Tallinna Ülikool

Matemaatika-loodusteaduskond

Informaatika osakond

Märt Ringmäe

PROGRAMM: AJAPLANEERIJA MOBIILTELEFONIS

Bakalaureusetöö

Juhendaja: Jaagup Kippar

Autor: ....."....."..... 2005

Juhendaja: ....."....."..... 2005

Osakonna juhataja: ....."....."..... 2005

Tallinn 2005

SISUKORD

SISSEJUHATUS

Elektroonika areng toimub meeletu kiirusega. "*Moore'i* seadus" ütleb, et arvutite võimsus kahekordistub iga pooleteise aasta järgi. Sama printsiip kandub ka teistele elektroonilistele seadmetele, seda nagu näiteks meie igaühe taskus olevatele mobiiltelefonidele. Aasta tagasi võis veel õhinaga rääkida, et turule on jõudnud uued ja uhked telefonid, mis võimaldavad kasutada *Java* programme. Tänaseks on sellest saanud tava. Raske on leida poest mõnda, millel ei oleks *Java* toetust. Kätte on jõudnud aeg, kus igal ühel on vanasõna meelde tuletades "tark mees taskus". Sellest tarkusest ei ole palju kasu, kui see ei haaku juba olemasolevate arvutite ja võrkudega. Bakalaureusetöö peamine eesmärk on vaadelda rakendust, mis ühendaks arvuti, Interneti ja mobiiltelefoni. Töö praktilises osas valmis mobiiltelefonile mõeldud kalender-märkmik ning veebisüsteem, mille vahel on võimalik andmeid sünkroniseerida ja tänu sellele keskendub töö peamiselt kalender-märkmiku loomiseks kulunud teadmiste vaagimisele. Töö baseerub juba varem valminud samanimelisel seminaritööl ning püüab süvendada selle teoreetilist külge ning siluda eelnevaid puudujääke. Valminud veebiportaal asub aadressil: *http://www.m2rt.pri.ee/mkalender*. Samalt aadressilt on võimalik laadida ka mobiiliprogrammi. Rakenduse peamine mõte on toonitada, et kõige tähtsam on info, mitte vahend või viis selle saamiseks. Tööga tutvunud inimene saab ettekujutuse, kuidas alustada saamlaadse süsteemi loomist. Järgnevatel lehekülgedel uuritakse arenduseks kasutatud vahendeid ja pakutakse välja üks võimalus, kuidas realiseerida mobiiltelefonidele sobiv ajaplaneerija.

1. LOODUD SÜSTEEMI TUTVUSTUS

1.1 Ajaplaneerija definitsioon

Ajaplaneerija on süsteem, mis hoiab endas andmeid nii toimunud kui ka toimuvate sündmuste ning tegevuste kohta, aidates seeläbi kasutajal oma aega paremini ülevaatlikuks muuta. Ajaplaneerija peamiseks ülesandeks on sündmuste algusaja ning kestvuse meelespidamine. Digitaalse planeerija põhiülesannete hulka kuulub ka andmete otsimine ja nende sobival viisil kasutajale kuvamine.

1.2 Loodud ajaplaneerija eesmärk

Valminud ajaplaneerimissüsteemi peamine eesmärk on ühendada omavahel kasutaja taskus olev mobiiltelefon, veebiportaal ja lauaarvuti, et kaotada ära olukord, kus andmed on ühes neist kolmest seadmest ja ei ole kättesaadav teisele kahele. Kõige tähtsamal kohal on info ja seda peab saama liigutada endale meeldivasse seadmesse ja endale meeldivale kujule. Lauaarvutitele on mõeldud palju kalendrisüsteeme, mis on väga kasutusmugavad ja paindlikud ning selles valdkonnas midagi märkimisväärset ära teha on küllalt raske. Seetõttu on kasutajale antud võimalus valida endale sobiv kalendrisüsteem ning bakalaureusetööna valmib mobiili- ja veebirakendus, mis ühildub lauaarvuti standardse kalendrisüsteemiga. Suurim rõhk loodud rakenduses on pandud sellele, et mobiili ja veebiportaali vahel toimuks andmevahetus, kuid mobiil ei laeks kõiki oma andmeid reaalajas portaalist, sest sellisel juhul võiks leviauku sattunud kasutaja jääda hätta. Kirjeldatud lähenemine hoiab kokku ka kallist andmevahetuse aega ja kulu. Ühilduvuse ja mugavuse raames laetakse andmed veebist *XML*-vormingus ja seejärel salvestatakse telefoni. Läbi telefoni saab andmeid vaadata, muuta ja otseloomulikult ka lisada ja otsida ning seejärel veebi tagasi laadida. Veebis tehtavad toimingud on sarnased mobiilivõimalustega, kuid neile lisandub veel üks valdkond – veebiportaal võimaldab andmeid tõlkida erinevatesse vormingutesse, mida saab lauaarvutisse laadida ja juba endale meeldiva programmi abil avada ning redigeerida.

Kalendrisüsteem katab valdava osa tavanõuetest:

- ühe- kui ka mitmekordsete sündmuste hoidmine;
- erinevate kordumismustrite märkimine;
- sündmuste aegumine;
- kirjete tähtsuste määramine;
- päeva-, nädala- ja kuuplaani kuvamine;
- enda poolt valitud parameetri järgi otsing.

Mobiilisüsteemis lisandub neile veel andmete sünkroniseerimise võimalus.

1.3 Arenduseks kasutatud vahendid ja standardid

Mobiilisüsteem on suunitletud *Java* toega telefonidele ning seetõttu kasutab *Java 2 Micro Edition* platvormi. Veebiportaal on üles ehitatud laialt levinud *PHP*'le ning andmebaasimootorina kasutatakse *MySQL*'i. Andmevahetuses telefoni ja portaali vahel kasutatakse *XML*'i ning portaalist lauaarvutisse võib andmeid laadida *iCalendar*, *xCal* ning *mCal* vormingus. Tehtud tööst ja mainitud standarditest täpsemalt järgnevatel lehekülgedel.

2. ÜLEVAADE JAVAST

2.1 Lühidalt Java ajaloost

1990'ndate alguses kui PC-tüüpi endistest odavamad arvutid olid tegemas revolutsiooni ning võrkudesse ühendatud arvutite tähtsus aina kasvas, tekkis vajadus programmeerimiskeele järgi, mis oleks teadlik uutest võrgu võimalustest, paindlik, kuid samas mitte nii detailidesse laskuv ja keeruline nagu seda oli senine vaieldamatu liider programmeerimiskeel *C*. Juba varastel 80'ndatel ideega "Võrk on arvuti." ("*The network is the computer.*") turule tulnud *Sun Microsystems* haaras härjal sarvist ja alustas seesuguse uue keele loomisega [1]. Peamine põhjus, miks *Sun* sellise suure ülesande endale võttis, oli küllalt lihtne – nende tollane plaan nägi ette "*targa koduelektroonika*" loomist, mille jaoks oli vaja senisest universaalsemat ja turvalisemat aluspõhja – süsteemi käsustikku ja seda kasutavat programmeerimisvahendit. Üheks nende tollaseks ideeks oli näiteks interaktiivse televisiooni loomine. Paraku ei saavutanud *Sun Microsystems'*i televisiooni projekt väga suurt edu, kuid töö käigus väljatöötatud tarkvara pakkus juba nii mõnelegi huvi. Umbes samal ajal algas Interneti võidukäik, mis tõmbas *Sun*i tähelepanu endale. Pooleli oleva arendusvahendi, senise nimega *Oak*, sihtgruppi laiendati veelgi. Umbes samal ajal nimetati *Oak* ümber *Javaks*. Uus keel, *Java*, oli fokusseeritud ei vähemale ega rohkemale kui kõikidele programmeeritavatele elektroonikaseadmetele. [2]

2.2 Java aluspõhi

Java kasutusvaldkond on seatud äärmiselt laiaks. Nagu juba mainitud, võib seda kasutada nii raadio, pihuarvuti kui ka võimas server. Seesuguse ühilduvuse tagamiseks ei saa olla kindel ei protsessori käsustikus ega ka milleski muus süsteemispetsiifilises teguris. *Sun Microsystems* tõstis arendajate poolt loodava rakendusekihi tavapärasest kõrgemale lisades juurde tarkvara, mis etendab rakendustele riistvaralist süsteemi, see tähendab virtuaalmasina. Kogu funktsionaalsuse kasutamine ja ressursside hõivamine ning vabastamine käib läbi virtuaalmasina. Seega on olemas koht, mis suudab tõlkida rakenduselt tulevaid soove kasutusel olevale riistvaralisele platvormile arusaadavaks ning vajadusel on olemas ka

võimalus kontrollida ja piirata töötava rakenduse funktsionaalsust. Iseenesest ei olnud seesugune lahendus midagi uut. Sarnaselt töötavad ju kõik interpreteeritavad keeled – skriptist loetud käsu peale käivitatakse nõutud operatsioon! *Java* uudsus seisnes selles, et kood tuleb esmalt kompileerida baitkoodi, mis iseenesest on kui universaalne masinkood. Baitkoodi loomisel peeti silmas, et seda oleks võimalikult lihtne realiseerida nii tark- kui ka riistvaras ja et kood sisaldaks palju metainformatsiooni, mille järgi saaks virtuaalmasin otsustada käskude turvalisuse üle. Seega võib *Java* baitkoodi käitlev virtuaalmasin olla ka tegelik riistvarasüsteem, mis võtab käske kui omaenda masinkoodi. Siinkohal ähmastub küsimus, et kas *Java* on kompileeritav või interpreteeritav keel. Mingil hetkel võivad mõlemad vastused olla õige. Erinevalt mitmetest varasematest interpreteeritavatest keeltest on *Javal* olemas võimalus missioonikriitilised kohad baitkoodis rakenduse käitlemise ajal masinkoodi tõlkida ("*just in time*" *compilation*). See võib tõsta oluliselt rakenduse kiirust. Sarnane optimeerimine on ka näiteks *Perlis* ja *GNU CLISP'*is. [3]

