СОФИЙСКИ УНИВЕРСИТЕТ
"СВ. КЛИМЕНТ ОХРИДСКИ"

Факултет по Математика и Информатика
Катедра „Информационни Технологии”

Д И П Л О М Н А Р А Б О Т А

Тема: *Уеб-базирана система за съвместно управление на проекти- слой база данни и бизнес логика*

Дипломант: ***Добрин Славов Иванов, фак. № М-21221***

Специалност: ***Информатика***

Специализация: ***Разпределени системи и мобилни технологии***

Научен ръководител: ***доц. д-р Боян Бончев***

София

2006

Съдържание

1. УВОД 6

1.1. Въведение 6
1.2. Цел и задачи на дипломната работа 7
1.3. Структура на дипломната работа 8
1.4. Благодарности 9

2. ОБЗОР НА СЪЩЕСТВУВАЩИТЕ СОФТУЕРНИ СИСТЕМИ В ОБЛАСТТА 10

2.1. Софтуерни решения за обсъждане, комуникация и съвместно вземане на решения .. 10
2.1.1. Функционалност, приложение, ограничения 10
2.1.2. Съществуващи решения 11
2.1.2.1. FirstClass 11
2.1.2.2. Web Crossing 13
2.1.2.3. EVE 13
2.1.2.4. BSCW 14
2.1.2.5. MS Exchange 15
2.2. Софтуерни решения за сътрудничество и съвместна работа 15
2.2.1. Функционалност, приложение, ограничения 15
2.2.2. Съществуващи решения 17
2.2.2.1. Groove 17
2.2.2.2. MS Netmeeting 17
2.3. Софтуерни решения за управление на проекти 17
2.3.1. Функционалност, приложение, ограничения 17
2.3.2. Съществуващи решения 19
2.3.2.1. ProjectPlace 19
2.3.2.2. ERoom 20
2.3.2.3. QuickPlace 20
2.3.2.4. MS Project 20
2.4. Софтуерни решения за представяне 21
2.4.1. Функционалност, приложение, ограничения 21
2.4.2. Съществуващи решения 22
2.4.2.1. Webex 22
2.4.2.2. Placeware 22
2.5. Софтуерни решения за проучване и анкета 23
2.5.1. Функционалност, приложение, ограничения 23
2.5.2. Съществуващи решения 24
2.5.2.1. Boomerang 24
2.5.2.2. Inquisite 25

3. АНАЛИЗ НА ПРОБЛЕМА. ПРОЕКТИРАНЕ НА РЕШЕНИЕТО 27

3.1. Анализ на изискванията 27
3.1.1. Характеристики на проектите, финансирани по договори 27
3.1.2. Характеристики на потребителите 29
3.1.3. Общи изисквания към платформата 29
3.2. Проект на решението 31
3.2.1. Основни понятия 32
3.2.1.1. Видове обекти 32
3.2.1.2. Работен прозорец и работно пространство 34
3.2.1.3. Същност и характеристики на основните обекти 34
3.2.2. Интерфейс 39
3.2.3. Функционалност 40
3.2.3.1. Обща функционалност 40
3.2.3.2. Функционалност по отношение на индивидуалните и групови ползватели 41
3.3. Дефиниране на актьорите и случаите на употреба за системата 45
3.3.1. Актьори 45
3.3.2. Случаи на употреба 46
3.3.2.1. Случаи на употреба за актьор Guest 46
3.3.2.2. Случаи на употреба за актьор User 47
3.3.2.3. Случаи на употреба за актьор Project Participant 48
3.3.2.4. Случаи на употреба за актьор Project Coordinator 49

3.3.2.5. Случаи на употреба за актьор Community Participant ... 50
3.3.2.6. Случаи на употреба за актьор Community Coordinator ... 51

