TARTU ÜLIKOOL
LOODUS- JA TEHNOLOOGIATEADUSKOND
Tehnoloogiainstituut
Arvutitehnika eriala

Tõnis Terasmaa

NÕUDED ROBOTMANNEKEENIDE FOTOSTUUDIO TEHNILISE TOE TARKVARALE

Bakalaureusetöö (12 EAP)

Juhendaja: Arvutitehnika programmijuht
Aleksander Tõnnisson

Luba kaitsmisele:

Juhendaja: ....."....." mai 2013

Programmijuht: ....."....." mai 2013

Tartu 2013

SISUKORD

SISSEJUHATUS

Koostöös Fits.Me'ga [1] on Tartu Ülikooli Tehnoloogiainstituudil [2] (edaspidi TÜTI) käsil projekt, mille eesmärgiks on arendada robotmannekeen [3] virtuaalse proovikabiini [4] tarbeks. Robotmannekeen (edaspidi mannekeen) on inimese ülakeha kujuga robot, mis suudab jäljendada erinevate mõõtudega inimeste ülakehasid. Mannekeeni juhtimiseks ja pildistamiseks on välja töötatud programmid, kuid mannekeenidel ilmnevatest vigadest teavitamine, nende vigade parandamise jälgimine ja mannekeenide hooldamine toimub manuaalselt.

Mannekeenide täiustamiseks on vaja arendusmeeskonnal koguda andmeid mannekeenidel ilmnevate probleemide kohta nii, et neid oleks võimalik hallata, sorteerida ja vaadelda statistilisest vaatepunktist. Kogutavat statistikat analüüsides on võimalik tuvastada ja likvideerida mannekeenide tüüpvead - olgu need seotud mehaanika, elektroonika või mannekeeni kattematerjalidega.

Mõistmiseni, et mannekeeni kohta ei koguta piisavalt andmeid, aitas jõuda mannekeenide fotostuudio (edasi stuudio) ja arendusmeeskonna poolne tagasiside. Stuudios on probleemiks, et ei teata, miks on mannekeen katki ning kui kaua selle parandamine aega võtab. Samuti ei osata anda stuudiost rikke korral piisavalt korrektseid veateateid. Mõne vea ilmnemisel ei peeta seda piisavalt tähtsaks, et tööd katkestada, kuid ei arvestata, et peale rikke teket töö jätkamine on mannekeenile kahjulik. Arendusmeeskond seevastu vajab andmeid tegevuste kohta, mida mannekeenidega tehti neil hetkedel, kui probleemid ilmnesid.

Tulenevalt sellest probleemist sai püstitatud lõputöö eesmärk koostada nõuded ja ülesehitus tarkvarale, mille:

- primaarseks ülesandeks on koguda statistilist informatsiooni mannekeeni töökindluse ning mannekeenide tehnilise toe tööefektiivsuse kohta;
- sekundaarseks ülesandeks on parandada tehnilise toe tööefektiivsust, vähendades robotmannekeenide hooldamisele kuluvat aega.

Töö on jagatud neljaks peatükiks. Esimeses peatükis räägitakse mannekeeni regulaarsest hooldusest, praegustest veateadete kogumise meetoditest ja olemasolevatest

veajälgimissüsteemidest [5]. Teises peatükis kirjeldatakse tarkvarale ja andmebaasile esitatavad nõuded. Kolmandas peatükis tutvustatakse tarkvaraarenduse seisu lõputöö esitamise kuupäevaga. Viimases peatükis on lühidalt räägitud tarkvara edasistest arendusvõimalustest.

1. ÜLEVAADE OLEMASOLEVATEST LAHENDUSTEST

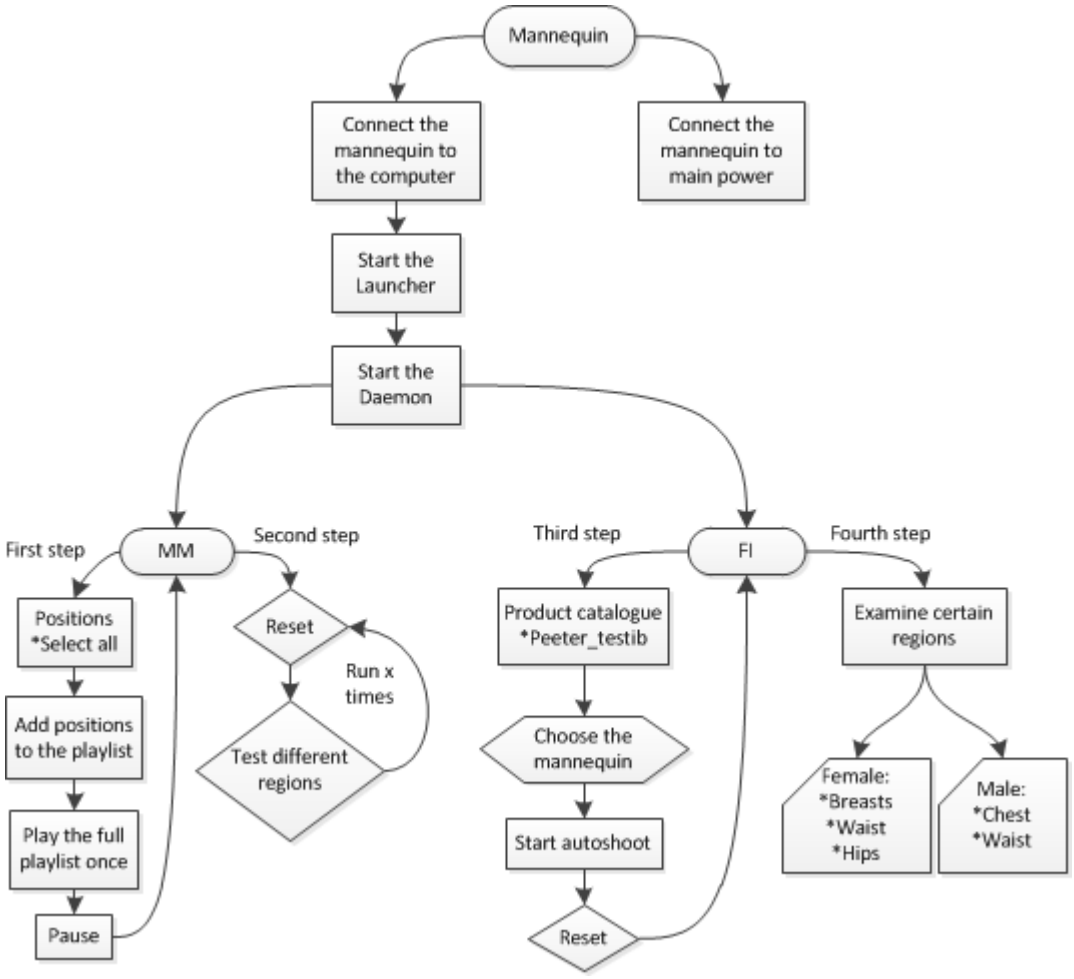
Selleks. et kaardistada probleemid kasutuses olevas veateatamise süsteemis, analüüsiti mannekeenidele teostatavat regulaarset hooldust, vigadest teatamise meetodit ning uuriti olemasolevate veajälgimissüsteemide kohta.

1.1. Mannekeenide regulaarne hooldus

Nagu kõigil elektrooniliste ja liikuvate mehaaniliste osadega masinatel, tuleb ka mannekeenidel läbi viia regulaarseid hooldustöid. Mannekeene tuleb hooldada iga kahe nädala möödudes. Põhilised probleemid, mis mannekeeni kasutamisel tekivad, on kattematerjali või aktuaatoritega seotud. Aktuaator on objekt, mis muudab energia mehaaniliseks liikumiseks. Pärast mannekeeni jaoks maksimaalse suurusega kehakuju võtmist võib juhtuda, et kattematerjali ühtlast struktuuri hoidvad jõhvid kiiluvad kinni ning selle tagajärjel tekib ebakõla sisestatud ja mannekeeni poolt sisse võetud positsiooni vahel. Selle probleemi ilmnemisel mõjuvad aktuaatoritele jõud ebasoodsate suunavektoritega ning võivad põhjustada seeläbi aktuaatori kinnikiilumise.