2.3 Turvalisus

Turvalisus on olnud läbi aegade üheks *Java* peamiseks argumendiks. Rääkides programmeerimiskeeltest jõutakse sageli kolme olulise tegurini – kiirus, turvalisus ja portatiivsus. Paljudes juba ennast sissetöötanud keeltes nagu näiteks *C*, *C++*, *Pascal* ja *Delphi* on peamine rõhk pandud just nimelt kiirusele. Programmeerija kirjutatud kood püütakse võimalikult optimaalseks masinkoodiks teha. Kuidas loodud tarkvara hiljem mõnele teisele süsteemile edasi tõsta, see on juba suuresti keele kasutaja enda probleem. Enamasti püütakse see lahendada universaalsete teekide ja erinevatele platvormidele sobilikuks kompileerimise teel, mis iseenesest tähendab rohkem tööd ja sellest hoolimata ebatäielikku ühilduvust. Sarnased puudujäägid varitsevad ka turvalisuse vallas. Igasugune kontroll tähendab lisatööd, mis omakorda rohkem kulutatud aega. Seega ei saa eelnevalt nimetatud keeled endale lubada reaalajas andmete kontrollimist. Kõik see on jäetud programmeerija õlule. Olemas on küll esmane kompileerimisel kasutatav tüübiteisenduste ja muu elementaarse süntaksi ja semantika analüüs, aga see on ka peaaegu kõik. Tänapäeva võimaste ja missioonikriitiliste arvutisüsteemide juures ei ole aga tarkvara ülisuur kiirus enam peamine – tähtis on siiski optimaalsus. Puhvrite ületäitumine, viitade väärtustamata jätmine,

funktsioonide kogemata või kuritahtlikult väärkasutamine ja muu selline on põhjustanud arvutimaailmas tohutul hulgal peavalu. Selle vältimiseks on tihti pidanud ettevõtted arendama omaenda andmete õigsust ja korrektsust tagavad vahekihid, mis jällegi tähendab lisatööd. Kõik see toob meid tagasi *Java* juurde, sest siin on need probleemid suures osas tänu virtuaalmasinale ja keele objektorienteeritusele juba eos lahendatud. Enne koodi käivitamist kontrollitakse see alati spetsiaalse tarkvaraosa ("*Verifier*") poolt üle. Kuna baitkood sisaldab küllalt palju metainformatsiooni, siis saab virtuaalmasin väga täpselt koodi üle otsustada. Reeglid, mille alusel otsustamine toimub, on otseloomulikult paindlikud ja vajadusel seadistatavad. Virtuaalmasin ei ole küll võimeline programmeerija kehva tööd heaks tegema, aga vähemalt leevendab see vea korral tulevaid kahjusid.

2.4 Süntaks

Süntaksilt on *Java* küllalt sarnane *C++*'iga. Mõlemad keeled püüavad lahendada probleeme võimalikult väheste sõnade ja reeglitega. Nii valiklaused, tsüklid, funktsioonid kui ka osaliselt objektid on mõlemas keeles ülesmärgitavad ühte moodi (vt. koodinäide 1 *Java* klass). Välisest sarnasusest hoolimata ei tasu tähelepanuta jätta mitut olulist seika:

- *Javas* ei ole preprotsessorit,
- viidad (*pointers*) on asendatud turvalisemate viidetega (*references*),
- puudub mallide (*templates*) süsteem,
- olemas on automaatne mäluvabastus (*automatic garbage collection*)
- objektorienteeritus on ülesehitatud erinevatel alustel.

Java objektid saavad olla päritud vaid ühest baasklassist, *C++*'is võib mitmest. Kriitikud on väitnud, et tegelikult ei ole *Java* puhtalt objektorienteeritud keel, sest selles leiduvad põhilised andmetüübid nagu näiteks *int*, *float*, *double* ja teised ei ole objektid [3]. Osalt on see väide õige, aga tegelikult leiduvad kõikidele põhilistele andmetüüpidele on ka vastavad klassid, näiteks *Integer*, *Float*, *Double*, mida küll väga sageli ei kasutata. Süntaksilt ja ülesehituselt on *Java* kahtlemata tänapäevane keel. Kui vaadata teisi hetkel laialt levinud keeli, siis oma kasutusmugavuselt suudab *Javale* konkurentsi pakkuda vaid *C#*, mis on üks uuemaid populaarsemaid programmeerimiskeeli ja seetõttu on selles kajastatud enamus

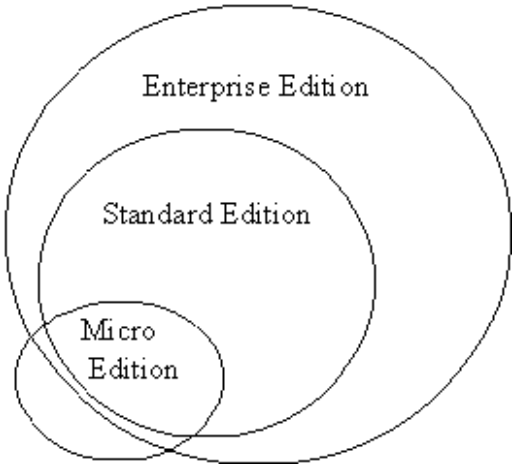
obkjektorienteeritud keele tänapäevaseid võimalusi. Paljud neist on võetud just *Javalt*. Seetõttu on ka süntaks *C#*'l äravahetamiseni sarnane *Javaga*.

```
public class Person {
	String firstName;
	String lastName;

	public void printName(boolean onlyFirstName) {
		String name;
		if(onlyFirstName)
			name = firstName;
		else
			name = firstName + " " + lastName;

		System.out.println("Name : " + name);
	}
}
```

Koodinäide 1. *Java* klass

2.5 Platvormide jagunemine

Java programmeerimiskeel nägi ilmavalgust 1995 aasta lõpus. Pärast seda on arendusplatvorme kui ka keelt ennast täiendatud. 1998 aastal toodi *Javasse* mitmeid suuremaid uuendusi ja alates sellest ajast on lisandunud nime lõppu number kaks. *Java 2* platvormid jagunevad võimaluste ja kasutusvaldkondade järgi kolmeks: *Enterprise*, *Standard* ja *Micro Edition* (vt. joonis 1 *Java* platvormide standardklasside näiline jaotus). [3]



Joonis 1. Java platvormide standardklasside näiline jaotus

Standard Edition (J2SE): platvorm, mis on mõeldud töötama tavalisel kodu või tööarvutil, sisaldab peamist funktsionaalsust kasutajaliidese loomise, veebiga suhtlemise jms alal.

Enterprise Edition (J2EE): sisaldab samu klasse, mis *Standard Edition* ning lisaks servlettide, *JSP* ja *XML*'i vahendeid; mõeldud peamiselt serveripoolse funktsionaalsuse loomiseks.

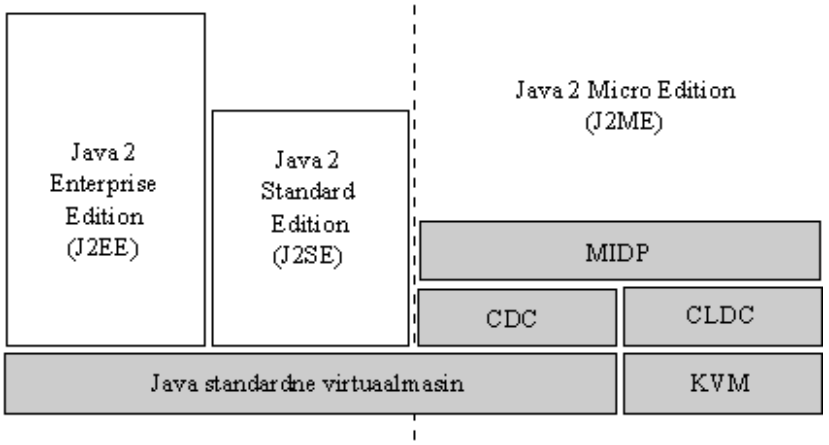
Micro Edition (J2ME): suunatud seadmetele, mille mälu hulk, ekraani suurus kui ka andmetöötluskiirus on piiratud; sisaldab vaid osa *Standard Edition* klassidest ja lisaks neile mõnda mikroseadme võimaluste kasutuseks mõeldud paketti.

2.6 Java 2 Micro Edition

Java 2 Micro Edition (edaspidi *J2ME*) on *Java* platvorm piiratud jõudlusega seadmetele nagu mobiiltelefonid, pihuarvutid ja muud programmeeritavad väikeseadmed. *J2ME* on väga sarnane oma sugulaste *J2SE* (*Java 2 Platform, Standard Edition*) ja *J2EE*'ga (*Java 2 Platform, Enterprise Edition*). Peamine erinevus seisneb esimese oluliselt väikesemas *API*'s. Keel, *Java*, ja seega ka loodava programmi aluspõhi on kõigil mainitud tarkvaraarendusplatvormidel muidugi sama. Sarnased on ka nende standardvarustuses olevad klassid ja meetodid, erinevus on vaid viimaste hulgas. *J2SE* mitmete tuhandete asemel on *J2ME*'s vaid umbes poolsada klassi. Sellest hoolimata on vajadusel võimalik programm või

selle osa ühelt platvormilt küllalt minimaalsete muudatustega ümber tõsta teisele. Eeldusel muidugi, et tegemist ei ole väga spetsiifilise rakendusega. Paraku on olemas ka mõned *J2ME* piirangud, millest täpsemalt pisut hiljem. [4]

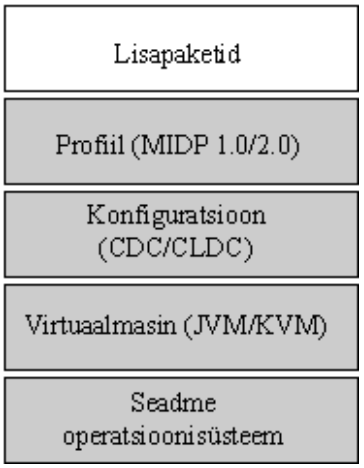
J2ME peamine eesmärk on olla võimalikult seadmesõbralik. Tänu sellele on viimasel paindlik kasutajaliidese loomise süsteem, jäik turvakontroll ja lai sisseehitatud võrguprotokollide hulk, mis annab vajadusel võimaluse luua üksteisega suhtlevaid programme. Oma paljudest headest külgedest hoolimata peab *J2ME* arvestama väga limiteeritud ressurssidega. *J2ME* võib olla kahesuguse aluspõhja-seadistusega, mis kirjeldavad ära süsteemi peamised riistvaralised võimalused: *CLDC* (*Connected Limited Device Configuration*) ja *CDC* (*Connected Device Configuration*). *CLDC* on mõeldud jõudluselt nõrgematele seadmetele, näiteks nagu valdav osa mobiiltelefone. *CLDC* süsteemid omavad aeglast 12-30MHz ja 16 või 32bitist protsessorit. Kasutatav mälu on piiratud 128-512kilobaidiga [5]. *CDC* töötab vähemalt 2megabaidise mälu ja vähemalt 32bitise protsessoriga kõrgema hinnaklassi pihuarvutitel. *CDC* kasutab täies mõõdus *Java* virtuaalmasinat ja suuremat osa *J2SE* klasse, seega on ka rohkem võimalusi. Tänu samale virtuaalmasinale on tarkvara ümbertõstmine *CDC* seadmele lihtsustatud. Töö praktilises osas, kalender-märkmikus, kasutuses olevas *CLDC* seadistusega mobiilis on aga standardne virtuaalmasin vahetatud kokkuhoiu mõttes välja *K virtuaalmasinaga* (kohati nimetatakse seda ka *Kilobait virtuaalmasinaks* või lühidalt *KVM*'ks). Viimane nõuab töötamiseks vaid 80KB mälu, on võimalikult kiire ja kergesti muudetav. Tänu sellele on *KVM* paljudele väikestele süsteemidele sobilik. Võrreldes standardse *Java* virtuaalmasinaga on K virtuaalmasinas tehtud mitmeid muudatusi, osad neist on seotud ressursside kokkuhoiuga, teised jälle, et vältida võimalikke turvaprobleeme. Kuna käesolev töö püüab lahata vaid *CLDC* programmi loomise võimalusi ja piiranguid, siis on jäetud *CDC* seadistusega süsteemid vaatluse alt välja. Vt. joonis 2 *J2ME* seos ülejäänud *Java* platvormidega.



