Kauno technologijos universitetas

Informatikos fakultetas

Sveikos gyvensenos planavimo internetinė programa

Baigiamasis bakalauro studijų projektas

Tautvydas Dikšas

Projekto autorius

prof. Eduardas Bareiša

Vadovas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Sveikos gyvensenos planavimo internetinė programa

Baigiamasis bakalauro studijų projektas

Programų sistemos (612I30002)

Tautvydas Dikšas

Projekto autorius

prof. Eduardas Bareiša

Vadovas

doc. Recenzentas Recenzaitis

Recenzentas

Kaunas, 2021

Kauno technologijos universitetas

Informatikos fakultetas

Tautvydas Dikšas

Sveikos gyvensenos planavimo internetinė programa

Akademinio sąžiningumo deklaracija

Patvirtinu, kad mano, Studentės Studentaitės, baigiamasis projektas tema „Sveikos gyvensenos planavimo internetinė programa“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) (parašas)

Tautvydas Dikšas. Sveikos gyvensenos planavimo internetinė programa. Bakalauro studijų baigiamasis projektas / vadovas prof. Eduardas Bareiša; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai, Programų sistemos.

Reikšminiai žodžiai: laiko planavimas, internetinė, mobili aplikacija.

Kaunas, 2021. 22 p.p.

Santrauka

Šiame darbe pristatoma sveikos gyvensenos planavimo sistema. Daug žmonių, kurių gyvenimo tempas yra nemažas ir sudėtinga spėti į kelias vietas vienu metu. Kuriama sistema yra aktuali dėl suteikiančių galimybių stebėti informaciją apie save veiklos metu taip leidžiant asmeniui kontroliuoti save. Taigi, pagrindinis darbo tikslas - palengvinti žmogaus būsenos stebėjimą veiklų metu.

Analizuojant konkurentų technologijas, buvo išskiriami privalumai ir trūkumai, lyginamas dabartinis funkcionalumas. Surinkta informacija iškėlė idėją, kad dabartinės populiariausios sistemos yra koncentruotos į minimalų funkcionalumą, taip išlaikant sistemą paprastą naudotojui. Darbo metu buvo parengtas sveikos gyvensenos planavimo sistemos projektas.

Sistema buvo realizuota naudojant Javascript ir Typescript programavimo kalbas, Vue.JS bei Express.JS, Nativescript karkasais. Sistemos programiniui kodui buvo parašyti unit testai, kurie leido užtikrinti kodo paleidimo sklandumą. Atliktas detalus vartotojo vadovas leidžiantis naujiems klientams lengviau naviguoti. Sukurta laiko planavimosi sistemėlė, kuri naudoja Fitbit technologijas.

Tautvydas Dikšas. Health planner web app. Bachelor's Final Degree Project / supervisor prof. Eduardas Bareiša; Informatics Faculty, Kaunas University of Technology.

Study field and area (study field group): Computer Sciences, Software Systems.

Keywords: time planning, web app, mobile app

Kaunas, 2021. 22 p.p.

Summary

Lorem ipsum dolor sit amet, eam ex decore persequeris, sit at illud lobortis atomorum. Sed dolorem quaerendum ne, prompta instructior ne pri. Et mel partiendo suscipiantur, docendi abhorreant ea sit. Recteque imperdiet eum te.

Eu eum decore inimicus consetetur, cu usu habeo corpora intellegam. Ut antiopam efficiendi deterruisset sit. Mel sint eirmod id, qui quot virtute id, dolor nemore forensibus usu id. Fugit dolore voluptatum cu vim. An vix veniam graecis insolens, sit posse iusto id. Ut vim ceteros percipit, id quo ubique recusabo, eum sint lucilius ea. In sumo inani numquam has.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų žodynas

Santrumpos:

Doc. – docentas;

Lekt. – lektorius;

Prof. – profesorius.

Terminai:

Saityno analitika – lorem ipsum dolor sit amet, eam ex decore persequeris, sit at illud lobortis atomorum. Sed dolorem quaerendum ne, prompta instructior ne pri. Et mel partiendo suscipiantur, docendi abhorreant ea sit. Recteque imperdiet eum te.