Joonisel 1 on ära toodud regulaarse hoolduse käigus tehnilise toe poolt läbiviidavad sammud. Alustuseks tuleb hooldatav mannekeen ühendada vooluvõrku ja hoolduseks vajalike programmidega varustatud arvuti külge. Seejärel tuleb käivitada mannekeeni käsitsijuhtimise tarkvara *Mannequin Manager* [6] (edasi MM). MMi kasutades tuleb esmalt proovida läbi kõik erinevad kehakuju asendid. Sellele järgnevalt tuleb katsetada mõne kindla asendi võtmist mitu korda järjest, naastes pärast igat sooritust algasendisse. Pärast MMiga tehtud sammude läbimist tuleb käivitada Fits.Me programmeerijate poolt loodud mannekeenide pildistamise tarkvara FI (tuletatud lühendina Fits.Me nimest). FId kasutades tuleb testida automaatset pildistamisrežiimi ning mõõta, kas mannekeeni mõõdud vastavad soovitud mõõtudele.

Antud lõputöös on joonised ja tabelid inglise keeles, sest arendustöö käib rahvusvahelises keskkonnas.



Joonis 1. Stuudio tehnilise toe töö kulg regulaarse hoolduse korral

1.2. Kasutusel olev veajälgimissüsteem

Hetkel rakendatava veajälgimissüsteemi kasutamine ja haldamine on tülikas ning aeganõudev, sest kõike tuleb teha manuaalselt. Tööprotsess kulgeb järgmiselt:

1. stuudiost raporteeritakse probleemidest meili teel;
2. projektijuht sorteerib meilidest välja reaalsed vead;
3. projektijuht teeb veateadete kohta sissekanded vigade nimekirja (ingl k *buglist*);
    a. vajadusel tehakse stuudiole täiendavad infopäringud vigade kohta;
4. kui sissekanded on täiendatud infoga, lisab projektijuht vastavad ülesanded (ingl k *task*) ülesannete nimekirja (ingl k *tasklist*);

a. projektijuht määrab *taskile* veatüübi ja lahendaja (tehniline tugi või arendusmeeskond);

b. lahendajad saavad tekstiformaadis märkida üles oma progressi ülesande lahendamisel, kui palju aega on seni lahendamisele kulutatud ning lõpliku parandamise järel märkida probleem lahendatuks;

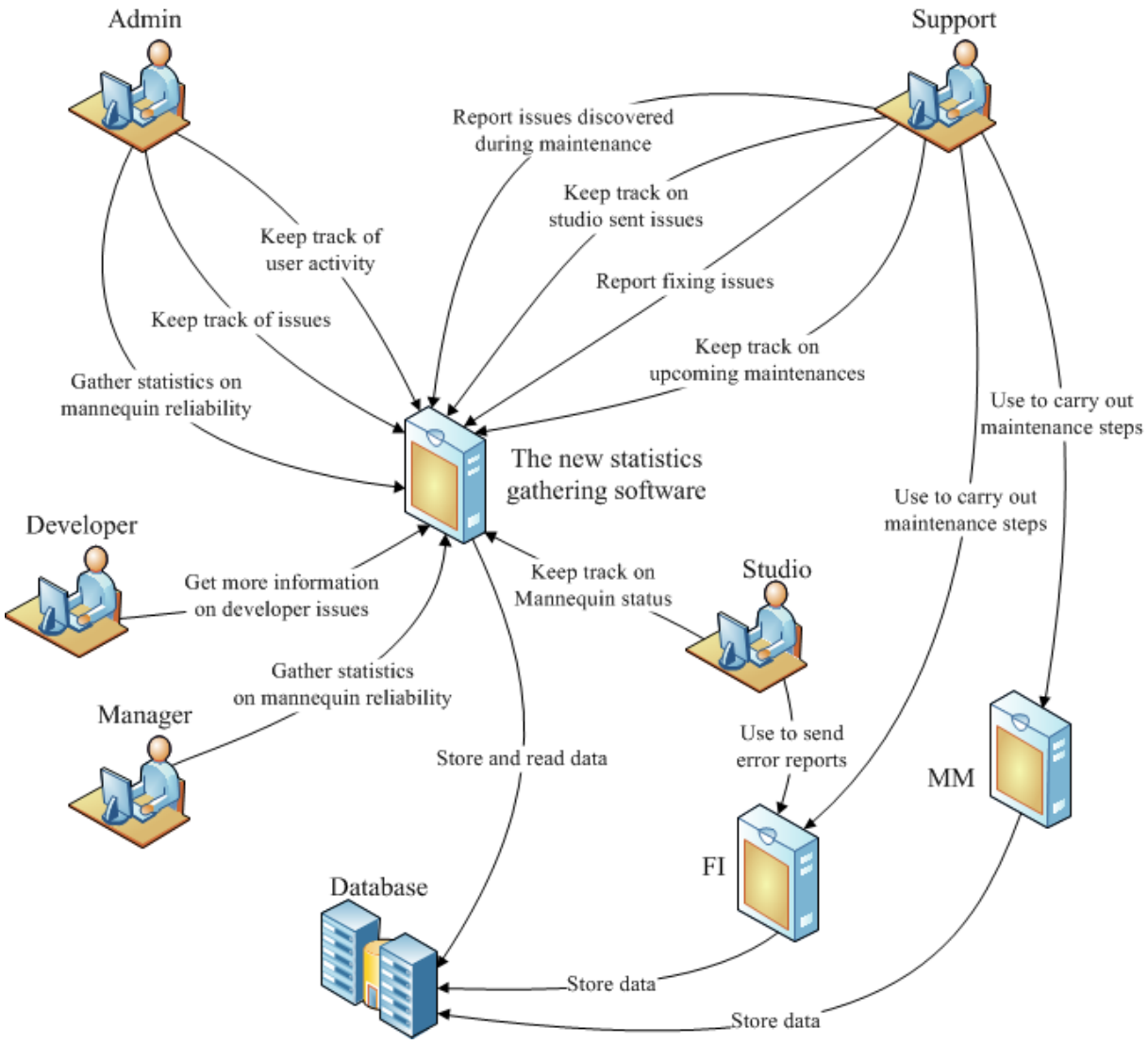
c. *taski* avamisest ja sulgemisest on võimalik automaatselt meili teel teavitada nii *taski* määrajat kui ka lahendajat.

Lisaks stuudiost saadetud veateadetele kogutakse teavet ka automaatselt. MM salvestab logisse mannekeenidelt tulnud tarkvara ja riistvara veateated ning silumise (ingl k *debug*) informatsiooni. FI logib oma tegevuse ja veateated, mida mannekeeni madalataseme tarkvara edastab.

1.3. Olemasolevad veajälgimissüsteemid

Esmalt tuli koostada üldine skeem (Joonis 2), mida programm peaks võimaldama ja suutma. Sellele järgnevalt tuli uurida, milliseid veajälgimissüsteeme leidub ning kas need on rakendatavad antud probleemi lahendamiseks. Lõpuks pidi hindama, kas on mõttekam maksta sellise programmi litsentsi eest, mis täidab mõned nõuded või programmeerida ise programm, mis vastaks täpselt seatud nõuetele.