Joonis 2. *J2ME* seos ülejäänud Java platvormidega[6]

Järgmine kiht *J2ME* platvormis kannab nime *MIDP* (*Mobile Information Device Profile*). *MIDP* pakub põhilist funktsionaalsust, mida rakendustes kasutada saab. *MIDP* sisaldab nii kasutajaliidese, võrguühenduse, andmehoidla kui ka rakenduse haldamiseks vajalikke klasse. Nii nagu *CLDC* kirjeldab riistvara, omab *MIDP* infot tarkvara võimalustest [4]. Käesoleval hetkel on väljatöötatud kaks *MIDP* versiooni: *MIDP 1.0* ja *MIDP 2.0*. Suur osa poes müügil ja ringluses olevatest *Java* toega telefonidest kasutab *MIDP 1.0*, mis võrreldes järgmise versiooniga omab vähem võimalusi eelkõige mängude, aga ka tavalist kasutajaliidest loovate klasside näol. Valdav osa asju, mida on võimalik uuema profiiliga teha, on võimalik ka vanemas realiseerida. Selleks tuleb aga ise natuke rohkem vaeva näha ja tulemus ei pruugi ka kõige optimaalsem tulla. Pisut kallimad telefonid realiseerivad juba *MIDP 2.0* standardit. Paraku ei ütle keskmisele telefoniostjale nimetatud standardid mitte midagi. Peamiselt pööratakse vaid tähelepanu, et kas on *Java* "mängude" tugi või mitte. Loodud kalendermärkmik on töötab uuemal tarkvara profiilil.

Viimaseks *J2ME* kihiks on lisatarkvarapaketid, mida saavad tootjad ise kaasa panna. Nende eriliste klasside ülesanne on võimaldada seadmele iseloomulike eriomaduste, nagu näiteks infrapunaliidese, mis ei pruugi kõikide süsteemidega alati kaasas olla, kasutamist. Lisapakettide hulk on piiratud vaid seadme jõudluse ja võimalustega (vt. joonis 3 *J2ME* platvormi osad) [4].



Joonis 3. *J2ME* platvormi osad [6]

2.7 J2ME eripärad ja lähenemisviisid

2.7.1 Abstraktsioon ja Võimaluste avastamine

J2ME programmeerimise kaks kõige olulisemat meelespead on "tundmatud olud" ja "kitsad tingimused". Kuna *J2ME* on mõeldud väga laiale seadmetehulgale, alustades mänguasjadest, lõpetades võimsa pihuarvutiga, on programmeerijal väga raske hinnata keskkonda, kus ta loometöö lõpuks jooksma hakkab. Teada on vaid, et masinal peab olema vähemalt 94x54 ühebitise värvisügavusega ekraan, *QWERTY*-klaviatuur, võimalus minna Internetti ning kasutada saab kõiki *J2ME* standardikohaseid funktsioone (*MIDP* standard) [7]. Samas ei ole kusagil paika pandud ekraani maksimumsuurust ega ka masinal küljes olevate nuppude või kangikeste suurimat arvu. Programm peab samahästi töötama nii värvikireva puuteekraani kui ka kahetoonilise mobiiliekraaniga. Selle probleemi lahendamiseks kasutatakse kahte meetodit: abstraktsioon (*abstraction*) ja võimaluste avastamine (*discovery*). [8]

- **Abstraktsioon**

Valdav osa mobiili kasutajaliidesest koosneb programmeerija jaoks abstraktsetest klassidest, mille poolt kirjeldatavate ekraaniobjektide täpset kuju, asukohta ega suurust kodeerimise

käigus ei ole võimalik teada. Mainitud parameetrid paneb paika süsteem ise, millel rakendus kunagi jooksma hakkab. Tänu sellisele lähenemisele on saavutatud suur osa seadmestsõltumatust. Isegi programmi autor ei saa öelda, kuhu ilmub vajadusel nupp „Tagasi“. Olenevalt seadmest võib see ilmuda kuhu iganes. [8]

Teine osa abstraktsioonist puudutab sisendseadmeid. Kuigi *MIDP*-standard ütleb, et seadmel peab olema vähemalt mobiilile iseloomulik numbrilaud, QWERTY-klaviatuur või puuteekraan ei pane see tegelikult täielikku piirangut sisendseadmetele. Heaks programmeerimise stiiliks nimetatakse nö *soft key* kasutamist. *Soft key* on abstraktne nupp, mis võib, kuid ei pruugi seadmel olla. Kui soovitud nuppu pole, siis tekib see programmeerijast sõltumata, iseenesest ekraanile. Samas, alati võib ühte nuppu füüsiliselt esindada kuitahes palju erinevaid sisendseadmeid. *Soft key* tüübid kannavad küllalt iseloomulikke nimetusi, nagu näiteks: üles, alla, vasakule, paremale. Kuigi ei saa olla täiesti kindel, et nupp „vasakule“ asub ka ise vasakul, võib programmeerija neid südamerahuga kasutada ja olla veendunud, et valminud tarkvara töötab võimalikult laial seadmeteringil. *Soft key*’d on enamasti kasutusel mängudes, tavalise kasutajaliidese loomisel neid sageli vaja ei lähe. [9]

- **Võimaluste avastamine**

Oluliseks seadme parameetriks on selle ekraani suurus, millest lähtuvalt peab programm otsustama, kui palju infot korraga kasutajale kuvada. Väikesel tavatelefonil võib neli rida infot olla liiga palju, pihuarvutil on seda aga selgelt liiga vähe. Kahjuks ei ole siin enam lihtsat võimalust lasta süsteemil oma tarkust näidata ja programmeerija eest otsustada. Praegusel juhul peab kasutama teist võimalust – rakenduse töötamise ajal tuleb ekraani mõõtmed tuvastada ja vastavalt neile otsustada, kui palju andmeid kuvada. Sarnaselt tuleb käituda ka mälu ja andmete salvestuseks mõeldud meediumi kasutamisel. Mikroseadmes on kõik võimalused tavapärasemast kitsamad.

2.7.2 Ekraaniobjektide tekst

Kõikide nuppude, akende, teatekastide või teiste kasutajaliidese osade tekstid peaksid olema võimalikult lühikesed, kuid samas end hästi kirjeldavad. Näiteks ühe esimese massidele suunatud *Java* toega mobiili *Siemens C55* nupp saab kanda maksimaalselt kaheksat tähemärki. Seega ei oleks praktiline luua võimalus „Redigeeri“, kuna viimane ei mahuks lihtsalt nupule ära. Liiga pika sõna sabast võtab süsteem automaatselt üleliigse osa maha, mille tulemuseks võib kõige õnnetumal juhul tekkida isegi eksitav info. Välja nimega "Redigeeri" asemel saab kasutada lühemaid sõnu, näiteks "Muuda" või "Paranda". Veel üheks võimaluseks on kasutada lühendeid, kuid need võivad olla kasutajale esmapilgul veelgi arusaamatumad. Näiteks juba mainitud *C55*'l viib "vast-ta" meid vastamata ja "sisen" vastatud kõnede juurde. Inglise keeles on tehtud küllalt suur töö, et leida lühikesi ja seesugusteks asjadeks sobilikke sõnu. Näiteks pakkus *Siemens* oma arendajatele mõeldud veebilehel spetsiaalset sõnavara, mida saaks kasutada erinevatele nuppudele pealkirjade andmisel [9]. Peale inglisekeelsete väljendite oli lehel veel mõnede suuremate keelte nagu näiteks saksakeele sõnavara. Kahjuks ei ole Eestis mobiilile sobilikku sõnavara teadaolevalt analüüsitud. Üha areneva mobiilimaailma juures ei pruugi seda isegi enam vaja olla. Uute seadmete ekraanid muutuvad aina kvaliteetsemateks ja viimastel mudelitel “kaheksa tähemärgi probleemi” enam ei ole. Sellest hoolimata ei tasuks kohe vanu seadmeid unustada.

Valdaval osal tänapäevastel telefonidel on värviline ekraan, kuid samas ei tasuks tähelepanuta jätta olukorda, kus seadme kuvaril on vaid kaks tooni! Kõik värvid arvutatakse alati ümber süsteemile sobilikuks. Paljud toonid, mis programmeerija ekraanile on pannud, võivad kasutajale paista äravahetamiseni sarnased. Isegi värvilise ekraani puhul ei ole mingisugust garantiid, et kõik toonid säilivad algupärastena. Kui soovida, et info igas olukorras välja paistaks, on vaja kasutada küllalt suuri kontraste [9].

2.7.3 Vähesed ressursid

Mobiilisüsteemi ressursid on võrreldes tavapärase lauaarvuti omadega kaduvväikesed. Seega tuleb neid kasutada harjumuspärasest hoolikamalt. Ajaplaneerijas on kõige mahukamaks

tööks mitmed stringioperatsioonid. Eelkõige nõuab protsessorijõudlust *XML*'i lugemine ja tekitamine, mis koosneb paljudest tekstiga seotud tehetest. Kui võimalik, siis tuleks seesuguseid käsklusi piirata miinimumini. Soovitav on isegi tekstide liitmise asemel kasutada eelnevalt defineeritud konstante, kus tehe on juba tehtud. Antud juhul oli aga selline lähenemisviis võimatu. Teiseks suuremaks jõudlust vajavaks kohaks on kalendrist korduvate sündmuste otsimine. Ka siin puudub hea lahendus, sest kõik kirjed tuleb igal juhul läbi käia.