Tinklaraštis – lorem ipsum dolor sit amet, eam ex decore persequeris, sit at illud lobortis atomorum. Sed dolorem quaerendum ne, prompta instructior ne pri. Et mel partiendo suscipiantur, docendi abhorreant ea sit. Recteque imperdiet eum te.

Beje, darbe rekomenduojame pateikti tik svarbesnes ir mažiau žinomas santrumpas bei terminus (tarkime tokių santrumpų kaip HTML, PC, IT paaiškinti nereikia)

Įvadas

Supažindinama su darbo specifika, aktualumu, išdėstomi tikslai bei uždaviniai, aptariama dokumento struktūra. Šiame skyriuje apie darbą kalbama abstrakčiai, nederėtų pateikti nuorodų į kitus šaltinius (1 – 2 lapai).

Darbo problematika ir aktualumas

Apibrėžiama darbo problematika ir aptariamas aktualumas. Šiame poskyryje taip pat nurodoma su darbu susijusi sritis, praktinė darbo reikšmė.

Darbo tikslas ir uždaviniai

Suformuluojamas pagrindinis darbo tikslas, kuris išskaidomas į kelis uždavinius (3 – 6 uždaviniai), skirtus tikslui pasiekti. Išvados dokumento pabaigoje formuluojamos uždavinių pagrindu.

1. Išanalizuoti a
2. Pirmasis uždavinys
3. Antrasis uždavinys
4. Trečiasis uždavinys

Darbo struktūra

Aptariama dokumento struktūra. Nurodoma kiek ir kokių skyrių dokumente yra ir kokia informacija juose pateikiama.

Sistemos apimtis

Nors tai nėra tiesiogiai įvadinis sistemos aprašas, rekomenduojame būtent čia nurodyti jau realizuotos sistemos apimtį. Matai gali būti įvairūs: darbo valandos, kodo eilučių skaičius, komponentų, klasių, modulių kiekis, teikiamų paslaugų skaičius ir pan.

1. Analizė

1.1. Techninis pasiūlymas

1.1.1. Sistemos apibrėžimas

„PopoPlan” - tai sveikos gyvensenos planavimosi sistema, kuri leidžia naudotojui planuotis veiklas ir stebėti surinktą informaciją apie save. Kuriamas laiko planavimo sprendimas padėtų asmeniui analizuoti dabartinę sveikatos būseną ir optimizuoti darbo krūvius.

Sistemą sudaro šios dalys:

- Klientinė dalis - grafinė naudotojo sąsaja
- Serverio dalis - programų sąsaja (angl. API), atsakinga už duomenų valdymą
- Mobili aplikacija - grafinė naudotojo sąsaja, pritaikyta išnaudoti mobiliaus prietaiso funkcijas
- Laikrodžio aplikacija - programinė įranga, skirta sekti vartotojo būseną ir siųsti informaciją į serverį.

1.1.2. Bendras veiklos tikslas

Bendras veiklos tikslas - palengvinti duomenų prieinamumą vartotojo sveikatos analizei veiklos metu. Duomenys, tokie kaip pulso matavimo rezultatai, gali padėti aptikti problemas, pavyzdžiui, per didelį stresą darbo metu, pervargimą. Tokiu būdu vartotojas gali atlikti išvadas siekant geresnės fizinės savijautos.

Sėkmingai įdiegta sistema gali sudominti ir motyvuoti žmones atlikti savo veiklos grafiko optimizavimą.

1.1.3. Sistemos pagrįstumas

Rinkoje yra daug planuoklių, kurie leidžia vartotojams susikurti veiklas, taip sudarydami savo tvarkaraštį. Kadangi daugelis tokių aplikacijų susikoncentravusios į minimalistinį funkcionalumą, kurios fiksuoja tik veiklas, todėl vartotojui būtų naudinga turėti sistemą, kurioje gali laikyti visą informaciją bei gauti įdėjų veikloms kaip mityba ir sportas,

Taip pat vienas iš veiksnių sukurti „popoPlan” sveikatingumo planavimo sistemą buvo noras išnaudoti fitbit suteiktas technologijas. Dabartinia fitbiti prietaisai turi daug funkcijų, kurie gali būti laikomi medicinoje pagalbiniais prietaisais.