Uurimisel leiti, et olemasolevatest veajälgimissüsteemidest enamik sisaldab automatiseeritud teavitamisteenust, integreeritud wikit [7] ning mõni sisaldab ka integreerimise võimalust versioonihaldussüsteemidega [8], nagu Subversion, Git ja Mercurial. Andmete jälgimine ja sisestamine toimub interneti vahendusel ning teavitused vigadest ja nende parandamisest saadetakse meili teel. Põhilised kasutaja autentimismeetodid on veebivormi või LDAP [9] (Lightweight Directory Access Protocol) kasutamine.



Joonis 2. Esialgne soovitud tööprotsess

On palju erinevaid veajälgimissüsteeme, näiteks Apache Bloodhound, Bugzilla, Google Code Hosting, JIRA, Launchpad jt. Neist kõige tuntum on JIRA [10]. Lähemal uurimisel selgus, et JIRA peamiseks rivaaliks on TrackStudio [11]. Mõlemad on veebirakendused [12], mis kirjutatud Java programmeerimiskeeles, ning toetavad integratsiooni tarkvara arenduskeskkondadega [13] nagu Eclipse [14] või IntelliJ IDEA [15]. Need võimaldavad raporteerida vigadest ja neid ka hallata. Tarkvaraarenduse meeskondade projektijuhid saavad jälgida töötajate progressi ja määrata ülesandeid lahendamiseks. Tarkvaraarendajad saavad märkida üles, mis nad antud probleemide lahendamiseks on teinud.

TrackStudio võimaldab sarnaselt JIRAle veahaldust ja projektijuhtimist, kuid lisaks võimaldab TrackStudio dokumentide ja failide jagamist ning haldamist. Kasutajaid saab jagada erinevatesse kasutajagruppidesse ning on võimalik anda õigusi kindlatele kasutajatele kindlate failide vaatamiseks. TrackStudio plussiks on kasutuslitsentsi maksumus, mis on võrreldes JIRA litsentsiga palju soodsam. Miinuseks on see, et TrackStudiot uuendatakse vaid korra aastas, JIRAt kord kvartalis.

Kahjuks olemasolevad veajälgimissüsteemid on suunitletud tarkvaraarenduse haldamiseks, mitte mehaanika ja riistvara, tarkvara ning nende vahelise suhtluse vigade haldamiseks, hooldamiseks ega statistika kogumiseks. Seetõttu võeti vastu otsus, et programm tuleb ise välja arendada.

2. TARKVARA ÜLESEHITUS

Eelnevalt kirjeldatud probleemi lahendamiseks otsustati koostada veebirakendus, mille arendamiseks kasutati tarkvara arendamise *waterfall* [16] mudelit, tagades kõige optimaalsema tööjärjekorra antud probleemi lahendamiseks. Enne programmeerimise alustamist analüüsiti põhjalikult vajaminevat funktsionaalsust. Teised mudelid lubavad programmeerida paralleelselt nõuete seadmisele, mida antud juhul ei peetud kasulikuks.

2.1. Nõuded veebirakendusele ja kasutajaliidesele

Koostatav tarkvara peab andmebaasist küsima andmeid veateadete, mannekeenide ja hooldustega seonduva kohta, kuid peab ka võimaldama muuta hoolduste ja veateadetega seonduvaid andmeid. Näiteks aktuaatori kinnikiilumise korral saadab madala taseme tarkvara veateate, mis salvestatakse andmebaasi. Hoolduse käigus tuvastatud mehaanika viga või katkine detail tuleb hooldust läbiviival isikul samuti andmebaasi üles märkida. Sellele lisaks peab tarkvara kuvama peatselt saabuvaid regulaarseid hooldusi mannekeenidele.

Selleks et programm oleks kergesti kasutatav kõigile, kes mannekeenidega töötavad, peab andmete kuvamiseks, sisestamiseks ja muutmiseks olema graafiline kasutajaliides. Veebirakendus peab olema kättesaadav kasutades internetiga varustatud arvutit. On vajalik, et programmi kasutamiseks ei peaks kasutaja midagi alla laadima ega paigaldama. Rakenduse kasutamine peab olema lihtne.

Mannekeeni arendamise tarbeks on eelnevalt loodud programme kasutades selleks C# programmeerimiskeelt. Ühilduvuse säilitamiseks otsustati luua programm ASP.NET [17] rakendusena ning tarkvaraarendus platvormiks valiti Windows. ASP.NET on serveripoolne veebirakenduse raamistik, mis käitab mistahes .NET programmeerimiskeeles kirjutatud koodi. ASP.NET rakendusi on soovitatav käitada Windows serveristes, sest tegu on Microsofti [18] arendatud raamistikuga. ASP.NETi on võimalik käitada ka Linux [19] serverites.

Uue andmebaasi skeemi koostamise käigus määrati kindlaks, millist informatsiooni on vaja kõige rohkem vaadelda. Seetõttu hakatakse graafilises kasutajaliideses andmeid kuvama vastavalt sellele, millisesse kasutajagruppi kasutaja kuulub. Tarkvaral saab olema viis

põhilist kasutajagruppi: *admins*, *developers*, *supervisors*, *supports* ja *managers*. Administraatoriteks on projektijuhid, neil on ligipääs kõikjale, ainuõigus registreerida uusi kasutajaid ja määrata kasutajate gruppidesse kuuluvust. Arendusmeeskonna liikmed kuuluvad gruppi *developers*. Staažikamad tehnilise toe liikmed kuuluvad gruppi *supervisors* ning ülejäänud tehnilise toe liikmed gruppi *support*. Gruppi *managers* kuuluvad isikud, kellel on vaja näha statistikat vigade esinemissageduse ja tööefektiivsuse kohta. Kasutajad, kes ei kuulu ühessegi kasutajagruppi, sealhulgas stuudio töötajad, näevad ainult mannekeenide staatust.

Veebirakenduse koostamisel otsustati kasutada MVC [20] (lühend tuleneb inglise keelest - *Model-View-Controller*) tarkvara arhitektuuri kujundamise mustrit [21], kus programm on liigendatud kolme põhilisse kihti:

- *Model* hoiab ja uuendab andmeid, mida *View* kuvab ning *Controller* muudab;
- *View* kuvab kasutajale andmeid, mis asuvad *Modelis* ning lubab kasutajal sisestada omapoolseid andmeid, mille haldamisega tegeleb *Controller*;
- *Controller* manipuleerib *Modelis* asuvaid andmeid vastavalt sellele, mida kasutaja *Views* on sisestanud või teinud.

MVC võimaldab muuta andmeid ja vaateid ilma, et muutused ühes mõjutaksid teist. Koostatava rakenduse puhul on vaja kasutajale andmeid kuvada vastavalt tema gruppidesse kuuluvusele, mida on lihtne saavutada erinevate *Viewde* defineerimisel. Tarkvaraarenduskeskkonnaks valiti Microsoft Visual Studio [22] (edasi VS) kahel põhjusel. Esiteks on VSis olemas *template* [23] MVC programmeerimiseks. Teiseks on VSis integreeritud võimalus jooksvalt testida koostatavat veebirakendust kogu mahus. Käivitatakse virtuaalne server, kuhu luuakse vastav rakendus ja näidis andmebaas, mida on võimalik modifitseerida vastavalt vajadusele. Seega on ka võimalik testida suhtlust andmebaasiga.

Et kasutada MVCd korrektselt, oli vaja koostada tabel toomaks välja kõik erinevad *Modelid*, *Viewd* ja *Controllerid*, mida veebirakendus peab sisaldama. (Tabel 1.)