4. ИЗБОР НА ТЕХНОЛОГИЧНИ ПЛАТФОРМИ ЗА РЕАЛИЗАЦИЯТА НА ПРОЕКТА ... 52

4.1. Избор на База Данни – MySQL ... 52

5. ОПИСАНИЕ НА РЕАЛИЗАЦИЯТА ... 54

5.1. Слой база данни ... 55
5.1.1. Таблица master ... 56
5.1.2. Таблица users ... 57
5.1.3. Таблица permissions ... 58
5.1.4. Таблица roles ... 58
5.1.5. Таблица role_permissions ... 59
5.1.6. Таблица access_rights ... 59
5.1.7. Таблица workspaces ... 60
5.1.8. Таблица messages ... 61
5.1.9. Таблица tasks ... 62
5.1.10. Таблица reminders ... 64
5.1.11. Таблица contacts ... 64
5.2. Слой бизнес логика ... 66
5.2.1. Проект на модела на данните ... 66
5.2.1.1. Описание на класовете и интерфейсите ... 67
5.2.2. Управление на цикъла заявка-отговор ... 76
5.2.2.1. Стратегия за управление на заявката ... 76
5.2.2.2. Абстракция на цикъла заявка-отговор ... 77
5.2.2.3. Жизнен цикъл на заявката ... 79
5.2.3. Модел на контрола на достъпа, базиран на роли ... 83
5.2.3.1. Реализация и управление на контрола на достъпа ... 84
5.2.4. Кеширане на обектите от домейна ... 88
5.2.4.1. Кеш в рамките на един цикъл заявка-отговор ... 88
5.2.4.2. Кеш в рамките на една потребителска сесия ... 89
5.2.5. Поведение на системата ... 89
5.2.5.1. Обработка на съобщенията ... 90
5.2.6. Запазване на обектите от домейна в базата данни ... 94

6. ТЕСТВАНЕ И ВНЕДРЯВАНЕ ... 96

7. ЗАКЛЮЧЕНИЕ ... 99

8. ИЗПОЛЗВАНА ЛИТЕРАТУРА ... 102

9. ПРИЛОЖЕНИЯ ... 105

9.1. Ръководство на потребителя ... 105
9.1.1. Как да започнем? ... 105
9.1.2. Регистрация ... 107
9.1.3. Влизане в системата WEPROM ... 109
9.1.4. Моето потребителско пространство. ... 110
9.1.4.1. Проекти ... 111
9.1.4.2. Групи по интереси ... 114
9.1.4.3. Задачи ... 117
9.1.4.4. Съобщения ... 118
9.1.4.5. Контакти ... 122
9.1.4.6. Календар ... 125
9.1.5. Управление на проект. ... 131
9.1.5.1. Проектен профил. ... 131
9.1.5.2. Членове на проект. ... 134
9.1.5.3. Задачи на проекта. ... 143
9.1.5.4. Календар на проекта. ... 153
9.1.5.5. Папки и документи ... 155
9.1.6. . Управление на група по интереси. ... 156
9.1.6.1. Профил на групата по интереси. ... 157
9.1.6.2. Членове на група по интереси. ... 159

9.1.6.3. Календар на групата по интереси. ... 168
9.1.6.4. Папки и документи ... 170
9.1.7. Хоризонтално меню за бърз достъп и навигация ... 171
9.2. ЛИСТИНГИ НА ЧАСТИ ОТ ПРОГРАМНИЯ КОД НА СЛОЯ БИЗНЕС ЛОГИКА ... 174
9.2.1. Пакет com.fmi.weprom.api ... 174
9.2.1.1. AccessRights.java ... 174
9.2.1.2. Calendar.java ... 174
9.2.1.3. Community.java ... 175
9.2.1.4. Contact.java ... 176
9.2.1.5. File.java ... 176
9.2.1.6. Message.java ... 177
9.2.1.7. Permission.java ... 179
9.2.1.8. Project.java ... 181
9.2.1.9. Reminder.java ... 182
9.2.1.10. Role.java ... 183
9.2.1.11. Task.java ... 184
9.2.1.12. User.java ... 185
9.2.1.13. UserWorkspace.java ... 187
9.2.1.14. WEPROMException.java ... 188
9.2.1.15. WEPROMFactory.java ... 188
9.2.1.16. WEPROMSystem.java ... 189
9.2.1.17. Workspace.java ... 190
9.2.2. Пакет com.fmi.weprom.util ... 190
9.2.2.1. WepromRequestCycle.java ... 190
9.2.2.2. WepromRequestCycleFactory.java ... 190
9.2.2.3. WepromRequestCycleUtil.java ... 190
9.2.2.4. WepromRequestFilter.java ... 193
9.2.3. Други по-интересни класове ... 195
9.2.3.1. WEPROMBasePage.java ... 195
9.2.3.2. PageValidateListener.java ... 196