1.1.4. Konkurencija rinkoje

1 lentelė. Konkurentų apžvalga

Lyginimo kriterijai	**Sistema A**	**Sistema B**	**Sistema C**
Savybė 1	Realizuota	Nerealizuota	Realizuota iš dalies
Savybė 2	1000 naudotojų[1]	5000 naudotojų	20000 naudotojų
Savybė 3	Android	iOS	Android
Savybė 4	+	+	-
Savybė 5	3.99€	19.99€	Nemokama
…	…	…	…

Po lentele taip pat rekomenduojama aprašyti palyginimo kriterijus - ką jie reiškia, kodėl jie svarbūs, kodėl buvo pasirinkti.

[1] Pateikiant statistiką ar skaičius reikia nurodyti šaltinį, kurį rekomenduojama įtraukti į literatūros sąrašą

1.1.5. Prototipai ir pagalbinė informacija

Kuriant internetinę aplikaciją buvo naudojamos „wireframe" technologijos, kurios pritaikytos kurti preliminarų tinklapio įvaizdį su navigavimu.

Dirbant prie mobilios aplikacijos, buvo naudojami esami NativeScript komponentai. Kiekvienos sistemos kūrimo iteracijos metu, komponentų vietos buvo koreaguojamos.

1.1.6. Ištekliai, reikalingi sistemai sukurti

„popoPlan" sistema turi 4 dalis: kliento dalis, serveris, mobilus ir laikrodis. Norint saugoti kodą ir įkelti į debesį, reikia keturių atskirų kodo talpyklų įrankiuose kaip GitHub, kurių paskirtis leisti programuotojui patogiai talpinti programas, kurias vartotojai galėtų pasiekti. Pačios sistemos kūrimui užtenka vieno programuotojo. Sukurti sistemai reikėjo apie 300 darbo valandų.

1.2. Galimybių analizė

1.2.1. Techninės galimybės

Internetinės aplikacijos grafinė sąsaja buvo realizuota naudojant Vue.JS karkasą su JavaScript programavimo kalba. Šis karkasas buvo naudojamas dėl to, nes yra greitesnis nei alternatyvos kaip Angular ir React. Vue karkasas taip pat

Serverio aplikacija buvo aprašoma naudojant Express.JS karkasą su TypeScript programavimo kalba. Express.JS yra patogi technologija norint sukurti API. Juo lengvai galima aprašyti API puslapių kelius, HTTP metodus su pranešimais, aiškiai dokumentuotas ir lengva išmokti. TS programavimo kalba buvo naudojama, kadangi serverio aplikacijoje apdorojami duomenys, todėl statiniai tipai gali praversti rašant aiškesnį kodą. Taip pat naudojmas Mongoose ORM (angl. Object-relational mapper) karkasas, kuris leidžia lengvai komunikuoti su MongoDB duomenų baze vykdant užklausas per objektus. MongoDB duomenų bazė pasirinkta, nes yra greita ir efektyvi mažoms aplikajoms.

Mobiliai aplikacijai buvo naudojamas NativeScript su TypeScript programavimo kalba. NativeScript karkasas suteikia galimybę kurti aplikacijas Android ir iOS operacinėms sistemoms. NS leidžia sutaupyti daugiau laiko nekuriant atskirų aplikacijų Android OS su Java/Kotlin kalba, o ant iOS - Swift. Taip pat yra puikus pasirinkimas internetinių aplikacijų kūrėjams, nes karkasas veikia su Vue.js, React, Angular technologijomis. Kadangi NativeScript yra open-source community įrankis, buvo susidurta su dokumentacijos trūkumais, ne aiškiais pavyzdžiais ir lėtai užkraunamu projektu. TypeScript kalba su Vue karkasu buvo užkrauta sukūrus projektą.