Tabel 1. MVC tarkvara arhitektuuri kujundamise mustri tarvis koostatud tabel, kus on näidatud kõik vajaminevad struktuuriüksused

	Model		View		Controller	
Account	LogOn		LogOn		LogOn	
	LogOff		Register		LogOff	
	Register		ChangePassword		Register	
	ChangePassword		ChangePasswordSuccess		ChangePassword	
					ChangePasswordSuccess	
					ErrorReport	
Issue	CommonIssues	Hotfix	Issues	Collapsed	SortIssue	
	PbIssues	Resolved		Expanded	FixedResolved	
	PbImages	IssueData				
	MaintIssues					
	SendToDev					
Activity	CommonIssues		UserActivities		Issues	Hotfix
	Maintenance		ToBeResolved			Resolved
					Maintenances	
Maintenance	Maint		MaintenanceSchedule		MaintSchedule	CalcSchedule
	Mann		PerformMaintenance		StartMaint	Start
	Maint_Issue					Step
	Maint_Step					Pause
	Maint_Step_Result					Resume
	Mann_Type_Has_Maint_Steps					Done
	Mann_type				(Need to log time per step)	
	User					
Report	Maintenance		ReportWindow		ReportWindow	ReportInfo
	CommonIssues					SelectDetails
	Maint_Issue					SubmitReport
Search	CommonIssues		SearchWindow		SearchWindow	EditEntry
	Maintenance		EditWindow			SearchDetails
	Maint_Step_result					
	User					
	PbIssue					
	MaintIssue					
	Mannequin					
Home			Index		Home	PageNavigation
			About			
			Search			

		Report		
		Maintenance		
		UserActivities		
		Statistics		
		Issues		
Statistics	CommonIssues	ShowStatistics	Statistics	CalculateStatistics
	Maintenance			GenerateGraphs
	Maint_Step_result			GenerateTables
	User			SelectData
	PbIssue			
	MaintIssue			
	Mannequin			

2.2. Nõuded andmebaasile

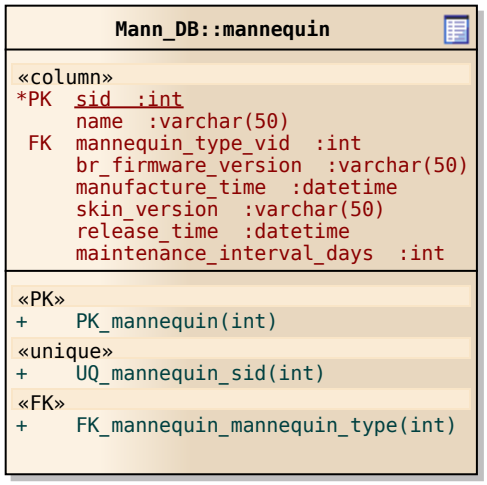
Kõige suuremat rolli andmete talletamisel ja statistika tegemisel omab andmebaasi struktuur. Lisaks veebirakenduse nõuete seadmisele, tuli ka olemasolev andmebaas uuendada, et tulevikus oleks sinna kerge talletada andmeid nii, et nende vahel oleks võimalikult lihtne luua sidemeid. Samuti on tarvilik, et andmeid oleks kerge andmebaasi sisestada ja sealt lugeda. Andmebaasi skeem koostamiseks pandi kirja teadaolevalt kõik andmed, mida ning millisel kujul (nt: *integer*, *string*, *varchar* jne) on vaja säilitada. Andmebaasi skeemi loomisel tuli arvestada sellega, et andmebaas oleks võimalikult modulaarne ja detailne, juhuks kui tulevikus on tarvis lisada juurde uusi väljasid või tabeleid.

Andmebaasi skeemi koostamiseks kasutati tarkvara Enterprise Architect [24] (edasi EA), sest EAga on võimalik genereerida fail, kuhu salvestatakse kõik SQL [25] (ingl k *Structured Query Language*) päringud, mida on tarvis, et andmebaasi sisestada kõik vajalikud tabelid ja nende vahelised seosed. Samuti on EAd kasutades mugav dokumenteerida tehtud tööd.

Uusim Microsofti andmebaasi struktuur, mille jaoks EA suudab koodi genereerida on SQL Server 2008 [26]. Sellest tulenevalt peab kasutatav andmebaas olema SQL Server 2008 tüüpi ning andmebaasis peavad kindlasti olema järgnevad tabelid:

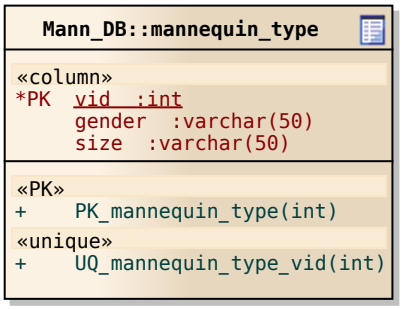
- Mannequin (Joonis 3) – Selles tabelis hoitakse mannekeeni eksemplarile omaseid andmeid: sid (seeria id; peamine võti), name (mannekeeni nimi),

mannequin_type_vid (mannekeeni tüübi versiooni id; väline võti), br_firmware_version (*bridge* plaadi püsivara versioon), manufacture_time (mannekeeni eksemplari tootmise aeg), skin_version (mannekeeni katte versioon), release_time (mannekeeni eksemplari väljastamise aeg), maintenance_intervals _days (mannekeeni regulaarse hoolduse intervalli periood päevades);



Joonis 3. Andmebaasi tabel "mannequin"

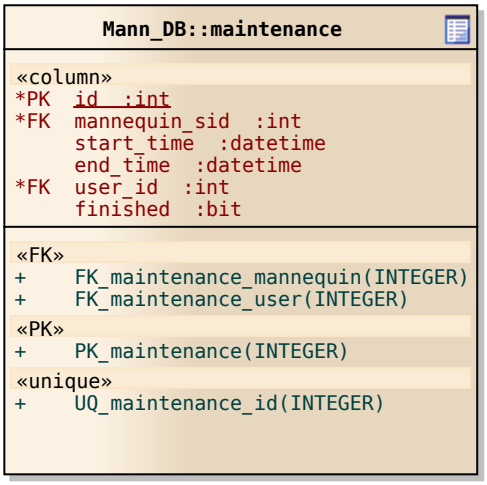
- Mannequin_type (Joonis 4) – Selles tabelis hoitakse mannekeeni tüübile omaseid andmeid: vid (versiooni id; peamine võti), gender (mannekeeni esindatav sugu), size (mannekeeni esindatav kehakuju suurus);



Joonis 4. Andmebaasi tabel "mannequin_type"

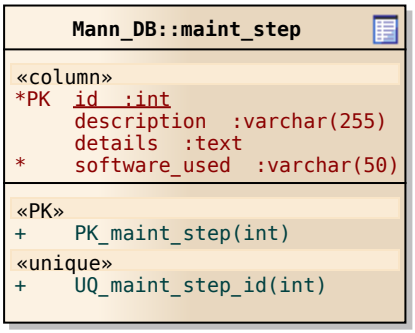
- Maintenance (Joonis 5) – Selles tabelis hoitakse mannekeni hooldusele omaseid andmeid: id (hoolduse id; peamine võti), mannequin_sid (hooldatava mannekeeni seeria id; väline võti), start_time (hoolduse alustamise aeg), end_time (hoolduse

lõpetamise aeg), user_id (hooldust läbiviiva kasutaja id; väline võti), finished (olek, kas hooldus on lõpule viidud või mitte);



Joonis 5. Andmebaasi tabel "maintenance"

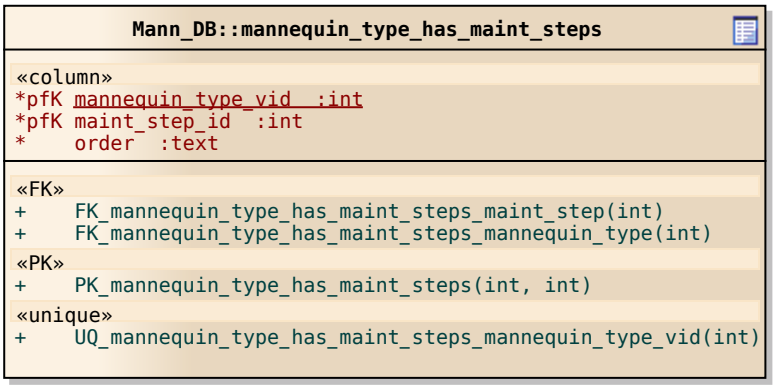
- Maint_step (Joonis 6) – Selles tabelis hoitakse hoolduse sammudele omaseid andmeid: id (hoolduse sammu id; peamine võti), description (hoolduse sammu lühikirjeldus), details (hoolduse sammu täpsed juhised), software_used (hoolduse sammu läbiviimiseks vajalik tarkvara);