Списък съкращения и специални термини

W-EPROM (Web-Environment for PROject Management) –	–	*Уеб-базирана система за управление на проекти*
Framework	–	*Рамка за приложения*
JSP	–	Java Server Pages
CIST (Centre of Information Society Technologies)	–	*Център за информационни обществени технологии*
Онлайн (online) –	–	*Конферирane в реално време по Интернет*
BBS	–	*Bulletin-Board System*
Chat	–	*Разговор/беседа*
Моментални съобщения (instant messaging)	–	*Синхронен обмен на съобщения от различен тип (текст, аудио, видео)*
Groupware	–	*Софтуер за съвместна работа*
Web-browser	–	*Програма за разглеждане на web-базирана информация*
URL	–	*Universal Resource Locator*
HTML	–	*HyperText Markup Language*
БД	–	*База данни*
UML	–	*Unified Modeling Language*
GUI (Graphical User Interface)	–	*Графичен потребителски интерфейс*

1. Увод

1.1. Въведение

Деловата сфера на днешния забързан и динамичен свят все по-основателно може да се нарече “базирана на проекти”. До известна степен, причина за това са класическите проекти, които големите стопански организации е нормално да изпълняват за осъществяване на своите дейности. Подобни проекти сега по същество имат същите характеристики, както и проектите в миналото, но в настоящата ситуация, когато “времето е пари” и всичко е свързано със строго спазване на срокове, организациите са принудени да влагат огромни човешки и други ресурси при тяхното изпълнение.

Ето защо, наличните софтуерни приложения и web-базирани услуги за планиране и управление на проекти, предлагат множество от услуги, които отразяват типични за този тип проекти сценарии, като например финансова отчетност, времево планиране и др.

Истинският мотив, обаче, за охарактеризиране на съвременността с определението “проектно-базирана” са проектите финансирани по договори. Обявяването на конкурси и отпускането на субсидии за финансиране на конкретни дейности е вече световна практика във всички области на стопанския, социалния и културния живот. Горното важи в пълна сила и за сферата на висшето образование, защото интензивността и продължителността на промените в тази област, както и хипер-конкурентността, предизвикана от глобализацията, изискват поемане на все повече колективна отговорност при осъществяване на възможностите и посрещане на предизвикателствата.

В тази връзка има нарастващо търсене, най-вече на web-базирани или с web-поддръжка услуги за планиране и управление на проекти, подходящи за проекти, финансирани по договори и разработвани от не особено голям колектив. Има и съответно предлагане на такива услуги, като цените им варират от безплатни, до високи пазарни.

В този ред на мисли, предметът на дипломната работа не е да запълни някаква празна ниша в софтуерните средства.

1.2. Цел и задачи на дипломната работа

Целта на дипломната работа е от една страна да се направи проучване на специфичните нужди от софтуер за планиране и управление на проекти в областта на висшето образование и научните изследвания, които се разработват на колективна основа и на принципите на самоорганизирането и самоуправлението, а от друга - да се разработят слоят „База данни” и слоят „Бизнес логика” на прототип на съответно web-базирано приложение за оперативно управление на проекти.