Laikrodžiui naudojamos technologijos buvo paties Fitbit SDK (angl. Source development kit).

1.2.2. Vartotojų pasiruošimo analizė

Norint pasinaudoti sukurta „popoPlan" sistema

2. Projektas

Aprašoma detali sistemos specifikacija (20 – 50 lapų). Apibrėžiama kuriamo produkto vizija (koncepcija). Skyriaus struktūra ir pavadinimas priklauso nuo baigiamojo darbo specializacijos ir pačios temos specifikos, bet turi turėti funkcinių ir nefunkcinių reikalavimų skyrius.

2.1. Reikalavimų specifikacija

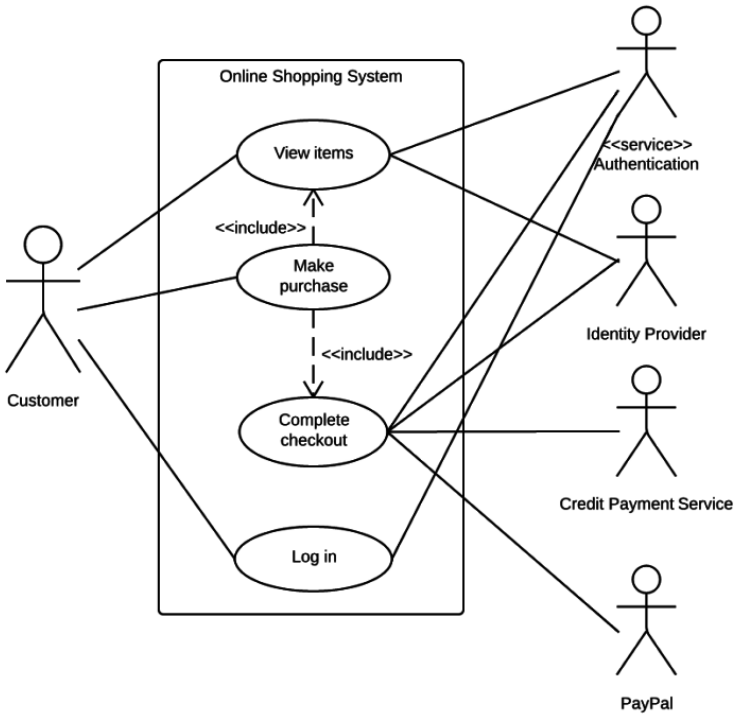
2.1.1. Komercinė specifikacija

Šiame skyrelyje nurodykite projekto užsakovą, projekto vykdytojus, aprašykite produkto vartotojus, detalizuokite projekto realizacijos laiko ir kainos apribojimus. Čia taip pat galima nurodyti verslo procesus (pvz., UML veiklos diagramomis), jei šie yra netipiniai ar reikalaujantys papildomo paaiškinimo.

Pvz.: „Maisto perdavimu protokolas yra užsakytas Jungtinių Tautų, jį kuria 10 IT specialistų. Pagrindiniai maisto perdavimo protokolo vartotojai – Afrikos vaikai ir jų tėvai, tačiau sistema aktyviai naudosis ir paprasti namų vartotojai visame pasaulyje“.

2.1.2. Sistemos funkcijos

Čia būtų išvardinamos visos sistemos funkcijos (aprašomi **funkciniai reikalavimai**), pavaizduotos UML panaudojimo atvejų diagrama ar diagramomis. Po kiekvienos iliustracijos turi būti papildomi kiekvienos funkcijos detalūs žodiniai aprašymai. Jei norite, galite aprašyti kiekvieną panaudos atvejį išsamiai (PA specifikacijos lentele): atvejo pavadinimas, tikslas, aprašymas, prieš sąlyga, po sąlyga, susiję panaudojimo atvejai (*include*, *extend*), aktorius kt.