Joonis 6. Andmebaasi tabel "maint_step"

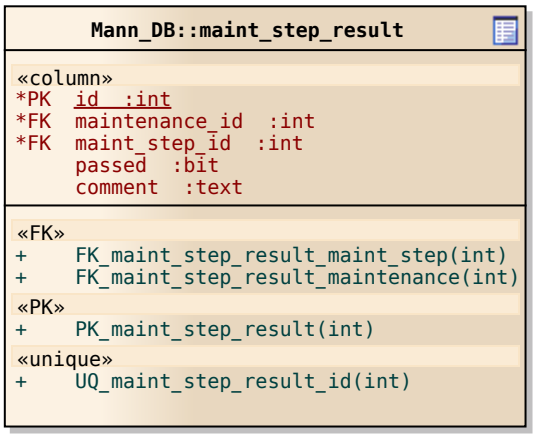
- Mannequin_type_has_maintenance_steps (Joonis 7) – See on automaatselt genereeritud lisatabel, et väljendada mitu-mitmele suhet kahe eelneva tabeli vahel. See tabel sisaldab andmeid selle kohta, millisel mannekeeni tüübil milliseid hooldussamme peab läbi tegema: mannequin_type_vid (mannekeeni tüübi versiooni

id; peamine väline võti), maint_step_id (hoolduse sammu id; peamine väline võti), order (sammude läbimise järjekorra kirjeldus);



Joonis 7. Andmebaasi tabel "mannequin_type_has_maint_steps"

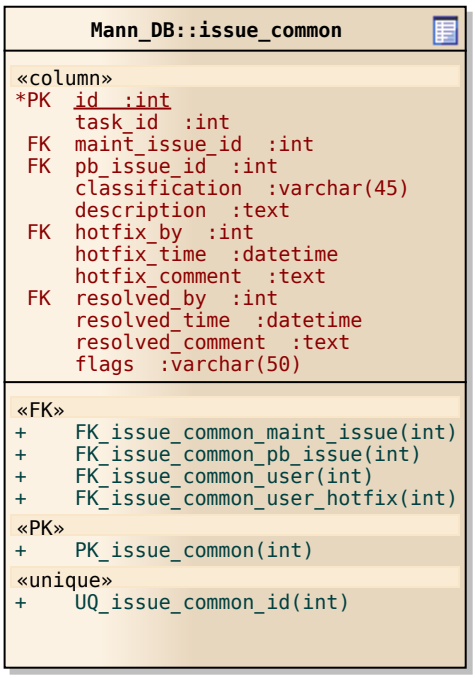
- Maint_step_result (Joonis 8) – Selles tabelis hoitakse hoolduse sammu tulemusele omaseid andmeid: id (hoolduse sammu tulemuse id; peamine võti), maintenance_id (hoolduse id; väline võti), maint_step_id (hoolduse sammu id; väline võti), passed (olek, kas samm lõpetati edukalt või mitte), comment (kommentaar sammu tulemuse kohta);



Joonis 8. Andmebaasi tabel "maint_step_result"

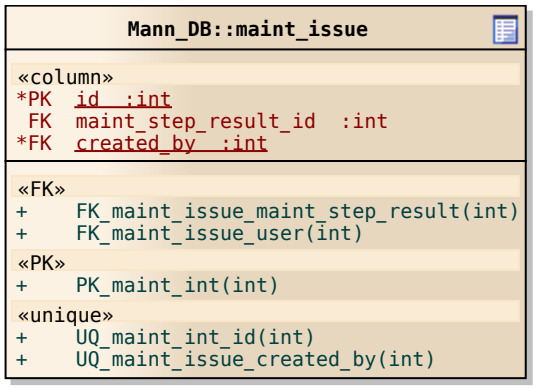
- Issue_common (Joonis 9) – selles tabelis hoitakse veale omaseid andmeid: id (vea id; peamine võti), task_id (vea lahendamiseks tekitatud taski id), maint_issue_id (kui tegu on hoolduse käigus ilmnenud veaga, siis tuleb siia vastava veateate id, vastasel

juhul jäetakse väli tühjaks; väline võti), pb_issue_id (kui tegu on stuudio poolt saadetud veateatega, siis tuleb siia vastava veateate id, vastasel juhul jäetakse väli tühjaks; väline võti), classification (vea klassifikatsioon), description (veateate kirjeldus), hotfix_by (kiirparanduse läbiviinud kasutaja id; väline võti), hotfix_time (kiirparanduse läbiviimise aeg), hotfix_comment (kommentaar läbiviidud kiirparanduse kohta), resolved_by (vea täieliku parandamise läbiviinud kasutaja id; väline võti), resolved_time (täieliku parandamise läbiviimise aeg), resolved_comment (kommentaar läbiviidud täieliku paranduse kohta), flags (bitijada, kus iga bitt määrab, kas mingi omadus on tõene või väär);



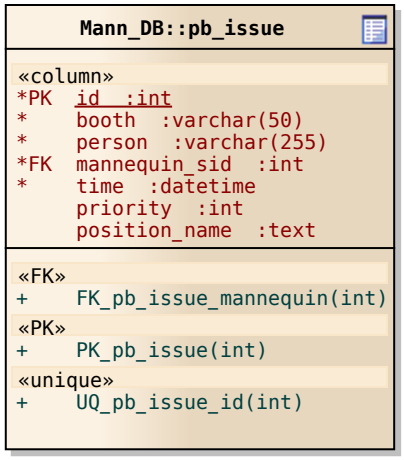
Joonis 9. Andmebaasi tabel "issue_common"

- Maint_issue (Joonis 10) – selles tabelis hoitakse hoolduse käigus ilmnenud vigadele omaseid andmeid: id (hoolduse veateate id; peamine võti), maint_step_result_id (hoolduse käigus ebaeduka tulemuse saanud sammu id; väline võti), created_by (hoolduse veateate sisestanud kasutaja id; väline võti);



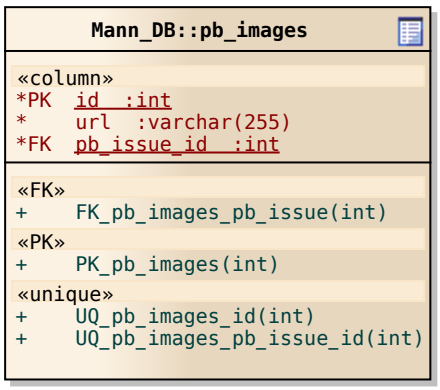
Joonis 10. Andmebaasi tabel "maint_issue"

- Pb_issue (Joonis 11) – selles tabelis hoitakse stuudio poolt saadetud veateadetele omaseid andmeid: id (stuudio poolt saadetud veateate id; peamine võti), booth (stuudio boksi nimi, kust veateade saadeti), person (veateate edastanud isiku nimi), mannequin_sid (boksis kasutatava mannekeeni seeria id; väline võti), time (veateate edastamise aeg), priority (stuudio poolt defineeritud probleemi prioriteedi järk), position_name (vea ilmnemisel kasutatud positsiooni nimi);



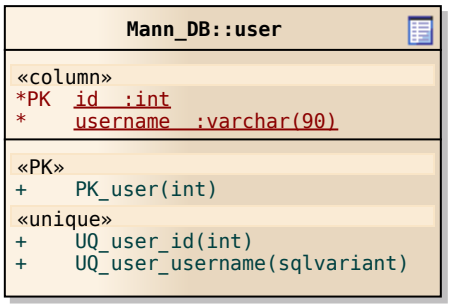
Joonis 11. Andmebaasi tabel "pb_issue"

- Pb_images (Joonis 12) – selles tabelis hoitakse stuudio poolt saadetud veateadetele lisatud piltidega seonduvad andmed: id (stuudio poolt tehtud piltide kogumi id; peamine võti), url (veebilink pildifailidele), pb_issue_id (stuudio poolt saadetud veateate id, millele pildid kaasa panid; väline võti);



Joonis 12. Andmebaasi tabel "pb_images"

- Users (Joonis 13) – selles tabelis hoitakse kasutajatele omaseid andmeid: id (kasutaja id; peamine võti), username (kasutaja nimi). Turvalisuse säilitamise eesmärgil ei ole lubatud selle tabeli kohta rohkem informatsiooni jagada.