Постигането на поставените цели налага последователното решаване на следните задачи:

Задача 1. Проучване на проблемната област и теоретична обосновка на предлаганото решение за създаване на система за съвместната работа и управление на проекти;

Задача 2. Анализ на потребностите по отношение на подходящ софтуер в разглежданата проблемна област.

Задача 3. Предлагане на идеен проект на софтуерната система, обект на работата;

Задача 4. Проектиране на сценариите на употреба за софтуерното приложение въз основа на бизнес изискванията;

Задача 5. Избор на подходящи технологии за разработване на слоя „База данни” и слоя „Бизнес логика”;

Задача 6. Проектиране и разработване на софтуерната система с трислойна логическа архитектура, включваща три подзадачи, за проектиране и разработване на:

Задача 6.1. Слой на базата данни;

Задача 6.2. Слой на бизнес логиката;

Задача 6.3. Презентационен слой;

Задача 7. Тестване, оценка, усъвършенстване и внедряване.

Настоящата дипломна работа, представя решаването на задачи 1., 2., 3., 4. и 7. в съавторство с [52] и самостоятелното решаване на задачи 5, 6.1 и 6.2. от изброените по-горе.

1.3. Структура на дипломната работа

Дипломната работа се състои от 9 глави: „Увод", „Обзор на съществуващите софтуерни системи в областта", „Анализ на проблема. Проектиране на решението", „Избор на технологични платформи за реализацията на слоя база данни и слоя бизнес логика на проекта", „Реализация", „Тестване и внедряване", „Заключение", „Използвана литература" и „Приложения".

Уводът (**Глава 1.**) съдържа кратко въведение в областта на дипломната работа, формулират се целта и задачите на работата и начинът на структуриране на изложението й.

В **Глава 2.** „Обзор на съществуващите софтуерни системи в областта " е направен обстоен преглед на съществуващи системи за съвместната работа и управление на проекти и потребностите, които те задоволяват. Тук се решава първата от задачите (**Задача 1**) на дипломната работа, като се проучват основните проблемни направления и съществуващите решения в изграждането на такъв тип софтуер.

В **Глава 3.** „Анализ на проблема. Проектиране на решението**"** се прави анализ на потребностите по отношение на подходящ софтуер, с приложение в по-горе дефинираната проблемна област, формулира се идейният проект на конкретната програмна система, както и случаите на употреба, с което се решават **Задачи 2, 3** и **4.**

В **Глава 4.** „Избор на технологични платформи за реализацията на проекта" изцяло е решена **Задача 5**, като е обоснован направеният избор на база данни.

Част от решението на **Задача 6**, а именно **Задача 6.1** и **Задача 6.2** е описано обширно в **Глава 5 „Описание на реализацията"**. Тази глава е разделена на две части, всяка от които- съответна на отделен пласт от трислойната логическа архитектура на приложението. Първо са представени структурата и релациите на базата данни (БД), а след това начинът на и подходите при моделирането на данните на ниво бизнес логика.

В **Глава 6.** „Тестване и внедряване" са описани начините на тестване на системата и възможностите за нейното внедряване в реални условия и по този начин е решена **Задача 7**.

Заключението подробно представя приносите на дипломната работа и възможните перспективи за развитието ѝ.

Основният текст на дипломната работа се съпровожда от „Списък съкращения и речник на специалните термини”, списък с „Използвана литература”. Двете приложения от Глава 9. „Приложения“ включват „Листинги на части от програмния код на слоя бизнес логика” на разработената система и „Ръководство на потребителя”, което съдържа примерни екрани и обяснения относно работата с приложението и услугите, които то предлага на потребителите.

1.4. Благодарности

Особена благодарност дължа на научния си ръководител доц. д-р Боян Бончев за препоръките и помощта оказана ми през периода на разработка на дипломната работа. Благодаря и на своята колежка Невена Донева, с която заедно проектирахме и разработихме дипломната работа.