1 pav. Sistemos panaudojimo atvejų diagrama

2.1.3. Vartotojo sąsajos specifikacija

Vartotojo sąsajos specifikacijoje turi būti nurodomi reikalavimai vartotojo sąsajos vaizdai. Čia nereikia ir negalima dėti jau egzistuojančios programos screenshot'ų! Šiame etape tik nusakoma, kokia turi būti vartotojo sąsaja (rekomenduojame **Balsamiq Mockups**, **Axure RP** ir panašius įrankius), tačiau galutinis sąsajos vaizdas nurodomas tik vėlesniuose skyriuose. Jei pradinėje kūrimo fazėje buvo naudojami vartotojo sąsajos eskizai ar prototipai, juos reikia dėti būtent į šį skyrelį.

2.1.4. Realizacijai keliami reikalavimai

Realizacijai gali būti keliami tokie **nefunkciniai reikalavimai**: reikalavimai sistemos išvaizdai, reikalavimai panaudojamumui, reikalavimai vykdymo charakteristikoms, reikalavimai veikimo sąlygoms, reikalavimai sistemos priežiūrai, reikalavimai saugumui, kultūriniai-politiniai reikalavimai, teisiniai reikalavimai. Jie išvardinami ir aprašomi šiame skyrelyje. Pavyzdžiui:

1. Maisto perdavimo protokolas privalo būti saugus (neautentifikuoti kaimynai negali sužinoti, kokį maistą siunčiasi vartotojas)
2. Maistas suskaidytas paketais, turi pasiekti vartotoją nepagedęs
3. Maisto perdavimo protokolas turėtų palaikyti lietuviškos virtuvės produktus

Šiame punkte gali būti išvardinti (jeigu nustatyti) tokio tipo apribojimai: apribojimai sprendimui, diegimo aplinka, bendradarbiaujančios sistemos, komerciniai specializuoti programų paketai, numatoma darbo vietos aplinka, sistemos kūrimo terminai, sistemos kūrimo biudžetas. Jei reikalinga specifinė duomenų kontrolė (kokia informacija turi būti tikrinama įvedimo ar sistemos veikimo metu), ji taip pat aprašoma šiame skyrelyje.

2.1.5. Techninė specifikacija

Skyrelyje aprašykite techninę ir papildomą programinę įrangą, reikalingą sistemai. Nurodykite minimalius įrangos parametrus. Šis skyrelis, priklausomai nuo situacijos, gali būti formuluojamas kaip sąrašas, ką užsakovams reikės turėti, jeigu norės naudotis sistema, arba kokios aplinkos reikalauja užsakovas.

Pvz.: „Maisto perdavimo protokolui realizuoti būtinas interneto ryšys, 1 kilolitro maisto ir gėrimų perdavimo linija, specializuotas šaldytuvas, namų kompiuteris".

2.2. Projektavimo metodai

2.2.1. Projektavimo valdymas ir eiga

Šiame punkte nurodykite, kokį programinės įrangos kūrimo modelį (ar modelius) naudojote kurdami savo sistemą. Tai gali būti krioklio, iteracinis ar kitas modelis. Galite nurodyti kaip suskirstėte darbus ir kokiu eiliškumu juos atlikote.

2.2.2. Projektavimo technologija

Šiame punkte nurodykite, kokią naudojote projektavimo technologiją, standartus ir programinius įrankius projekto kūrimui. Aprašykite kokia ar kokiomis notacijomis (formaliais tekstiniais ir grafiniais žymėjimo / aprašymo standartais) naudojotės kurdami sistemos projektą.

2.2.3. Programavimo kalbos, derinimo, automatizavimo priemonės, operacinės sistemos

Šiame punkte, tiesiog, aprašykite, kokią programinę įrangą naudojote kurdami savo baigiamąjį darbą. Tiesa, rašyti kokią programinę įrangą naudojote šiai ataskaitai sukurti – nebūtina.

2.3. Sistemos projektas

Sistemos projektas – tai jūsų sistemos veikimo aprašymas. Tai dažniausiai nagrinėjama dokumento vieta (be išvadų) darbo peržiūros ir gynimo metu.