2.7.4 Võimalikud katkestused

Rakenduse tööd võib igal ajal katkestada sissetulev kõne, *SMS* või tühjenev aku. Telefonilt tulevatele teadetele on alati antud kõrgeim prioriteet ja need võivad peatada programmi töö ükskõik mis ajal või mis olukorras see ka ei oleks. Kui kõnet vastu ei võeta, siis võib programm oma tööd jätkata. Kõne vastuvõtmisel lõpetatakse automaatselt ka avatud *Java* rakenduse töö. *J2ME* programmile on antud võimalus andmeid enne kirjeldatud olukorda salvestada. Vajadusel tuleks seda kindlasti kasutada. [9]

2.8 J2ME tüüpilise mobiilirakenduse struktuur

J2ME programm, mis on ülesehitatud vastavalt *MIDP* standardile peab omama vähemalt ühte klassi, mis on päritud *MIDlet* klassist. *MIDlet* on oma olemuselt küllalt sarnane veebiprogrammeerimiseks kasutatava *appletiga*. Mõlematest alustatakse laadimist ja mõlemal on teatud hulk olekuid. Eripäraks on see, et *MIDlet* klassi olekud ei ole niivõrd seotud ekraanil pildi kuvamisega kui *appleti* omad. *MIDlet* võib olla neljas olekus: laetud (*loaded*), aktiivne (*active*), peatatud (*pause*) või lõpetatud (*destroyed*). Kui rakendus on seadme mällu viidud ja peaklassi konstruktor käivitatud, pannakse *MIDleti* olekusse laetud. Peale seda kutsub süsteem välja meetodi *startApp*, mis viib programmi olekusse aktiivne. Klass jääb sellesse olekusse kuni rakenduse töö peatatakse meetodiga *pauseApp* või hoopis lõpetatakse kutsudes välja *destroyApp*. Meetodit *pauseApp* rakendatakse siis, kui telefonil on vaja kasutajale midagi teatada. Näiteks sissetulev kõne, tühjenev aku või muu selline viib automaatselt avatud rakenduse peatatud olekusse. Rakenduse lõpetamiseks kasutatav meetod

destroyApp omab parameetriks tõeväärtust *true* või *false*. Kui tõeväärtuseks on seatud *false*, siis võib rakendus sulgemisest keelduda, saates välja erindi *MIDletStateChangeException*. *MIDlet* võib ise oma olekuid muuta kasutades *notifyPaused* ja *notifyDestoyed* meetodeid. Standardse *MIDleti* ülesehitust kirjeldab koodinäide 2.

```
import javax.microedition.midlet.MIDlet;

public class SampleMIDlet extends MIDlet{
      public SampleMIDlet(){
      }

      protected void startApp() throws MIDletStateChangeException{
            Display.getDisplay(this).setCurrent(some_displayable);
      }

      protected void pauseApp(){
      }

      protected void destroyApp(boolean unconditional)
                  throws MIDletStateChangeException{
                if (!unconditional)
                {
                      // continue
                      throw new MIDletStateChangeException();
                }
      }
}
```

Koodinäide 2. *MIDlet* programmi ülesehitus

MIDlet võib töötada taustaks ilma kasutajale midagi kuvamata või võib sisaldada kasutajaliidest. Kasutajaliideseks sobivad kõik klassid, mis on päritud abstraktsest baasklassist *displayable*. *Displayable* klassist tulenevad kaks peamist kasutajaliidese haru: *screen* ja *canvas. Screenist* on tuletatud erinevad nupud, teatekastid ja muud kõrgema taseme

ekraaniobjektid. *Canvas* pakub aga madalama tasandilist juurdepääsu ekraanile ning seda kasutatakse peamiselt mängude loomisel. [10]

2.9 J2ME kasutajaliidese mustrid

Väheste ressurssidega mobiiltelefoni kasutajaliidese loomiseks on välja pakutud mitmeid disainimustreid, millest loodud rakendus on püüdnud enamasti kinni pidada. Kuna telefoni ekraan on väike, siis peamiseks disaini küsimuseks ongi probleem, et kuidas vähendada korraga kuvatava info mahtu, kuid samas võimaldada pakutavale lihtsat juurdepääsu. Veebilehel *javaworld.com* saadavas artiklis "*Big design for small devices*" [11] on tutvustatud nelja järgmist lahendust:

- menüüde ja alammenüüde süsteem

  Kui valikute menüü venib nii pikaks, et tekib kartus, et kasutajal on raske saada head ülevaadet (näiteks kui menüü on rohkem kui kolm lehekülge pikk), siis on mõtteks menüüd jagada lühikesteks alammenüüdeks. Vajadusel võib viimaseid veelkord osadeks jagada. Loodud ajaplaneerijas on menüüd enamasti kahetasandilised, et olla ülevaatlikum.

- võlur (*wizard*)

  Kui andmete vorm on mahukas, siis võiks seda esitada mitmel leheküljel, mis on kokku koondatud nagu lauaarvutist tuttavad "võlurid". Kasutajal on mugavam vajutada nuppe "edasi" ja "tagasi", kui kerida pikka küsitlusahelat ülesse ja alla. Võluri abil on hea anda ülevaade vormi täitmise protsessist. Näiteks võib akna pealkirjaks olla tekst, mis näitab mitmes küsitlusleht on ja kui palju on lehekülgi kokku.

- otsingutulemuste jaotamine lehtedeks

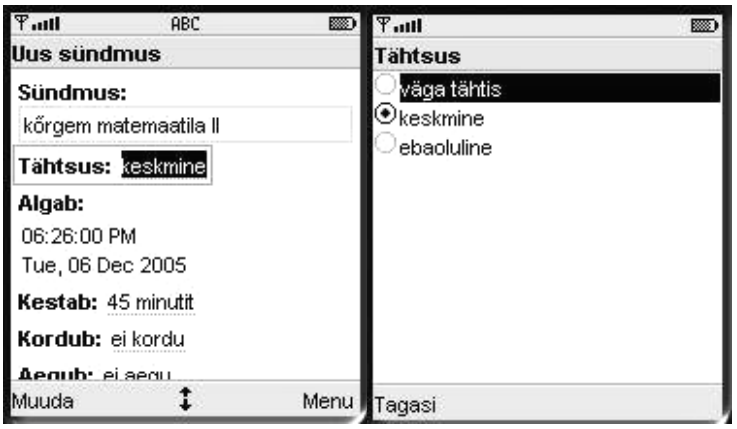
  Mahukate otsingute tulemused on otstarbekas mitmele lehele jaotada. Korraga mitmekümne rea kuvamine on esiteks ressursinõudlik ja teiseks ei ole see kasutajale üldsegi mugav. Samas tuleks vältida liigset tükeldamist.

- automaatne esitlus

Viimaseks väljapakutud lahenduseks on lehekülgede automaatne esitlus, see tähendab kindlaks määratud järjekorras vahetuvaid ekraanivorme. Automaatika vabastab kasutaja tülikast nuppude vajutamisest, kuid samas ei lase tal kontrollida ekraanil kuvatavat. See on ühtlasi nii hea kui ka halb. Automaatset esitlust saab kasutada ainult teatud olukordades, kus protsessi jätkamiseks ei ole oluliselt eriskummalisi võimalusi. Näiteks võib seda disaini kasutada pildialbumi kuvamises. Küll aga ei sobi see kalendrikirjete vaatamiseks.

Loodud ajaplaneerijas on kasutatud veel mõnda viisi kuidas pakkuda head ülevaadet andmetest. Uue kalendrikirje sisestamisel on mitmed valikud nagu näiteks kestvus, kordumine, aegumine, tähtsus ja mõned muud esitatud tekstilise nupuna. Kui valida tekst, siis avaneb aken, kus on võimalik määrata uus väärtus valitud parameetrile (vt joonis 4 loodud ajaplaneerijas kirje tähtsuse valimine). Tänu sellele ei pruugi kasutaja uue kirje sisestamisel kõiki võimalikke parameetreid vaadata ega seadistada. Kui võrrelda ajaplaneerijas realiseeritud andmete küsimist ja esitamist artiklis *"Big design for small devices*" välja pakutud sobilikuma variandi – võluriga, siis on loodud rakenduse realisatsioon mugavam. Ühekordse sündmuse sisestamiseks oleks võluris pidanud kasutaja läbi käima kõik kirje seadistused või oleks pidanud andmed poole võluri täitmise pealt salvestama ja lootma, et ülejäänud osas esitatavad küsimused omasid “õigeid” vaikimisi väärtusi. Ajaplaneerijas näeb kasutaja vaikimisi seadistust lühikese kokkuvõttena ekraanil ja seetõttu ei pea ta midagi eeldama vaid võib olla kindel, et sisestatud kirje on õige.

Mugavuse tõstmiseks on ajaplaneerija menüüd järjestatud kasutussageduste järgi. See tähendab, et kõige kõrgemal positsioonil on valik, mida kasutatakse kõige rohkem. See vähendab nuppude vajutamist ja kiirendab programmi kasutamist. Ka uue kirje sisestamisel küsitavad väljad on proovitud panna nii järjekorda, et esimesed väärtused oleksid alati vajalikud ja viimastest võib aegajalt üldsegi loobuda (näiteks iga kirje ei pruugi korduda, seega ei ole vaja kordumist ega ka aegumist seadistada).



Joonis 4 Loodud ajaplaneerijas kirje tähtsuse valimine

3. LEVINUD KALENDRITE FORMAAT E

3.1 iCalendar

Valdaval osal kalendrisüsteemidel on omaenda andmete formaat. Naeruväärne ja lausa võimatu oleks püüda kõiki neid formaate realiseerida. Parem on leida mõni universaalne kalender-märkmiku andmete vorming, mida kasutatakse laias hulgas programmides. Üheks selliseks formaadiks on *iCalendar* (tuntud ka kui *iCal*), mis on *RFC* 2445 standard [12]. *ICalendar*'i tunnevad näiteks sellised gigandid nagu: *Microsoft Outlook*, *Apple iCal* ja *Mozilla* kalender (kaasaarvatud *Mozilla Sunbird*) [13]. *ICalendar* formaat loodi 90'ndatel *Internet Engineering Task Force* (*IETF*) poolt võttes suuresti eeskuju juba olemas olnud *vCalendar* formaadist, mille autoriks on *Internet Mail Consortium* (*IMC*). *VCalendar* peamine eesmärk oli võimaldada e-kirjade abil kalendriandmeid automaatselt saata ja vastu võtta. *ICalendar* andmete *MIME*-tüüp on "text/calendar" ja faililühendiks enamasti "ics". *Apple Macintoshis* on faililühendiks hoopis "iCal". Võimalik on salvestada ka vaba või hõivatud ajajaotust, sellisel juhul peaksid faililühendid olema vastavalt "ifb" ja "iFBf". [13]

ICalendari ja *vCalendari* süntaks on äravahetamiseni sarnane. Mõlemad on tekstilised formaadid, mis koosnevad erinevate algus- ja lõputäiste vahele kirjutatud blokkidest. Peamine erinevus on vaid kasutusel olevate võimaluste hulgas, mis on uuemal standardil otseloomulikult suurem. Koodinäites 3 on toodud *iCalendar* kirje, mis oma lihtsuse tõttu vastab ka *vCalendar* standardile. Näites on kirja pandud ühekordne sündmus "Bastille Day Party", mis algab 14 juulil 1997 kell 17:00 ja kestab sekund vähem kui 11 tundi. Näide on võetud *RFC* 2445 standardist [12]. Aegade kirjutamisel tuleb tähele panna, et "T" eraldab kuupäeva ja kellaaega ning "Z"-lõpp näitab, et tegemist on *UTC* ajaga.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
DTSTART:19970714T170000Z
DTEND:19970715T035959Z
```