2. Обзор на съществуващите софтуерни системи в областта

За подобряване на съвместната работа по проекти и управлението на проекти, в съвременността, се разработват и използват няколко различни типа софтуерни решения: за обсъждане, комуникация и съвместно вземане на решения; за сътрудничество и съвместна работа; за управление на проекти; за представяне; за проучване и анкета.

В настоящата глава са разгледани характерните черти на различните категории софтуер, изброени по-горе, както и на някои типични техни представители, с цел да се направи преглед и анализ на тази проблемна област, в която попада и разработката по настоящата дипломна работа.

2.1. Софтуерни решения за обсъждане, комуникация и съвместно вземане на решения

Най-общо, системите за обсъждане, комуникация и съвместно вземане на решения типично се концентрират върху осъществяването на възможността – хората да общуват и обменят мнения и идеи, чрез добавяне на текстови съобщения в една обща папка.

2.1.1. Функционалност, приложение, ограничения

Разглежданата категория софтуерни приложения позволява осъществяване на различни по маниер дискусии – линейни, нишковидни, web- или email-базирани. Разбира се, всеки маниер налага свои изисквания, както относно медийната среда, която се използва, така и към начина, по който общуването се визуализира и структурира. Поради факта, че по време на дискусията, нуждите на отделните участници могат да се развиват, тези системи трябва да имат възможност за промяна на начина на комуникация между тях (като chat, моментални съобщения, ”бяла дъска” и др.), когато това е необходимо.

При по-развитите системи от този вид е възможен цялостен контрол, по отношение на това дали дискусията да бъде “за четене и/или писане”, ”публична и/или частна” и дали обсъждането да става чрез използване на web-страници и/или email. В някои от тях са включени и възможности за промяна на средството за комуникация, например промяна от форум в “chat в реално вре-

ме” или за избор на начин на визуализиране на публикуваните мнения и коментари – между линеен или нишковиден. По време на дискусия, мненията могат да се изразяват под разнообразна форма, например чрез попълване на анкета за бързо гласуване или чрез изпращане на „моментални съобщения”, или много други. Потребителят може да укаже настройки за уведомяване, което означава, че когато друг участник публикува отговор на негов въпрос или свое мнение във форума, първият ще получи съобщение на личния си email адрес.

Най-характерни ограничения при системите за обсъждане и съвместното взимане на решения са, че те рядко поддържат удобни средства за манипулиране и публикуване на електронно съдържание. Те не улесняват особено общуването по мрежа между участниците, като сътрудничеството между тях се осъществява само посредством различни комуникационни форми. Обикновено не е възможно директно да се работи съвместно върху съдържанието, на базата на предварително разпределени права за редактиране, превод и др. Комуникацията, която те предлагат е предимно чрез един основен асинхронен (не в реално време) начин на общуване. Поддръжката на синхронна (в реално време) комуникация, като моментални съобщения или chat, липсва или е слаба и поради това, не е възможно участниците в дискусията свободно да променят начина си на общуване.

2.1.2. Съществуващи решения

По-долу са разгледани особеностите на някои от най-популярните системи за обсъждане, комуникация и съвместно вземане на решения.