2.3.1. Statinis sistemos vaizdas

Šiame punkte reikėtų detalizuoti sistemos struktūrą. Priklausomai nuo projekto tipo (rekomenduojame pasikonsultuoti su vadovu) turėtumėte aprašyti savo sistemą panaudodami UML diagramas:

- Išdėstymo (angl. *UML deployment diagram*) – nepakeičiama tuo atveju, jei sistema naudoja išorinius servisus ar yra paskirstyta per keletą įrenginių. Geriausia pradėti nuo šios diagramos, nes ji greičiausiai supažindina su bendra sistemos sudėtimi.
- Komponentų (angl. *UML component diagram*) – geriausiai tinka tuomet, kai naudojamas komponentinis sistemos kūrimo būdas ir sistema susideda iš komponentų teikiančių programavimo sąsają (API).
- Paketų (angl. *UML package diagram*) - labai naudinga tuomet, jei sistema sugrupuota paketais.
- Klasių (angl. *UML class diagram*) - geriausiai tinka atvaizduoti sistemos struktūros detales. Jei projekte klasių naudojama daug, rekomenduojama detalizuoti tik esmines klases, o likusią struktūrą pateikti paketų diagrama.
- Aprašant statinį sistemos vaizdą taip pat turėtų būti pateikta ir duomenų bazės schema. Šiam tikslui gali būti naudojama esybių ryšių diagrama arba (geriausia) UML klasių diagrama. Jei naudojama ne reliacinė duomenų bazė, tuomet naudoti tokį duomenų bazės specifikavimo būdą, kurį siūlo kūrėjai arba bendruomenė.

2.3.2. Dinaminis sistemos vaizdas

Dinaminiame sistemos vaizde parodoma kaip sistema veikia naudojama. Šiame punkte pagal poreikį galima pavaizduoti sistemos veiksmus UML veiklos, sekų ir/arba būsenų diagramomis. Galite pasirinkti vieną iš jų, galite naudoti ir kelias (priklausomai nuo sistemos specifikos).

3. Testavimas

Aprašoma su sukurtos įrangos testavimu susijusi informacija (8 – 12 lapai). Skyriaus struktūra ir pavadinimas priklauso nuo baigiamojo darbo specializacijos ir pačios temos specifikos.

Nurodomas įrangos testavimo planas, testavimo duomenų rinkiniai ir gauti rezultatai. Nurodoma sistemos specifikacija ir sąlygos, prie kurių buvo atliekamas testavimas.

3.1. Testavimo planas

Testavimo planas – tai jūsų pasirinkta testų atlikimo tvarka. Galimas testavimo planas: komponentų testavimas, po kurio seka integracinis testavimas, o vėliau būna sąsajos testavimas.

3.2. Testavimo kriterijai

Šiame punkte aprašykite kriterijus, kurie jums buvo svarbūs testavimo metu. Tai gali būti ne tik informacijos ar skaičiavimų korektiškumas, bet ir kodo pertekliškumo analizė, informacijos perdavimo laikas, sistemos atitikimas funkciniams ir nefunkciniams reikalavimams.

3.3. Komponentų testavimas

Šiame punkte reiktų aprašyti kokiais metodais testavote smulkias programos dalis (žr. wiki *unit testing*). Komponentų testavimas privalo būti atliekamas naudojant automatines testavimo priemones.

3.4. Integracinis testavimas

Jei kurdami sistemą atlikote integracinį testavimą, jį aprašykite šiame skyrelyje. Integracinis testavimas privalo būti atliekamas naudojant automatines testavimo priemones.

3.5. Vartotojo sąsajos testavimas

Šiame punkte reiktų aprašyti kokiais metodais testavote vartotojo sąsają. Dažniausiai pasitaikantis metodas – „rankinis“, t.y. kai sąsaja testuojama vartotojui (testuotojui) bandant atsitiktinai ar pagal scenarijų spaudyti mygtukus, įvedinėti tekstą į laukus ir kt. Kur kas geresnis variantas tuomet, kai testuojama automatiškai – pavyzdžiui, sukuriama programa ar testavimo tvarkyklė, kuris spaudymo ar įvedimo veiksmus atlieka be vartotojo įsikišimo. Panaudotas automatinis testavimas, dažniausiai, papildomai (teigiamai) įvertinamas baigiamojo darbo gynimo metu. Pasinaudokite automatizavimo priemonėmis, tokiomis kaip **Selenium IDE**.