```
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

Koodinäide 3. Kalendrikirje iCalendar formaadis [12]

Standard näeb ette, et *iCalendar* objekte peab saama vahetada nii *SMTP*, *HTTP* kui ka infrapuna või mingisuguste neljandate kanalite kaudu. Tegemist on väga paindliku formaadiga, mille abil saab kirjeldada keerulisi ajastruktuure. *ICalendar*'i saab salvestada laias laastus nelja tüüpi objekte:

- VEVENT – tüüpiline kalendrikirje nagu näiteks kohtumine, loeng või sünnipäev
- VTODO – ülesanne või planeeritav tegevus
- VJOURNAL – ülevaade igapäevastest toimingutest
- *VFREEBUSY* – vaba ja hõivatud aja märkimine

VEVENT ja VTODO võivad omada veel mitmeid alamstruktuure nagu näiteks VALARM, mis kirjeldab sündmusest või tegevusest teatamise viisi. Iga struktuuri nimi algab v-tähega ja iga objekt on piiratud ridadega *BEGIN: struktuuri nimi* ja *END: struktuuri nimi.*

Iga andmerida algab enamasti atribuudi nimega, mille järgi on pandud koolon ning lisatud andmed. Rea maksimaalne pikkus on 75 tähemärki. Kui andmeid on rohkem, siis võib seda jätka järgmiselt realt, pannes rea esimeseks sümboliks tühiku. Koodinäide 4 esitleb ühte lõiku *iCalendar* failis, kuhu on salvestatud nii ühe kui ka mitmerealine andmeväli. Tähelepanu on eelkõige tühikutel!

```
SUMMARY:üherealine informatsioon
DESCRIPTION:väga pikk mitmerealine inf
 ormatsioon, mis läheb edasi ka
  kolmandal real.
```

Koodinäide 4. Ühe- ja mitmerealine andmeväli iCalendar failis.

Kuna valminud rakendus tegeleb ühe- ja mitmekordsete kalendrikirjete salvestamisega, siis pälvib eelkõige tähelepanu *VEVENT* osa. *VEVENT* võib omada lausa 32 erinevat atribuuti,

mis on rohkem kui küllalt ühe tavalise ajaplaneerija vajaduste katmiseks. Enamkasutatud väljad on järgmised:

- UID – kalendri omaniku unikaalne identifikaator. Sobib suvaline tekst, mis kirjeldab üheselt kalendri omanikku. Näiteks veebipõhise kalender-märkmiku poolt genereeritud *iCalendar* faili UID'ks sobib hästi kasutaja e-maili aadress.
- SUMMARY – kirje lühikokkuvõite
- DESCRIPTION – kirjeldus
- DTSTART – sündmuse algusaeg (*Date Time START*)
- DTEND – lõppaeg (*Date Time END*)
- PRIORITY – ühest suurem või võrdne täisarv, mis määrab ära kirje tähtsuse. Standard näeb ette kolme tähtsusklassi: kõrge (1-4), keskmine (5) ja madal (6-9). Väiksem väärtus näitab suuremat tähtsust.
- DTSTAMP – kirje loomise aeg
- RRULE – kordumisreegel, mis omab mitmeid olulisi alamatribuute: FREQ – kordumis- ühik (YEARLY, MONTHLY, WEEKLY jne); INTERVAL – kordumisperioodide vaheline samm; COUNT – maksimaalne kordumiste arv; UNTIL – kordumiste lõppemise kuupäev. [12]

```
BEGIN:VCALENDAR
PRODID:-//m2Rt//NONSGML MCalendar 1.0//EN
VERSION:2.0
METHOD:PUBLISH
BEGIN:VEVENT
UID:m2rt@mcalender
PRIORITY:4
DTSTAMP:20051026T164300
SUMMARY:Teoreetilise informaatika harjutus ruumis p-416
DTSTART:20040902T100000
DTEND:20040902T113000
RRULE:FREQ=WEEKLY;INTERVAL=1;UNTIL=20041104T000000Z
END:VEVENT
END:VCALENDAR
```

Koodinäide 5. Korduv kalendrikirje iCalendar formaadis.

3.2 xCal

XCal formaat on *iCalendar* formaadi *XML*-kuju. Võimaluste, parameetrite ja nende väärtuste hulk on *xCal*'is täpselt sama, mis *iCalendar* formaadilgi. Kõik see on üle viidud oluliselt kergemini töödeldavale *XML*-vormingule. *MIME*-tüüp on *xCal*'il otseloomulikult sama, mis tavaliselt *XML*-failil – "application/xml" ja faililühendiks "xml". Kui lisada *xCal* failile *XSL*-stiilifail, siis on võimalik kalendrikirjet vaadata ka tavalises veebilehitsejas. Ka ei ole *xCal* struktuuril enam probleeme mahukate andmetega. Neid saab hoida nagu tavalises *XML*'is. Koodinäites 6 on toodud korduv kalendrikirje *xCal* vormingus. Kasutatud on samu andmeid, mis näites 5, kus vorminguks oli *iCalendar*. [14]

```
<?xml version="1.0" encoding="UTF-8"?>
<iCalendar>
<vcalendar version="2.0"
prodid="-//m2Rt//NONSGML MCalendar 1.0//EN">
<vevent>
<uid>m2rt@mcalender</uid>
<priority>4</priority>
<dtstamp>20051026T165338</dtstamp>
<summary>Teoreetilise informaatika harjutus p-416</summary>
<dtstart>20040902T100000</dtstart>
<dtend>20040902T113000</dtend>
<rrule>FREQ=WEEKLY;INTERVAL=1;UNTIL=20041104T000000Z</rrule>
</vevent>
</vcalendar>
</iCalendar>
```

Koodinäide 6. Korduv kalendrikirje xCal formaadis

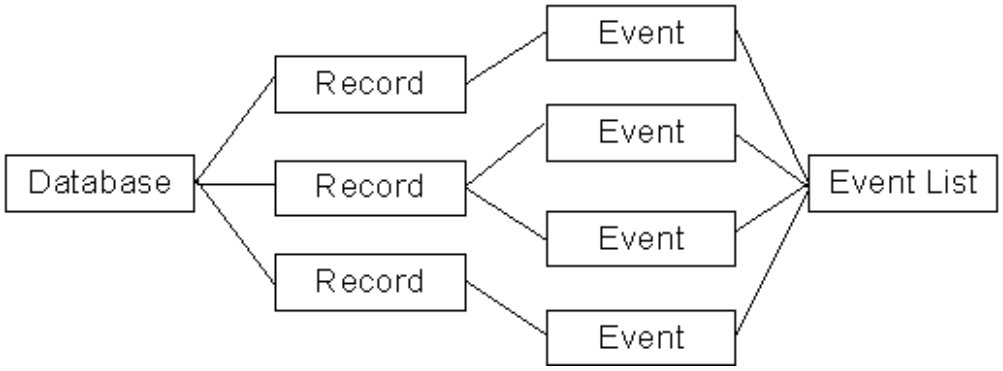
4. LOODUD PROGRAMM

4.1 Mobiilisüsteemi ülesehitus

Mobiilisüsteemi jaguneb laias laastus kolme ossa: tuum, see tähendab äriloogikaklassid, kuvatavad ekraanivormid ja korduvad kasutajaliidese osad. Viimaseid kasutatakse mitmetel vormidel ning on eraldatud vaid kodeerimise mugavuse ja kiiruse mõttes. Peaaegu iga kuvatav aken on tehtud eraldi klassiks, mis olenevalt vajadusest on tuletatud kas tavalisest *J2ME Form*'ist, *List*'ist või juba iseloodud kõrgema tasandi klassist. Näiteks on seadmete muutmise, andmete sisestamise kui ka mitmed otsinguaknad eraldi klassid. Äriloogikat realiseerivaid klasse on oluliselt vähem kui kasutajaliidesega tegelevaid, kuid esimeste tähtsus on seevastu tunduvalt suurem. Peamised äriloogikaklassid on *Record*, *Event* ja *EventList*.

4.2 Kalendrikirjete abstraktsioon

Ajaplaneerija üks keerukamaid ülesandeid oli leida viis, kuidas salvestada korduvate sündmuste struktuuri nii, et ei oleks ülemäärast andmete dubleerimist ning lisaks säiliks ka süsteemi terviklikkus. Allolev joonis 5. kujutab peamist kirjete teisendamise struktuuri. Alustades joonise lugemist vasakult näeme, et kõiki andmed hoitakse kusagil andmebaasis (*Database*). Baas võib sisaldada terve hulk kirjeid (*Record*), mis igaüks kirjeldavad teatud kordumismustriga sündmusteahela. Seega võib üks andmebaasikirje sisaldada infot mitme tegeliku sündmuse (*Event*) kohta. Näiteks baasis on kirje "kõrgem matemaatika II", mis kirjeldab ära terve kõrgema matemaatika kursuste loenguplaani. Olenevalt vaatluse all olevast perioodist võib antud kirje tähendada meile null kuni kasvõi tohutul hulgal sündmuste infot. Vaadeldes kahenädalast ajavahemikku võime öelda, et seal on neli matemaatika loengut. Muutes vahemikku suuremaks, kasvab ka loengute arv. Perioodil toimuvatest sündmustest koostatakse ajakava (*EventList*). Ajakava määrab ära perioodi alguse, lõpu kui ka filtri, mille alusel baasikirjetest (*Record*) sündmusi laetakse.



Joonis 5. Andmete loogiline kihistumine

Andmebaasiks kasutataks mobiili standardset *recordStore* süsteemi. *RecordStore* on seadmesse sisseehitatud salvestusvahend, mis oma olemuselt on justkui tavaline binaarfail, millele on lisaks kirjete filtreerimise ja sorteerimise süsteem. *RecordStore* on mõeldud peamiselt binaarkujul andmete salvestamiseks. Antud programmi vajadustele sobib see ideaalselt.

Record on klass, mis hoiab sündmusteahela andmeid kui ka selle kordumismustreid. *Record* suudab end ka binaarkujule teisendada, binaarkujul andmetega algväärtustuda kui ka teha muid olulisi asju, et võimaldada enda laadimist *recordStorest*.

Event on klass, mille peamine eesmärk on hoida ühte kindlat kuupäeva ning viidata *Record* objektile. Seega viib *Event* kokku sündmuse andmed koos kordumismustriga ning kindla kuupäeva, millal valitud tüüpi tegevus võiks toimuda. Näiteks kui *Record* kirjeldab kõrgema matemaatika loenguplaani, siis *Event* on juba üks kindel loeng (nt. teisipäeval, 22. novembril kell 10:00 toimuv). Kuna *Event* ise ei oma sündmuse kirjeldust, siis ei ole ka midagi dubleeritud. Sellest hoolimata on meil võimalus tühistada vaid üks konkreetne sündmus ahelast, märkides ära vaid valitud eriolukorra. Ülejäänud kordumismuster kehtib edasi. Nii on võimalik märkida lõpmatult pikki sündmusteahelaid tühiselt väikeste kuludega. Lisaks on võimalik *Record*'itest vaid kindlatele nõuetele vastavaid *Event'*e genereerida. Selline filter võib olla näiteks tähtsus või kasvõi nädalapäev.