2.1.2.1. FirstClass

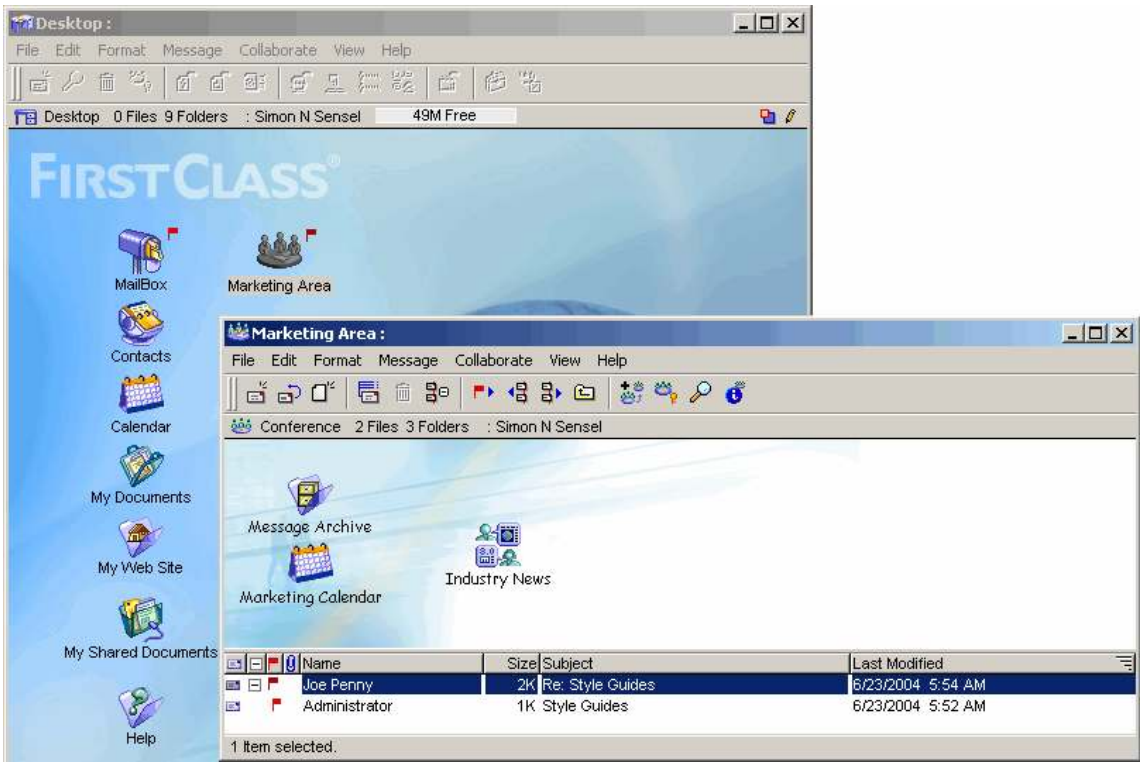
FirstClass [29] е система за електронна поща (email), конфериране в реално време (онлайн) по Интернет и BBS (Bulletin-Board System) за ОС Windows, Macintosh, и Linux.

FirstClass първоначално печели привърженици като BBS система за ОС Macintosh най-вече поради това, че има графичен потребителски интефейс подобен на предлагания от Macintosh [30]. Около средата на 90-те години на миналия век популярността на системат нараства с въвеждането на FirstClass-клиент за Microsoft Windows. Веднага след това обаче, поради бързото навли-

зане на Интернет, се налага системата да бъде доразвита и с email подсистема за корпоративно ползване. Благодарение на това, макар и бавно, продуктът заема една специфична ниша, най-вече в образователния пазар.

Днес FirstClass е собственост на корпорацията Open Text.

Най-новата версия на системата, FirstClass 8.3 [21] е рентабилна, много лесна за използване и богата на възможности система за обмен на съобщения и комуникация, подходяща за учебни заведения (дори в рамките на цял учебен кампус), обучаващи организации и бизнеса. FirstClass платформата (вж. Фиг. 2.1) е софтуер за колаборативна работа, потребителите на който могат ефективно да комуникират и съвместно да ползват ресурси и информация посредством email, конфериране, директории за общо ползване, chat, индивидуални и споделени календари. FirstClass се използва от хиляди организации за създаване на мощни онлаин общности, така че техните членове да изпълняват своите задължения по-ефективно.