4. Dokumentacija naudotojui

Dokumento dalis, skirta naudotojui, kur aprašomas visas naudotojui aktualus programinės (aparatūrinės) įrangos funkcionalumas (4 – 10 lapų).

Dokumentacija naudotojui – tai instrukcija kaip naudotis sistema. Dokumentacijoje turi būti aiškiai aprašyti naudojimosi sistema ypatumai, pradedant diegimu ir baigiant įprastinėmis funkcijomis. Rašydami dokumentaciją atsižvelkite į naudojamą terminologiją. Pavyzdžiui, jei sistemą instaliuos administratorius, o naudos paprasti vartotojai, pastarųjų stenkitės neapkrauti sudėtingesnėmis sąvokomis.

4.1. Apibendrintas sistemos galimybių aprašymas

Sistemos galimybės nuo reikalavimuose aprašyto funkcionalumo skiriasi tuo, kad ne visiems vartotojams būtina žinoti technines projekto detales. Pavyzdžiui, internetinio portalo vartotojui svarbu žinoti kokios naudingos funkcijos yra portale (pvz., paieška, naujienlaiškio prenumerata ir kt.), tačiau ne visos funkcijos įprastam vartotojui yra aktualios (pvz., reklamos skydelių palaikymas, SSL protokolas vartotojų autentifikacijai ir t.t.).

4.2. Vartotojo vadovas

Vartotojo vadovas yra neformalus įvadas į sistemą, aprašantis jos „normalų“ vartojimą. Kitaip tariant, vartotojui draugiška instrukcija su daug iliustracijų ir paaiškinimų. Neišvengiamai pradedantieji, nepriklausomai nuo patirties, daro klaidas. Lengvai randama informacija, kaip nuo šių klaidų grįžti prie naudingo darbo ir atstatyti galimus klaidų padarinius, turi būti sudėtinė šio dokumento dalis.

4.3. Diegimo vadovas

Sistemos diegimo dokumentas yra skiriamas sistemos administratoriams (dažniausiai tai kompiuterius prižiūrintis personalas, tačiau šie žmonės nebūtinai būna ir sistemos naudotojai). Jame turi būti nurodytos diegimo konkrečioje aplinkoje detalės, turi būti supažindinama su sistemą sudarančiais failais, minimalia reikalingos techninės įrangos konfigūracija.

4.4. Administravimo vadovas

Sistemos administratoriaus vadove turi būti aprašyti pranešimai, kaip sistema bendrauja su kitomis sistemomis ir kaip reaguoti į šiuos pranešimus. Būtų gerai nurodyti, kaip reaguoti į sistemos klaidas (sisteminių pranešimų paaiškinimai). Jei sistema apima ir techninę įrangą, jame turi būti aprašyti operatoriaus veiksmai palaikant šią techninę įrangą (pvz., kaip prijungti naujus periferinius įrenginius ir t.t.).

Rezultatai ir išvados

Bene svarbiausia viso darbo dalis – išvados. Išvados nenurodo, kas buvo padaryta darbe, bet pabrėžia atrastus dėsningumus, pastebėtas technologijų ar rinkos spragas, esminius įrangos privalumus. Išvados gali būti formuluojamos tik darbo metu sukurtos įrangos, technologijos, metodo ar susistemintos informacijos pagrindu (pvz., negalima cituoti šaltinių, vadovautis kitų autorių atrastais dėsningumais). Išvados numeruojamos, jų turėtų būti maždaug 4-9 (pvz., kiekvienam kūrimo etapui – reikalavimų analizei, projektavimui, realizacijai, testavimui, diegimui). Įprastai kiekviena išvada turėtų būti sudaryta iš atlikto veiksmo aprašymo ir gautų rezultatų. Išvadas galima gauti:

- Atlikus konkurentų analizę, kuomet būna išsiaiškinama esminiai konkurentų sistemų pranašumai ir trūkumai (pvz., „Buvo išanalizuotos analogiškos (konkrečiai nurodant kokios) sistemos, kurios pasižymėjo tokiais ir tokiais privalumais (apibendrintai), tačiau dėl tokių ar anokių trūkumų buvo nuspręsta kurti naują sistemą...“).
- Atlikus technologijų analizę, kuomet būna pagrindžiamas konkrečių programavimo kalbų, karkasų ar kitų technologijų pasirinkimas (pvz., „Išanalizavus x, y ir z technologijas buvo pasirinkta technologija z. Tai padėjo lengviau suprojektuoti, o vėliau ir realizuoti įrankio serverio pusės dalį, palaikyti vientisą programos kodo struktūrą...“).
- Atlikus testavimą, kuomet būna nurodoma kokį kodo padengimą pavyko pasiekti, kokias klaidas pavyko aptikti panaudojus pasirinktus testavimo metodus.
- Susidūrus su tam tikromis specifinėmis problemomis, kurioms išspręsti buvo panaudotas jūsų sugalvotas metodas („Kūrimo metu buvo susidurta su tokiomis ar anokiomis problemomis, kurios buvo sprendžiamos taip arba anaip...“). Galima įdėti ir išvadą apie nepasiteisinusius, tačiau jūsų išbandytus sprendimus (siekiant, kad kiti „neliptų ant to paties grėblio“). Jūsų parinkti problemų sprendimo būdai yra svarbios išvados, parodančios jūsų kompetenciją ir įsigilinimą į darbą.
- Realizavus pačią programą ar sistemą, kuri (greičiausiai) pakeitė ar pagerino iki tol vykusius verslo procesus (tai susiję su skyreliais „Bendras veiklos tikslas“ ir „Sistemos pagrindimas“) ar (jei tai buvo mokslinio pobūdžio darbas) tiesiog iki tol buvusius algoritmo / sprendimo rezultatus.

Šiame skyrelyje taip pat būtina pridėti ir papildomas išvadas-rezultatus apie tai:

- Kokia yra sistemos esama būklė. Verta paminėti, jei sistema yra praktiškai naudojama įmonėje ar (programėlės kūrimo atveju) programėlė yra įkelta į Google Play ar AppStore parduotuves.
- Kas planuojama atlikti tobulinant sistemą ateityje. KAdangi baigiamajam darbui sukurti yra skiriamas ribotas laikas, galbūt verta paminėti tas savybes, kurių dėl laiko apribojimų tiesiog nespėjote, bet planuojate įgyvendinti.

Darbe naudotos literatūros sąrašas (1 – 3 lapai). Sąrašas sudaromas vadovaujantis ISO 690 priimtu literatūros sąrašo ir citavimo stiliumi [**rastodarbai**]. Kaip sudarinėti literatūros sąrašą LaTeX priemonėmis, galima pasiskaityti `https://www.overleaf.com/learn/latex/bibliography_management_with_biblatex`.

Literatūros sąrašas turėtų apimti visus naudotus šaltinius. Literatūros šaltiniai pateikiami sunumeruoti citavimo tvarka. Darbo apraše turi būti pacituoti visi naudoti šaltiniai, pateikiant tekste nuorodas. Daugiau informacijos apie bendras citavimo taisykles galite rasti `https://biblioteka.ktu.edu/mokymai/#mokymosi-medziaga` „Kaip cituoti šaltinius ir parengti literatūros sąrašą. ISO 690:2010 standartas (skirta technologijos mokslams)“.

Ši pastraipa skirta nuorodoms į literatūros sąrašą: [**litnet**], [**ontologies**], [**valstybestarnyba**], [**spaudosdraudimas**], [**krastoapsauga**], [**citationguide**], [**velomobilis**].

Priedai

1. **Priedo pavadinimas**
2. **Antras priedas**