EventList koondab kokku teatud filtri ning perioodiga sündmused. *EventList* sisaldab piisavalt pädevat infot, et see kasvõi üks-ühele kasutajale ekraanile trükkida. Nimetatud teisendustega saadakse ka loodud ajaplaneerijas küllalt abstraktsetest andmetest juba väga konkreetsed päeva, nädala ja kuuplaanid.

4.3 Andmehoidlad

Programm kasutab oma töös kahte andmehoidlat (*recordStore) – settings* ja *records*. *Settings* hoiab endas süsteemi seadeid, nagu näiteks veebiportaali aadressi, kasutajanime, parooli ja veel mitmeid muid programmi tööks olulisi parameetreid. *Records* ülesandeks on kirjete talletamine. *Records* baasi salvestatakse objektide *record* andmed, mis sisaldavad tabelis 1 kirjeldatut.

Välja nimi	**Andmetüüp**	**Kirjeldus**
message	string	sündmuse selgitus
begin	long	algusaeg (millisekundites alates 1. jaanuar 1970 0:00)
priority	int	tähtsus
expireMode	int	aegumisviis
end	long	aeg, mille järel kirje aegub
maxRecurrence	int	kordumiste arv, mille järel kirje aegub
length	long	sündmuse pikkus
recurrence	int	kordumisviis
step	int	kordumissamm

Tabel 1. Salvestatava kirje väljad

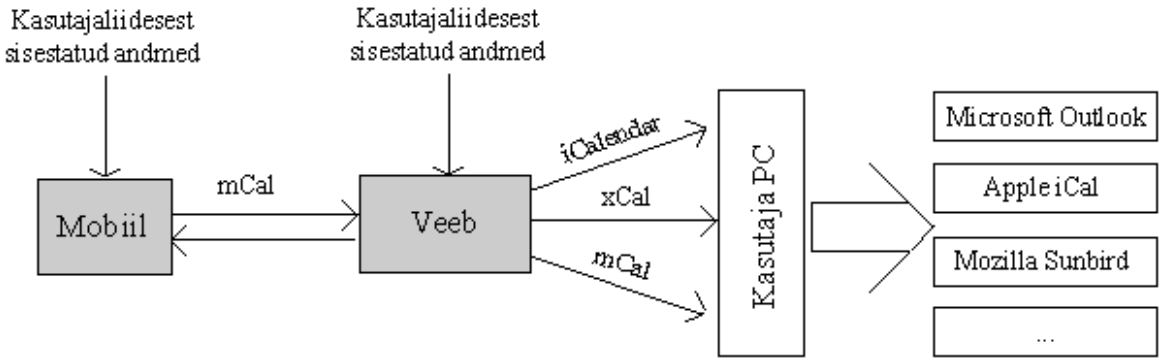
4.4 Andmevahetus

Andmevahetus veebiportaali ja mobiilirakenduse vahel käib *XML'*is omaloodud formaadi alusel. Esimene plaan nägi ette *xCal* standardi kasutamist, kuna aga loodud ajaplaneerija

realiseerib vaid tühist osa antud standardi funktsionaalsusest, siis tekkis probleem, et mida teha siis, kui süsteemilt nõutakse võimalusi, mida viimane ei suuda pakkuda. Selliste kaheldava väärtusega olukordade vältimiseks kasutatakse andmevahetuses *xCal*'ile sarnast, kuid oluliselt piiratud võimalustega formaati, millele nimeks sai *mCal* (vt. lisa 2 *mCal* faili struktuur).

XML'i lugemiseks on lisatud programmile vabavaraline *kXML parser*, mida saab laadida veebiportaalist http://kobjects.sourceforge.net/. Tegemist on väga kompaktse mobiiltelefonidele mõeldud *XML* parseriga, mis nõuab vähe ressursse ja seejuures on võimalikult kiire. *KXML* parseri kasutusjuhendi ning hulgaliselt näitematerjali leiab eelpool nimetatud veebilehelt.

Mobiilisüsteem suudab lugeda ja kirjutada vaid *mCal* formaati. Ka veebisüsteem suudab hetkel lugeda vaid *mCal*-formaati. Teiste vormingute lugemiseks ei ole olnud põhjust, sest süsteemi tööle ei ole otseselt vaja lisaks väliseid andmevorminguid. Kirjeid saab niigi sisestada läbi internetti ühendatud lauaarvuti või ka otse mobiilis. Väljundiks saadavaid vorminguid on aga mõnevõrra rohkem. Läbi veebisüsteemi on võimalik genereerida *iCalendar*, *xCal* ja loomulikult *mCal* faile. Joonis 6 kirjeldab üldist andmete liikumist loodud süsteemis.



Joonis 6. Andmete liikumine süsteemis

Mobiilisüsteemis, veebiportaalis ja lauaarvutis on võimalik andmeid käsitsi sisestada. Seejärel saab info läbi veebiportaali telefoni laadida ja hiljem muudatuste korral sinna tagasi

saata. Veebist saab tellida mitmeid andmeformaate, et info tagasi sobilikul kujul lauaarvutisse saada. Joonisel 6 kirjeldatud andmete liikumine ja vormindamine on eriti tähtis siis, kui soovida mõnda uut väljundvormingut juurde lisada. Uusi formaate saab lisada või vanu täiendada ilma, et mobiilis olevat tarkvara peaks uuendama. Kuna vormindamine on viidud piiratud võimsusega telefonist suurde serverisse, siis ei ole ka ressursisäästlukkus enam niivõrd oluline teema. Viimaseks oluliseks plussiks sellise skeemi juures on mugavus ja lihtsus. Vähesed suudavad mobiilist andmeid otse arvutisse suunata. Küll aga on levinud ja tuntud internetist failide allalaadimine. Praegusel juhul peab kasutaja oma nimega portaali sisse logima, vajutama sobival lingil ja soovitud vormingus fail laetaksegi arvutisse.

4.5 Andmeside efektiivsus

Andmeformaatide efektiivsuse uurimiseks lasti viiel inimesel kalender-märkmikusse igal ühel teha vähemalt 20 sissekannet. Kõik sissekanded pidid olema korrektsed ja sisaldama realistlikku infot. Sisestatud info ei tohtinud olla väga pikk, sest see pidi mahtuma ka mobiiltelefoni ekraanile. Tulemuste võrdlemiseks salvestati saadud andmed kolme tüüpi failideks: *iCalendar*, *xCal* ja omaloodud *mCal* formaati. Erinevate formaatide mahtude uurimiseks leiti ka binaarkujul andmete maht kokku. Seejärel leiti uuritavate vormingute suurus võrreldes binaarkujul olevate andmete mahuga.

Kasutatud valem:

```
vormingu suurus võrreldes binaarkujuga = vormingu faili maht / binaarkujul olevate andmete maht
```

Katse tulemus oli mõnevõrra etteaimatav (vt. tabel 2 faili suuruse uuringu tulemused). Kõige ebaefektiivsemaks osutus *xCal* formaat, mis oli ligikaudu üheksa korda mahukam kui binaarkujul salvestatud algandmed. Eelkõige tekitasid vormingus tarbetut mahtu sõnalised *XML*'i *tag*'id. Näiteks kulus sündmuse prioriteedi salvestamiseks lausa 22baiti, mida oleks saanud teha kasvõi ühe baidiga. Samal põhjusel võttis ka *mCal* palju ruumi. *XCal*'i ja *mCal*'i efektiivsuse vahe seisnes vaid *mCal*'i pisut lühemates *tag*'ide nimedes. Parima tulemuse

saavutas *iCalendar*, kus enamus parameetreid ei oma lõputähiseid ja selle arvelt on ka maht väiksem. Väga piiratud vahenditega mobiiltelefonis on iga bait arvel!

	iCalendar	xCal	mCal
Faili suurus võrreldes andmemahuga	~7x	~9x	~8x

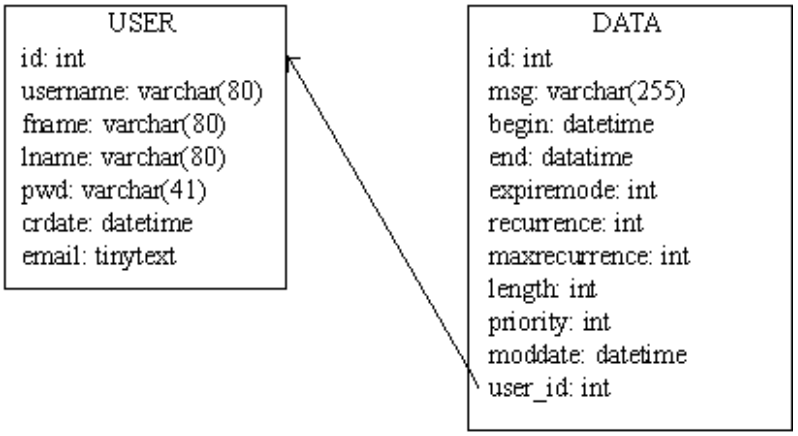
Tabel 2. Faili suuruse uuringu tulemused

Andmeformaatide töötlemise efektiivsuse uurimiseks katseid ei korraldatud. Küll aga on selge, et kõikvõimalike keerukate struktuuride töötlemine nõuab seadmelt suuremat jõudlust kui kindlaks määratud järjekorras binaarkujul andmete lugemine eales võtta võiks. Kui aga seadme töökiirus oluliseks takistuseks ei ole, siis ühilduvuse ja selguse mõttes on soovitatav kasutada mõnda tekstilist ning iseselgitavat vormingut nagu seda on *XML*. Katsetused näitasid, et vabavaralise *KXML* parseriga *XML*'i genereerimine mobiiltelefonis võtab küll aega ja ressursse, kuid jääb siiski taluvuse piiridesse. Sellepärast baseerub ka loodud kalender-märkmiku ja veebiliidese vaheline suhtlus *XML*'l. Küll aga oleks optimaalsem kui seesama suhtlus oleks binaarkujul ning omaloodud protokolli alusel.

4.6 Veebisüsteem

Ajaplaneerija juurde kuuluv veebisüsteem asub aadressil www.m2rt.pri.ee/mkalender. Tegemist on programmeerimiskeeles *PHP* ja andmebaasimootoriga *MySQL* tehtud portaaliga, kuhu saab uusi kirjeid lisada, muuta kui ka kustutada ja vaadata. *MySQL* baas koosneb vaid kahest tabelist: kasutajate tabel *users* ja kirjete tabel *data* (vt. tabel 3 andmebaasiskeem). Iga kirje juures on märge, kellele see kuulub. Tabelite veerud kannavad enda otstarvet selgitavaid nimetusi ja nende ümberjutustamisel pole suuremat tarvidust. Veebisüsteemi teiseks ülesandeks on andmete tõlkimine erinevatesse formaatidesse. Hetkel suudab süsteem anda väljundiks peale omaloodud *mcal* formaadi ka *iCalendar* ja *xCal* standarditele vastavaid faile. Tänu sellele saab nii mobiilis kui ka veebilehel loodud kirjeid üle viia mistahes teise tänapäevasesse kalendrisüsteemi, seal hulgas *Microsoft Outlooki.*

Kuna andmete formaatija asub alles veebitasandil, võib väljundstruktuure lisada ilma mobiilirakendust muutmata. See annab vabaduse süsteemi vajadusel hiljem täiendada.