Фигура 2.1. FirstClass платформата [21]

FirstClass поддържа също и специална опция, наречена „Унифицирана комуникация", която незабележимо интегрира в потребителската пощенска кутия

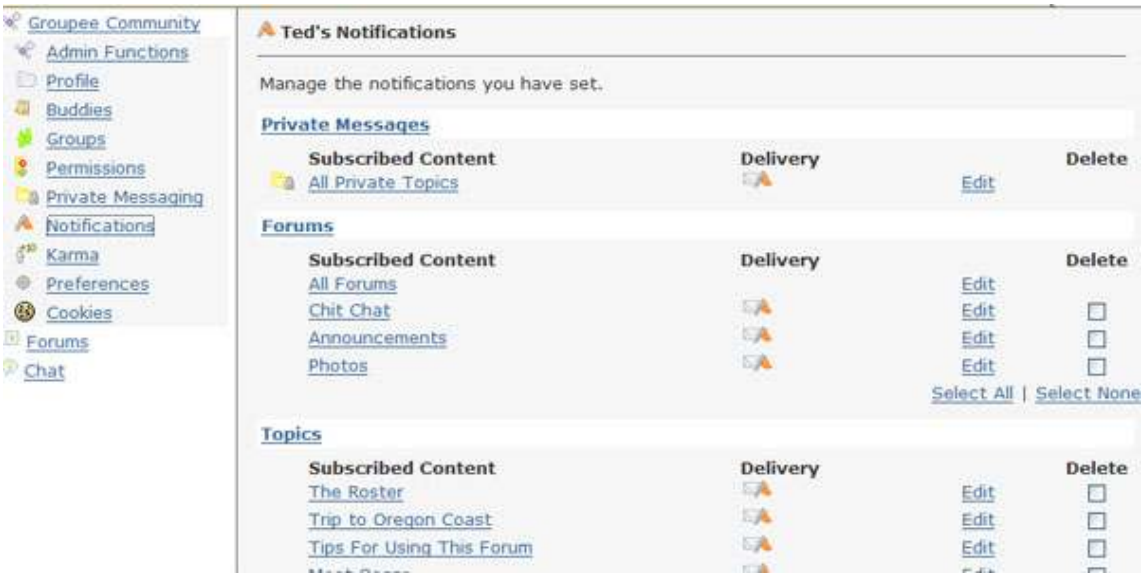
неговите email, гласови и факс съобщения. По този начин всеки потребител има широк набор от начини за достъп до всички свои съобщения по всяко време и на всяко място, чрез устройство по свой избор, включително чрез мобилен или обикновен телефон, персонален компютър, web-browser или персонален цифров асистент.

2.1.2.2. Web Crossing

Продуктите на Web Crossing [27] са насочени към широк спектър потребители от промишлената сфера и други приложни области, включително технологични фирми, общ бизнес, асоциации, организации с идеална цел и образователни институции, които имат нужда от изграждането на онлайн общности. Web Crossing продуктите включват BBS, форум, blog, пощенски списъци, средства за гласуване, Wiki и други инструменти за колаборативна работа (бреинсторм, социални контакти и др.).

2.1.2.3. EVE

Eve (известна по-рано като Ultimate BBS), сега се поддържана от Groupee Nation обществото, е богата на възможности услуга за организирани общности [7].



Фигура 2.2. Известяване в EVE [7]

Тя е хоствана услуга, която осигурява за крайния потребител платформа за онлайн взаимодействие и изява, включваща управление на права, админис-