Joonis 13. Andmebaasi tabel "user"

2.3. Suhtlus andmebaasiga

Peale MMi, FI ja tehnilise toe tarkvara, suhtlevad andmebaasiga veel teisedki robotmannekeeni arendusprotsessi käigus kasutusele võetud programmid, nagu ID programmeerimise tarkvara ja TestUnit.

ID programmeerimise tarkvara kasutatakse selleks, et programmeerida aktuaatoritele vastav püsivara ning vajalikud andmed, nagu selle keermekäigu pikkus või millise mannekeeni peal on aktuaator hetkel kasutuses jne.

TestUnit on käsurea tööriist, millega saab otse riistvarale käske saata ja ka riistvara kohta infopäringuid teha.

Et eraldada kogu andmebaasiga suhtlemise loogika muust loogikast, võetakse kasutusele spetsiaalselt selleks otstarbeks loodud andmebaasi programmiliides ehk DB API [27] (lühend tuleneb inglise keelest – *Data Base Application Programming Interface*). API sisaldab kindlaid meetodeid või funktsioone, mida programm välja kutsub, andes kaasa vajalikud sisendid ning seejärel küsib API meetod andmebaasist vastavad andmed ja tagastab need juba vajalikul kujul programmile. Sama toimub ka andmete andmebaasi kirjutamisel - programm kutsub välja API meetodi, andes kaasa vajalikud andmed ning API kirjutab seejärel antud andmed määratud andmebaasi tabelitesse.

2.4. Kasutusjuhud

Pärast andmebaasi skeemi koostamist ja graafilise kasutajaliidese kujundamist koostati tabel kõikide erinevate vigade klassifikatsioonidest, mis robotmannekeenide kasutamisel võivad esineda. Jõuti otsusele, et vead jagunevad nelja põhirühma: kasutaja vead, mehaanika vead, elektroonika vead ja tarkvara vead. Andes igale veateatele vajaliku veatüübi, on hiljem lihtsam statistilisi kokkuvõtteid teha (nt. kui palju esines kasutaja lohakusvigu või kui palju oli elektroonikaga seotud vigu jne).

Järgnevalt pandi paika peamised kasutusjuhud (Joonis 14). Tähtis oli läbi mõelda kes ja kuidas ning milleks antud programmi kasutama hakkavad. Programmi peamisteks kasutajateks on arendusmeeskonna projektijuhid, arendusmeeskond, tehnilist tuge pakkuvad isikud, stuudio töötajad, kes kasutavad mannekeene, ja Fits.Me esindajad.

Administraatoriteks on projektijuhid, kes kasutavad rakendust selleks, et vaadelda kogutud informatsiooni ning koguda erinevate mannekeenitüüpide statistilisi näitajaid erinevate vigade esinemissageduste kohta. Samuti võimaldab koostatav tarkvara koguda statistilisi andmeid tehnilise toe tööefektiivsuse kohta.

Arendusmeeskond kasutab rakendust selleks, et jälgida tekkinud vigasid ja vajadusel anda nõu tehnilisele toele probleemide lahendamisel. Samuti selleks, et lahendada tarkvara ja elektroonikaga seotud probleeme, mida tehniline tugi ei suuda lahendada.

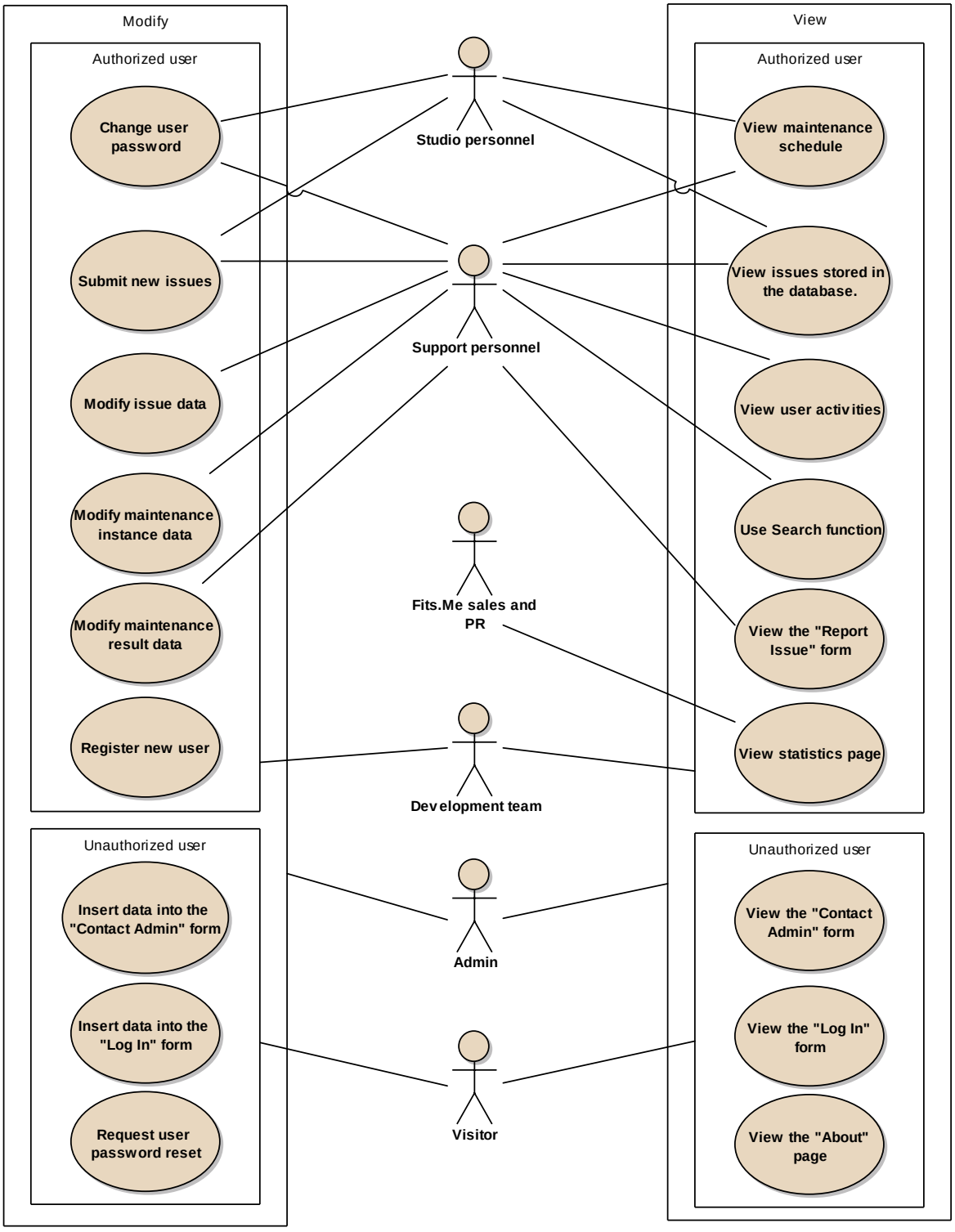
Tehniline tugi kasutab programmi, et märkida üles parandatud vead ning aja, mis sooritatud parandustöödele ja hooldustele kulus. Nagu nõuetes sai märgitud, kuvab rakendus tehnilisele

toele graafikut peatselt hooldust vajavatest mannekeenidest. Kui tehniline tugi alustab hooldust, siis rakendus kuvab täpsed juhised hoolduse läbiviimiseks ning salvestab andmebaasi hoolduse tulemused ja hoolduse sammude läbimiseks kulunud summaarse aja.