Tabel 3. Andmebaasiskeem

Portaali kujundamisel on püütud olla võimalikult ülevaatlik ja minimalistlik. Veebisüsteem koosneb vaid piiratud hulgal alamlehtedest: logimine, sissekannete vaatamine, uute kirjete lisamine, abi ja veel mõned lehed. Eesmärk on olla kiiresti ja kergesti arusaadav. Kasutatud on neutraalseid ja silmale sõbralikke värve (vt. joonis 7 veebisüsteemi kujundus).

Joonis 7. Veebisüsteemi kujundus

4.7 Mobiilirakenduse paigaldamine

Mobiilirakenduse paigaldamine on tehtud kasutajale võimalikult lihtsaks. Lisaks tavapärasele veebiportaalile loodi spetsiaalne wap-lehekülg, millel pakutakse mobiilirakenduse allalaadimise võimalust. Programmi installeerimiseks peab kasutaja minema oma telefoni wap-lehitsejaga aadressile http://m2rt.pri.ee/mkalender/download.wml. Avaneval lehel on ainult üks link "Lae MKalender". Valides lingi ilmub ekraanile info programmist ja küsimus, et kas laadida programm. Pärast tingimustega nõustumist paigaldatakse programm telefoni (vt joonis 8 Telefoni installeeritud MKalender.). Seejärel võib hakata tarkvara kasutama. Esimeseks tegevuseks on soovitav minna "seaded" lehele ja sisestada oma veebiportaali kasutajanimi ja parool. Vajadusel võib muuta ka teisi seadistusi.

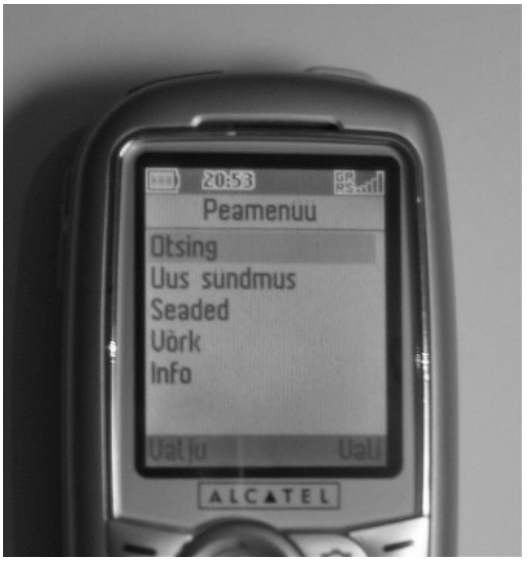
Joonis 8. Telefoni installeeritud MKalender

Programmi paigaldamisel tekivad probleemid vaid kasutajatel, kelle telefon ei vasta rakenduse nõuetele. Olenevalt telefoni mudelist võib tulla erinevaid veateateid, mis kõik viitavad sellele, et soovitud *Java* programmi miinimumtingimused ei ole täidetud. Veateadete saatjaks on telefon. Loodud rakendusel siin asja ei ole.

5. TAGASISIDE KASUTAJATELT

5.1 Mobiilirakenduse plussid

Kasutajatele meeldis mobiilirakenduse juures enim selle loogiline ülesehitus. Kohe pärast käivitumist kuvatakse menüü, mis on järjestatud tegevuste esinemissageduste järgi. Esimesel kohal on "Otsing", mille alla on koondatud kõik kirjete vaated – alustades päeva- ja kuuplaaniga ning lõpetades suvalisel perioodil toimuvate sündmuste otsinguga. Enamasti avatakse kalender just nimelt selle pärast, et sealt midagi vaadata ja kuna otsing on nimekirjas esimene, siis ei ole vaja rohkem nuppe vajutada, et jõuda õige valikuni (vt. joonis 9 Laetud rakenduse esimene ekraanipilt).



Joonis 9. Laetud rakenduse esimene ekraanipilt

Kiita sai ka avatud ekraanidelt tagasi peamenüüsse liikumine. Kõikidek ekraanivormidel on nupp "Tagasi", mis viib ühe taseme võrra peamenüüle lähemale. Seesugune navigeerimine oli ka algajatele kergesti mõistetav. Viimaseks oluliseks plussiks loeti rakenduse võrdlemisi kiire käivitumine. Laadimisel ei tee programm midagi mahukat ja ka koodi hulk on võrreldes keskmise *Java* mänguga kaduvväike.

5.2 Mobiilirakenduse miinused

Mõned ekraanivormid omavad liiga palju infot ja seetõttu on raskesti arusaadavad. Üheks selliseks vormiks on uue sündmuse sisestamise aken. Sündmuse parameetreid on küllalt palju ning nende nimekiri venib mitme leheküljepikkuseks, mis omakorda halvendab ülevaadet sisestavatest andmetest (Vt. joonis 10 Uue sündmuse sisestamine reaalses telefonis.). Ekraan koostati nii mahukana ainult sellepärast, et arenduseks kasutatud mobiiltelefoni *emulaator* omas keskmisest oluliselt suuremat ekraani ning probleemi ei osatud ette näha.

Joonis 10. Uus sündmuse sisestamine reaalses telefonis

Viimaseks rakenduse nõrgaks kohaks loeti see, et sünkroniseerida saab vaid kõiki andmeid korraga. Lahenduseks pakuti võimalus, kus kasutaja valib ise kirjed, mis saata või vastu võtta. See aga muudaks andmevahetuse oluliselt keerulisemaks protsessiks. Praeguse lahenduse eelis seisneb peamiselt lihtsuses ja arusaadavuses.

KOKKUVÕTE

Valminud ajaplaneerimissüsteem ei ole mahult väga suur. Veebiportaali ja mobiiliprogrammi loomiseks kulus vaid paar nädalat. Selle ajaga õnnestus luua töötav näide ühest võimalusest, kuidas siduda omavahel olemasolev veeb, lauaarvuti ja mobiiltelefon. Seega võib tõdeda, et olemas on head vahendid muutmaks info paremini ning mugavamalt kättesaadavaks. Seesuguste rakenduste vähesuses võib süüdistada vaid tehnoloogia uudsust.

Töös lahati mobiilse ajaplaneerija loomiseks vajalikke teadmisi:

- anti põgus ülevaade *Java 2 Micro Edition* platvormist,
- lahati mobiiltelefonidele programmeerimise kitsaskohti,
- tutvustati Interneti kalendristandardeid *ICalendar* ja *xCal*,
- kirjeldati loodud mobiilirakenduse peamist arhitektuuri,
- analüüsiti veebiportaali ja mobiilirakenduse vahelist andmevahetust,
- vaadeldi loodud süsteemi kasutajate esmamuljeid.

Töö eesmärgiks oli reaalse näite najal demonstreerida ühte viisi, kuidas olemasolevaid vahendeid paremini ära kasutades saab pakkuda veelgi paremat juurdepääsu oma andmetele ja luua uusi võimalusi rakenduse kasutamisel. Töö on kirjutatud ülevaatlikus vormis ja annab lugejale ettekujutuse sarnase süsteemi loomeprotsessist ning vajalikest teadmistest.

SUMMARY

The objective of the bachelor thesis was to study a possibility of linking together two different types of digital devices, a personal computer and a Java-enabled mobile phone, by using the Internet. The physical results of the work are two applications that are able to exchange data – a personal calendar organizer and time manager that runs in a Java-enabled mobile telephone and a web application that resides in a web server in the Internet. The theoretical part of the thesis focuses on presenting and explaining the technical solutions behind the mentioned systems.

The thesis is separated into two major topics. The first topic, the introductory material and theoretical background gives a brief overview of *Java 2 Micro Edition* platform, describes the problems and unconventional aspects of writing programs for Java-enabled mobile devices and introduces Internet calendar standards *ICalendar* and *xCal*.

The second half of the thesis gives a thorough description of the created applications. The emphasized aspects are:

- significant architectural solutions and decisions,
- data model and data structures used,
- data interchange formats and methods between the mobile phone and the web server,
- user reviews and opinions about the applications.

The thesis is structured so that it follows the creation process of the applications – from theoretical background to system analysis and design, to development aspects and problems ending with a look into end-user experience.

KASUTATUD KIRJANDUS

1. “Sun Microsystems biography”.
   http://sun-microsystems.biography.ms/, 10.11.2005
2. “Java technology: The early years”.
   http://java.sun.com/features/1998/05/birthday.html, 10.11.2005
3. “Java programming language”.
   http://en.wikipedia.org/wiki/Java_programming_language, 20.11.2005
4. “J2ME”.
   http://java.sun.com/j2me/docs/j2me-ds.pdf, 20.08.2004
5. “CLDC whitepaper”.
   http://java.sun.com/products/cldc/wp/CLDC_HI_WhitePaper.pdf, 20.08.2004
6. John W. Muchow. 2002. Core J2ME Technology & MIDP. Sun Microsystems, inc.
7. “MIDP Characteristics”.
   http://java.sun.com/j2me/docs/alt-html/midp-style-guide7/midp-char.html, 20.08.2004
8. Jonathan Knudsen. 2003. Wireless Java: Developing with J2ME, Second Edition. Apress.
9. Siemens AG. 2002. User Interface Guidelines for J2ME / SMTK
10. “MIDP Programming with J2ME”.
    http://www.developer.com/java/j2me/article.php/10934_1561591_1, 24.09.2004
11. “Big design for small devices”.
    http://www.javaworld.com/javaworld/jw-12-2002/jw-1213-j2medesign.html, 06.12.2005
12. “Internet Calendaring and Scheduling Core Object Specification”.
    http://www.ietf.org/rfc/rfc2445.txt, 12.11.2005
13. “iCalendar”.
    http://en.wikipedia.org/wiki/ICalendar, 15.11.2005
14. “iCalendar in XML Format”.
    http://www.ietf.org/internet-drafts/draft-royer-calsch-xcal-03.txt, 18.11.2005

KASUTATUD LÜHENDID JA MÕISTED

API - *Application Program Interface* ehk rakendusliides on reeglistik, mille alusel rakendusprogramm saab operatsioonisüsteemilt teenuseid (süsteemifunktsioone).

Applet – kliendi veebilehitseja poolt käideldav spetsiaalne *Java*-rakendus.

CDC - *Connected Device Configuration* on standard, mis kehtestab jõudluselt paremate *Java* toega väikeseadmete riistvaralised nõuded.

