
veljaven	tinyint(1)	D	
spremenjen	timestamp	N	
ustvarjen	timestamp	N	
VADBA_id	int(11)	D	FK
URNIK_id	int(11)	N	FK
INSTRUKTOR_id	int(11)	D	FK
parent_id	int(11)	D	FK
veljaven_do	timestamp	D	

URA_OMEJITEV

ime stolpca	*tip podatka*	*prazno*	*ključ*
omejitev	int(11)	N	
vadba_id	int(11)	N	PK
prostor_id	int(11)	N	PK

URNIK

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naziv	varchar(45)	D	
veljaOD	date	D	
veljaDO	date	D	
prioriteta	int(11)	D	
veljaven	tinyint(1)	N	
ustvarjen	timestamp	D	
posodobljen	timestamp	D	
prostor_id	int(11)	N	FK

VADBA

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
naziv	varchar(128)	D	
kratkoIme	varchar(45)	D	
opis	mediumtext	D	
link	varchar(256)	D	
kalorij	int(11)	D	
veljaDO	timestamp	D	
veljaven	int(1)	D	
ustvarjen	timestamp	D	
posodobljen	timestamp	D	
veljaOD	timestamp	D	
logo	varchar(128)	D	

VREME

ime stolpca	*tip podatka*	*prazno*	*ključ*
datum	date	N	PK
ura	time	N	PK
kraj	varchar(30)	D	
vreme	varchar(20)	D	
kolicina_oblacnosti	varchar(20)	D	
temperatura	tinyint(4)	D	
temp_ams	float	D	
vlaga	tinyint(4)	D	
temp_rosisca	tinyint(4)	D	
hitrost_vetra	tinyint(4)	D	
smer_vetra	varchar(3)	D	
sunki_vetra	tinyint(4)	D	
tlak	smallint(6)	D	
tlak_smer	varchar(20)	D	
padavine_mm	tinyint(4)	D	
sneg_cm	tinyint(4)	D	
sneg_cm_nov	tinyint(4)	D	
temp_ams_min	float	D	
temp_ams_avg	float	D	
temp_ams_max	float	D	
vlaga_ams	tinyint(4)	D	
hitrost_vetra_ams	tinyint(4)	D	
hitrost_vetra_avg_ams	tinyint(4)	D	
hitrost_vetra_max_ams	tinyint(4)	D	
tlak_avg_ams	smallint(6)	D	
padavine_acc_stanje	tinyint(4)	D	
padavine_interval_mm_ams	tinyint(4)	D	
soncno_obsevanje_ams	smallint(6)	D	
soncno_obsevanje_avg_ams	smallint(6)	D	
temp_tal_avg_30cm_ams	float	D	
temp_tal_avg_10cm_ams	float	D	

VREME_NAPOVED

ime stolpca	*tip podatka*	*prazno*	*ključ*
id	int(11)	N	PK
n_dni	tinyint(4)	D	
datum	date	D	
ura	time	D	
kraj	varchar(30)	D	
podrocje	varchar(30)	D	
za_dan	date	D	
oblacnost_icon	varchar(30)	D	
oblacnost_tekst	varchar(30)	D	
vreme_icon	varchar(30)	D	
vreme_tekst	varchar(30)	D	
min_temp	tinyint(4)	D	
max_temp	tinyint(4)	D	
temp	tinyint(4)	D	
vlaga	tinyint(4)	D	
temp_rosisca	tinyint(4)	D	
smer_vetra	varchar(3)	D	
hitrost_vetra	smallint(6)	D	
sunki_vetra	smallint(6)	D	
tlak	int(11)	D	
veljavna	tinyint(4)	D	

Slike

2.1 Evolucijski življenjski cikel 12
2.2 Klasična izgradnja spletne strani 17
2.3 Izgradnja dokumenta v formatu JSON 18
2.4 Razvojno okolje Eclipse 20
2.5 Razvojno okolje MySQL Workbench 21

3.1 Življenjski cikel urnika 24
3.2 Arhitektura IS 36
3.3 Diagram primerov uporabe 38
3.4 Podatkovni model 42
3.5 Posnetek zaslona uporabniškega vmesnika 44

4.1 Posnetek zaslona obrazca za prijavo v sistem 47
4.2 Posnetek zaslona prve strani 47
4.3 Posnetek zaslona administrativnega vmesnika - podmodul sporočila 48
4.4 Posnetek zaslona za sestavljanje urnika 49
4.5 Posnetek zaslona obrazca za urejanja profila 50
4.6 Posnetek zaslona modula sporočil 51
4.7 Posnetek zaslona aplikacije za pregled statistik 51
4.8 Graf meritev izvajanja prijave pod normalno obremenitvijo . . . 56
4.9 Graf meritev izvajanja prijave pod srednjo visoko obremenitvijo s 30 sočasnimi uporabniki 58
4.10 Graf meritev izvajanja prijave pod obremenitvijo 60 sočasnih uporabnikov 59
4.11 Izrez posnetka zaslona Upravitelja opravil 60

Tabele

3.1 Struktura map aplikacije. 35

4.1 Rezultati testa obremenitve z enim uporabnikom in 10 zahtev . 57
4.2 Rezultati testa obremenitve s 30 sočasnih uporabnikov (vsak 10 zahtev) 58
4.3 Rezultati testa obremenitve s 60 sočasnih uporabnikov (vsak 10 zahtev) 59

Literatura

[1] R. Vidgen, D. Avison, B. Wood, T. Wood-Harper, "Developing Web Information Systems, " Velika Britanija: Butterworth-Heinemann, 2002, pogl. 1

[2] J. Shore, S. Warden, "The Art of Agile Development, ", Združene države Amerike: O'Reilly Media, 2006 Velika Britanija: Butterworth-Heinemann

[3] R. Miles, K. Kamilon, "Learning UML 2.0," Sebastopol, Kalifornija, Združene države Amerike: O'Reilly Media, 2006

[4] D. E. Avison, G. Fitzgerald, "Information systems development : methodologies, techniques and tools, " London, Velika Britanija: McGraw-Hill, 1995

[5] B. Shneiderman, C. Plaisant, "Designing the User Interface: Strategies for Effective Human-Computer Interaction, " Boston: Addison-Wesley, 2010

[6] A. B. King, "Speed Up Your Site: Web Site Optimization", Boston, Združene dražve Amerike: New Riders, 2003

[7] (2010) PHP Documentation. Dostopno na: http://php.net/docs.php

[8] (2010) jQuery Documentation. Dostopno na: http://docs.jquery.com/Main_Page

[9] (2010) Introducing JSON. Dostopno na: http://www.json.org/

[10] (2010) Optimizing Page Load Time. Dostopno na: http://www.die.net/musings/page_load_time/