Stuudio töötajad kasutavad programmi, et jälgida, millal mingi kindel mannekeen jõuab hooldusest või parandusest tagasi stuudiosse ning on taas töövalmis.

Fits.Me juhatus kasutab programmi ainult mannekeenide tööefektiivsuse statistika kogumiseks.

Tuvastamata külastajatele on nähtav ainult sisselogimise lehekülg ning võimalus saata administraatorile päring kasutaja loomiseks.

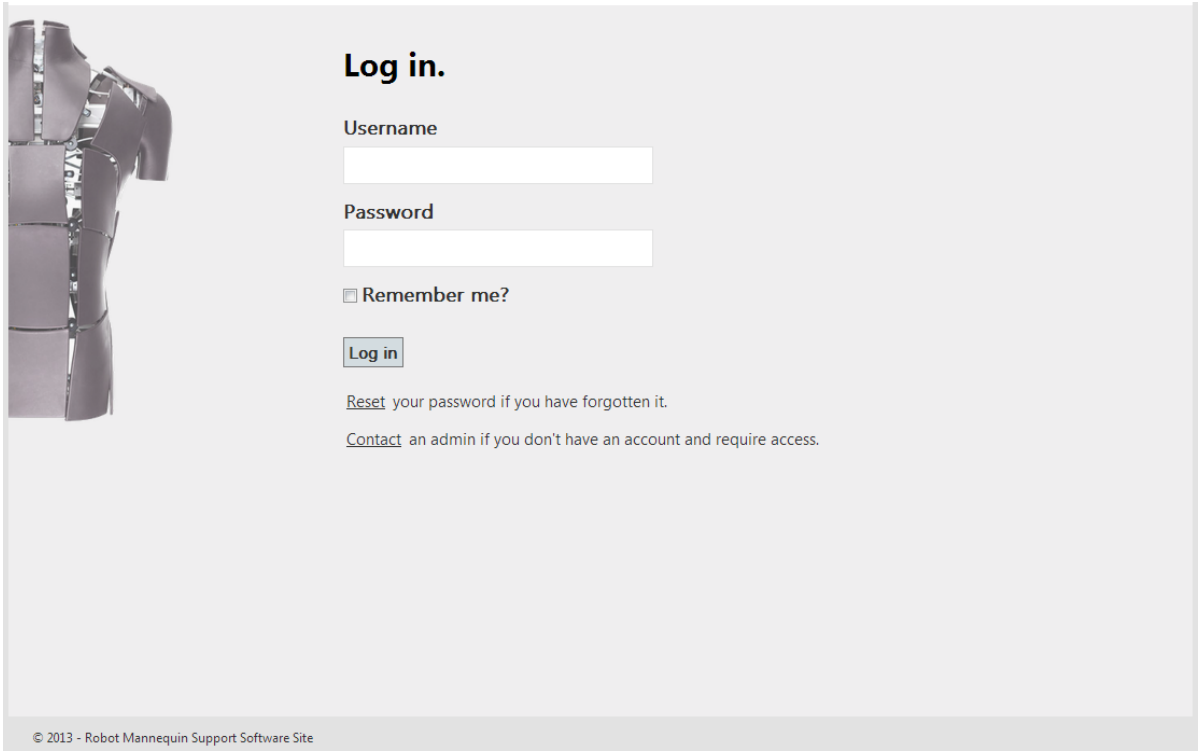


Joonis 14. Kasutusjuhud loodavale rakendusele

3. SENINE TARKVARAARENDUS

Tarkvara arendusega alustati lihtsa sisselogimise liidese programmeerimisest põhjusel, et töö autor ei olnud varem programmeerinud ASP.NET veebirakendusi. Samas pakkus see talle võimaluse tutvuda valitud tarkvaraarenduskeskkonna, programmeerimise mustri ja loodava lahenduse keerukusega.

Nii andmebaas kui ka kasutajaliidese struktuur on loodud vastavalt lõputöös püstitatud nõuetele. Samuti on valmis programmeeritud kasutajate sisselogimise leht (Joonis 15) veebirakendusse ning paigutatud on ka viited erinevatele vaadetele. On realiseeritud kasutajagruppidele põhinev andmete kuvamine.



Joonis 15. Tehnilise toe tarkvara sisse logimis liides

Andmebaasi on sisestatud kasutajad ning nõutud kasutajagrupid. Samuti on loodud seosed kasutajate ja kasutajagruppide vahel. Andmebaas ja veebirakendus on hetkel paigaldatud lokaalsesse serverisse, et võimaldada rakenduse juba implementeeritud funktsionaalsuse testimist.

Seni on probleeme valmistanud esialgse veebirakenduse juurutamine serverisse. Arenduseks kasutatavas arvutis testimisel rakendus töötab, kuid serverisse paigutatuna ilmnesid rakendusel mõned probleemid. Selgus, et serverist olid puudu mõned teegid [28] (ingl k library), mida programm tööks vajab. Teadmata põhjusel ei lisanud VS rakenduse juurutamisel kaasa vajalikke teekisid. Probleemi lahendas serverisse VS arenduskeskkonna paigaldamine.

Tähtsamateks tulemusteks on hetkel nõuete spetsifitseerimine ja kujunduse paikapanemine. Sellega on ka antud töö eesmärk täidetud ning edasi on vaja tegeleda antud veebirakenduse arendamisega vastavalt töös spetsifitseeritud nõuetele.

4. ARENDUSVÕIMALUSED

Antud veebirakendust oleks võimalik edasi arendada selles suunas, et luua sellest robotmannekeeni arendus- ja hooldustöö keskne üksus. Seda oleks võimalik saavutada lisades rakendusele funktsionaalsus, mis on hetkel eraldiseisev: *buglist*, *tasklist* ja wiki. Sellisel juhul toimuks kogu veajälgimine ja tehtud töö dokumenteerimine ühest keskkonnast.

Selle rakenduse saaks veel kesksemaks muuta, kui integreerida MM ja ID programmeerimise liides. Nii oleks võimalik projektijuhil hallata kogu tööd ühest kohast. Samuti hõlbustaks see kogu arendusmeeskonna tööd ning kaasaks stuudio töölisi arendusprotsessi, küsides pidevalt tagasisidet ja ideid mannekeenide arendamise kohta.

KOKKUVÕTE

Käesoleva bakalaureusetöö eesmärk oli koostada nõuete spetsifikatsioon ja ülesehitus tarkvarale, mis aitaks koguda statistilist informatsiooni mannekeenide töökindluse ja tehnilise toe tööefektiivsuse kohta ning samas aitaks parandada tehnilise toe tööefektiivsust ja vähendaks mannekeenide hooldusele kuluvat aega.

Töö käigus koostati üldine nõutav tööprotsessi skeem, et uurida olemasolevaid veajälgimissüsteeme ning võimalust neid rakendada eesmärgi lahendamiseks. Jõuti järeldusele, et olemasolevaid veajälgimissüsteeme ei ole kasulik probleemi lahendamiseks kasutada ning otsustati ise tarkvara luua. Koostati nõuete spetsifikatsioon veebirakendusele, kasutajaliidesele, andmebaasile ja suhtlusele andmebaasiga.