CLDC - *Connected, Limited Device Configuration* on standard, mis kehtestab jõudluselt nõrgemate *Java* toega väikeseadmete riistvaralised nõuded.

HTML - *Hyper Text Mark-Up Language* ehk hüperteksti märgendkeel.

HTTP - *Hypertext Transfer Protocol* on *TCP/IP* klient-server-protokoll *HTML*-dokumentide edastuseks veebis.

ICALENDAR – *Internet Calendar* on kalendriandmete vormindamise standard Internetis

IETF - *Internet Engineering Task Force* on organisatsioon, mis tegeleb Interneti standardiseerimisega.

IMC – *Internet Mail Consortium* on organisatsioon, mis tegeleb Interneti sõnumite standardiseerimisega.

J2EE – *Java 2 Enterprise Edition* on *Jav*a arendusplatvorm serveritele

J2ME– *Java 2 Micro Edition* on *Java* arendusplatvorm mikroseadmetele

J2ME - *Java 2 platform, Micro Edition* on *Java* platvorm vähenõudlikele väikeseadmetele nagu näiteks mobiiltelefonid ja pihuarvutid.

J2SE – *Java 2 Standard Edition* on *Java* arendusplatvorm lauaarvutitele

JSP – *Java Server Pages* on *Java* vahend dünaamiliste veebilehtede loomiseks

JVM - *Java Virtual Machine* ehk *Java* virtuaalmasin.

KVM - *K Virtual Machine* ehk *K* virtuaalmasin on väheste ressurssidega väikeseadmetele optimeeritud *Java* virtuaalmasin, mis võrreldes standardse *Java* virtuaalmasinaga omab teatavaid piiranguid.

KXML – vabavaraline *J2ME* platvormil töötav *XML*-parser.

MIDP - *Mobile Information Device Profile* on standard, mis kirjeldab *J2ME* töökeskkonna peamise funktsionaalsuse.

MySQL – vabavaraline relatsioonilise andmebaasi mootor.

PC – P*ersonal Computer* ehk personaalarvuti.

PHP - *PHP Hypertext Preprocessor* on skriptimiskeel dünaamiliste veebilehtede loomiseks.

QWERTY-klaviatuur – standardne lauaarvuti klaviatuur, mille esimesed tähed ülevalt vasakust reast on vastavalt QWERTY.

Servlet – veebiserveris töötav spetsiaalne *Java*-rakendus, mille peamiseks ülesandeks on genereerida dünaamilisi veebilehti.

SMS - *Short Message Servic* on *GSM*-telefonide sõnumiteenus, mille abil on võimalik saata kuni 160 tähemärgiseid teateid teisele telefonile.

SMTP - *Simple Mail Transfer Protocol* ehk „lihtne meiliedastusprotokoll" on protokoll, mida kasutatakse sõnumite (e-mailide) edastuseks serverite vahel.

UTC - *Universal Time Co-ordinated* on rahvusvaheline ajastandard, mis baseerub Greenwichi ajal (*GMT'l*).

virtuaalmasin - programmikeskkond, mis imiteerib riistvaralist süsteemi.

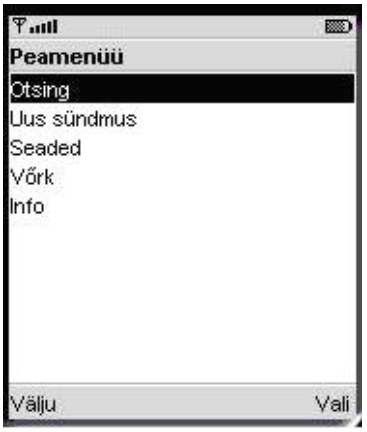
XCAL – *XML'*l baseeruv kalendriandmete vormindamise standard Internetis.

XML - *Extensible Markup Language* ehk laiendatav märgistuskeel.

XSL - *Extensible Stylesheet Language* ehk laiendatav stiililehtede keel.

LISA 1. MOBIILIROGRAMMI KASUTUSJUHEND

Programmi käivitamisel kuvatakse peamenüü. Menüüs olevad valikud on ennast ise kirjeldavad. Kui valitud toiming on sooritatud, siis tuleb programm alati tagasi peamenüüsse. Menüüs liikumiseks vajutage telefoni nuppe „üles“ või „alla“. Programmist väljumiseks vajutage „välju“ või nii kaua nuppu „tagasi“, kuni ilmub nupp „välju“.



Joonis 1. Programmi esimesed valikud

1.1 Sündmuse lisamine

Uue sissekande lisamiseks valige peamenüüst „uus sündmus“. Seejärel kuvatakse uue kalendrikirje lisamise vorm Vormil on kuus lahtrit, mis peavad sisaldama järgmist:

- **Sündmus** : lühikirjeldus toimuvast sündmusest.
- **Tähtsus**: sündmuse olulisus. Valides ekraani alt äärest nupu „muuda“, on võimalik seda ja ka kõiki teisi hetkel valitud parameetreid muuta.
- **Algab**: esimese selletaolise sündmuse algusaeg. Näiteks: esimese matemaatika loengu algusaeg on 2. september 2005 kell 10:00.
- **Kestab**: ühe selletaolise sündmuse kestvus. Matemaatika loengu kestvuseks on seatud 90 minutit ehk 1,5 tundi.
- **Kordub**: sündmus võib korduda igal aastal, kuul, nädalal, päeval või tunnil. Muuta saab ka sammu kordumise perioodi vahel. Loeng kordub igal nädalal.

- **Aegub**: korduval sündmusel saab märkida selle aegumise kuupäeva või aegumise korra. Matemaatika loeng aegub 14 korra möödumisel.

Joonis 2. Uue kirje lisamine

1.2 Kirjete kuvamine ja otsing

Olemasolevate kirjete kuvamiseks või otsimiseks tuleb minna peamenüüst otsingusse. Esmalt kuvatakse otsingu ajavahemikku täpsustav ekraan. Välja on toodud levinumad otsinguvahemikud. Võimalus on ka valida suvaline ajavahemik või paluda kuvada kõik kirjed.

Joonis 3. Otsinguvahemiku täpsustamine

Järgmisel sammul ilmub nimekiri kõikidest valitud perioodil toimuvatest sündmustest. Kui sündmuse kirjeldus on lubatud piirist pikem, siis näidatakse vaid selle algust ja kirje lisatakse

lõppu kolm punkti. Täpsemat infot kirje kohta saab näha, kui valida menüüst „Vaata“. Iga rea esimene märk tähistab kirje tähtsust. Mida rohkem on triipe, seda tähtsam:

- „=“ - tähtis
- „-„ - keskmine
- „ „ - vähetähtis

Kui sündmusi on palju, siis lükatakse kõik ühetaolised kokku, seejuures märgitakse ära nende arv ja esimese sündmuse algusaeg. Nt. kirje *„- 11 x matemaatika .. 10:00 23.09.04“* tähendab, et valitud perioodil toimub üksteist keskmise tähtsusega matemaatika loengut, millest esimene leiab aset 23. septembril kell 10:00. Arvu, millest alates sarnased kirjed üheks lükatakse, saab seadistada.

Kui sündmusi on veelgi rohkem, siis lõpetatakse loendamine ja lisatakse „>“ märk.

Näiteks: *„>50 x igal hommikul.. 8:45 23.09.04“* tähendab, et valitud perioodil toimub rohkem kui 50 korda sündmus „igal hommikul ..“. Maksimaalne loendamiste arv on sammuti muudetav.

Kuvatud sündmuste nimekirja saab sorteerida nii aja, hulga kui ka tähtsuse alusel. Võimalus on filtreerida vaid õigel tähtsusekünnisel olevad kirjed.

Joonis 4. Otsingu 7 päeva lõikes koos valikute menüüga

1.3 Andmevahetus veebiportaaliga

Andmevahetuseks tuleb peamenüüst klõpsata valikult „võrk“, mis võimaldab kõiki kirjeid Internetti saata või sealt allalaadida. Esmalt tuleb määrata andmevahetuse suund. Valides „Lae kõik“ asendatakse kõik olemasolevad kirjed võrgust laetutega! Olenemata valikust,

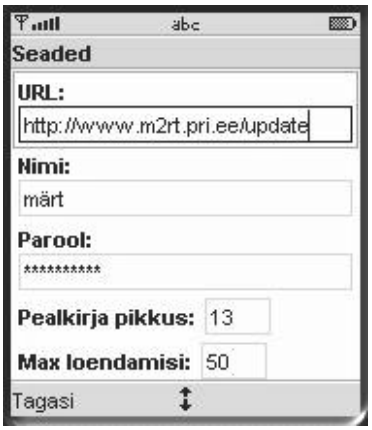
kuvatakse järgmisel ekraanil andmevahetuse hetkeolukord. Kui andmed on saadetud, siis võib minna tagasi peamenüüsse. Võimaliku vea korral informeeritakse kasutajat. Peamine veapõhjus on vale kasutajanimi, parool või veebiaadress.

Joonis 5. Andmevahetus veebiportaaliga

1.4 Seadete muutmine

Seadete ekraan koosneb kuuest lahtrist. Kõik muudatused salvestatakse automaatselt!

- **URL**: veebiportaali aadress, millega andmeid uuendatakse.
- **Nimi**: veebiportaalis oleva konto kasutajanimi
- **Parool**: veebiportaali konto parool
- **Peakirja pikkus**: maksimaalne otsingus kuvatava sündmuse kirjelduse pikkus
- **Max loendamisi**: suurim sündmuste loendamise arv otsingus
- **Max sarnaseid kirjeid**: suurim sarnaste kirjete arv otsingus



Joonis 6. Seadete muutmise aken

LISA 2. MCAL FAILI STRUKTUUR

MCal on *XML*-fail, mis võib sisaldada ühte või mitut kalendrit. Iga kalender (*calendar*) saab omada mitut sündmust (*event*), mis omakorda võivad omada järgmisi välju:

- summary – sündmuse lühikirjeldus
- priority – tähtsus. Võib omada kolme väärtust: HIGH, MEDIUM ja LOW.
- begin – algusaeg (millisekundites alates 01.01.1970 0:00)
- duration – kestvus
- recurrence – korduva sündmuse kordumisperiood. Võimalikud väärtused: YEARLY, MONTHLY, WEEKLY, DAILY või HOURLY.
- interval – kordumisperioodide vaheline samm (täisarv)
- count – maksimaalne kordumiste arv
- until – kordumiste aegumise kuupäev

```
<?xml version="1.0"?>
<mcalendar><calendar>
      <event>
            <summary>kõrgem matemaatika II</summary>
            <priority>MEDIUM</priority>
            <begin>1109674800000</begin>
            <duration>5400000</duration>
            <recurrence>WEEKLY</recurrence>
            <interval>1</interval>
            <count>10</count>
      </event>
</calendar>
</mcalendar>
```

Koodinäide 1. MCal näitefail