Sellega saavutati lõputöö eesmärk, millele järgnevalt alustati tarkvaraarendusega. Arenduse käigus on autor omandanud esmased teadmised ASP.NET veebirakenduste programmeerimise kohta ning on õppinud kasutama tarkvara arenduskeskkonda Visual Studio. Tarkvara arenduse käigus on esinenud probleeme, millele on leitud lahendused.

Tulevikus oleks võimalik antud rakenduse arendamist jätkates luua sellest robotmannekeeni arendus- ja hooldustöö keskne üksus.

SUMMARY

Requirements Specifications for the Robot Mannequin Photo Studio Technical Support Software

Bachelor Thesis

Tõnis Terasmaa

Summary

The aim of this thesis was to compose the requirements specifications for the studio support software. This software is to be used for gathering statistical information about the reliability of the robot mannequins and the work efficiency of the studio technical support personnel.

The thesis consisted of three parts. The first part of the thesis explains how the information about the reliability of robot mannequins is gathered preceding the release of this software. The first part also explains how the maintenance of the robot mannequins takes place and what other bug tracking software is out there. The second part of the thesis gives an overview of the requirements specifications that were set for the web application and the database that is to be used for storing the information. The third part of the thesis gives an overview of what development steps have already been taken to create the web application.

As a result of this thesis the requirements specifications for the studio technical support software were composed and the log in section of the web application and database have already been programmed and set up for testing.

ALLIKALOEND

[1] „About Fits.me,“ Fits.Me, [Võrgumaterjal]. Available: http://fits.me/about/about-fits-me/. [Kasutatud 19 mai 2013].

[2] „Tartu Ülikooli Tehnoloogiainstituut,“ Tartu Ülikool, [Võrgumaterjal]. Available: http://www.tuit.ut.ee/. [Kasutatud 19 mai 2013].

[3] „Tartu Ülikool,“ [Võrgumaterjal]. Available: https://www.ut.ee/et/ettevotlus/fitsme-robotmannekeen-veebikaubanduse-muugimees. [Kasutatud 19 mai 2013].

[4] „Virtual Fitting Room,“ Fits.Me, [Võrgumaterjal]. Available: http://fits.me/products/the-fits-me-virtual-fitting-room/. [Kasutatud 19 mai 2013].

[5] A.B. Kolluri, K. Tameezuddin, K. Gudikandula, „Effective Bug Tracking Systems: Theories and Implementation,“ *IOSR Journal of Computer Engineering*, kd. 4, nr 6, pp. 31-36, sept.-okt. 2012.

[6] M. Luik, *Robotmannekeeni käsitsijuhtimise tarkvara - Mannequin Manager*, Tartu: Tartu Ülikool, 2012.

[7] „Wiki,“ Wikipedia Foundation, Inc, 9 mai 2013. [Võrgumaterjal]. Available: http://en.wikipedia.org/wiki/Wiki. [Kasutatud 19 mai 2013].

[8] „GNU RCS,“ 6 mai 2013. [Võrgumaterjal]. Available: http://www.gnu.org/software/rcs/. [Kasutatud 19 mai 2013].

[9] „Pearson Higher Education,“ 23 jaanuar 2003. [Võrgumaterjal]. Available: http://www.pearsonhighered.com/samplechapter/020178792X.pdf. [Kasutatud 19 mai 2013].

[10] „Issue and Project Tracking Software | Atlassian JIRA,“ Atlassian, Inc, [Võrgumaterjal]. Available: http://www.atlassian.com/software/jira/overview?_mid=74118c8a9a9e4e4cf8020689c62dacd6&gclid=CJzM68LGorcCFU2N3godU1AAMQ. [Kasutatud 19 mai 2013].

[11] „TrackStudio,“ TrackStudio Ltd., [Võrgumaterjal]. Available: http://www.trackstudio.com/. [Kasutatud 19 mai 2013].

[12] „Web application,“ Wikipedia Foundation, Inc, 12 mai 2013. [Võrgumaterjal]. Available: http://en.wikipedia.org/wiki/Web_application. [Kasutatud 19 mai 2013].

[13] „Integrated development environment,“ Wikipedia Foundation, Inc, 21 aprill 2013. [Võrgumaterjal]. Available: http://en.wikipedia.org/wiki/Integrated_development_environment. [Kasutatud 19 mai 2013].

[14] „About the Eclipse Foundation,“ The Eclipse Foundation, 2013. [Võrgumaterjal]. Available: http://www.eclipse.org/org/. [Kasutatud 19 mai 2013].

[15] „IntelliJ IDEA,“ JetBrains, [Võrgumaterjal]. Available: http://www.jetbrains.com/idea/. [Kasutatud 19 mai 2013].

[16] N.M.A. Munassar, A. Govardhan, „A Comparison Between Five Models Of Software Engineering,“ *International Journal of Computer Science Issues,* kd. 7, nr 5, pp. 94-101, september 2010.

[17] „ASP.NET,“ Microsoft, 2013. [Võrgumaterjal]. Available: http://www.asp.net/. [Kasutatud 19 mai 2013].

[18] „About | Microsoft Eesti,“ Microsoft, 2013. [Võrgumaterjal]. Available: http://www.microsoft.com/et-ee/about/default.aspx. [Kasutatud 19 mai 2013].

[19] „About Linux Foundation,“ The Linux Foundation, [Võrgumaterjal]. Available: http://www.linuxfoundation.org/about. [Kasutatud 19 mai 2013].

[20] „ASP.NET MVC Overview,“ ASP.NET, 27 jaanuar 2009. [Võrgumaterjal]. Available: http://www.asp.net/mvc/tutorials/older-versions/overview/asp-net-mvc-overview. [Kasutatud 19 mai 2013].

[21] „Architectural pattern,“ Wikipedia Foundation, Inc, 12 märts 2013. [Võrgumaterjal]. Available: http://en.wikipedia.org/wiki/Architectural_pattern. [Kasutatud 19 mai 2013].

[22] „Visual Studio,“ Microsoft, [Võrgumaterjal]. Available: http://www.microsoft.com/visualstudio/eng/whats-new#story-whats-new. [Kasutatud 19 mai 2013].

[23] „Template method,“ Wikipedia Foundation, Inc, 3 mai 2013. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Template_method_pattern. [Kasutatud 19 mai 2013].

[24] „Enterprise Architect,“ Sparx Systems Pty Ltd., [Võrgumaterjal]. Available: http://www.sparxsystems.com/. [Kasutatud 19 mai 2013].

[25] „SQL,“ Britannica Academic Edition, 2013. [Võrgumaterjal]. Available: http://www.britannica.com/EBchecked/topic/569684/SQL. [Kasutatud 19 mai 2013].

[26] „SQL Server,“ Microsoft, 16 mai 2013. [Võrgumaterjal]. Available: http://www.microsoft.com/en-us/sqlserver/product-info.aspx. [Kasutatud 19 mai 2013].

[27] „Application programming interface,“ Wikipedia Foundation, Inc, 19 mai 2013. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Api. [Kasutatud 19 mai 2013].

[28] „Library (computing),“ Wikipedia Foundation, Inc, 19 mai 2013. [Võrgumaterjal]. Available: http://en.wikipedia.org/wiki/Library_(computing). [Kasutatud 19 mai 2013].

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina _____TÕNIS TERASMAA_____
(autori nimi)
(sünnikuupäev: _____31.08.1988_____)

annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
NÕUDED ROBOTMANNEKEENIDE FOTOSTUUDIO TEHNILISE TOE TARKVARALE
_____
_____
,
(lõputöö pealkiri)

mille juhendaja on _____ALEKSANDER TÕNNISSON_____,
(juhendaja nimi)

reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.

kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **22.05.2013**