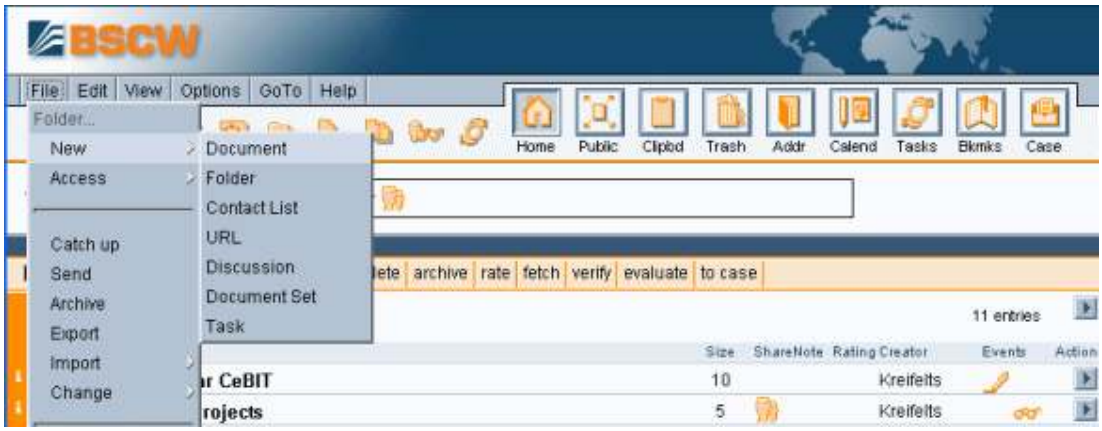
триране и търсене на общо съдържание, известяване (вж. Фиг. 2.2), RSS and Content Islands (за поддръжка на информираността на членовете), статистика, chat (позволяващи разговор в реално време между членовете на общността), както и платен достъп до специфични възможности и ресурси.

2.1.2.4. BSCW

BSCW (Basic Support for Cooperative Work) позволява web-базирано сътрудничество и взаимодействие [6].

Разработването на системата започва през 1996-1997 с финансова помощ по проекта CoopWWW. През 1998 и 1999 разработването на системата BSCW продължава като част от работата се извършва по проекта CESAR, финансиран от Европейската комисия, чрез програмата „Telematics Applications".

BSCW може да се нарече система за „споделено работно пространство", което позволява зареждане на документи, известяване за събития, управление на групи от потребители и др. Единственото, от което се нуждае потребителят за достъп до някое работно пространство е стандартен web-browser (вж. Фиг. 2.3).



Фигура 2.3. Споделеното работно пространство на BSCW [6]

Потребителите които имат интерес, могат да използват системата безплатно чрез специално поддържания публичен сървър. Освен това, всеки потребител може да използва свой собствен BSCW сървър, който може да бъде изтеглен и инсталиран. Сървърът работи на повечето Unix ОС, на Windows NT, Windows 2000 и Windows XP).

2.1.2.5. MS Exchange

MS Exchange [19] е приложен софтуер (разработен и поддържан от фирмата Microsoft), който осигурява за бизнеса удобна платформа за електронна поща, моментални съобщения, възможности за споделяне и управление на информацията, съвместна работа върху документи, изграждане на системи за документооборот (вж. Фиг. 2.4). В него са внедрени всички най-модерни системи за сигурност, като например електронен цифров подпис и електронен цифров плик.

Фигура 2.4. Възможности на MS Exchange [19]

2.2. Софтуерни решения за сътрудничество и съвместна работа

Програмните продукти за сътрудничество и съвместна работа обикновено имат за цел да осигурят защитена чрез парола среда, достъпна само за група участници, които комуникират, обменят на файлове, сътрудничат помежду си.

2.2.1. Функционалност, приложение, ограничения

При тези приложения, сътрудничеството и колективната работа в мрежа се разглежда като дейност, която се осъществява в защитената среда, само между група участници, които предварително се познават. За да може група хора успешно да използват такъв тип софтуер, е желателно те да са работили

предварително заедно и единственото нещо, което е необходимо за подобряване на работата е “online” среда за обмен на файлове, email-и и текстов или гласов chat.

Напоследък, при разработване на системи за сътрудничество и съвместна работа все повече се налага разбирането, че групата потребители трябва да бъде част от по-обширна среда, а не отделна единица, която няма взаимодействие с останалата част на света. Съвместният проект, върху който те работят, в допълнение към „частната” за групата среда, разполага и с „публична”среда, чрез която е възможно взаимодействие, предаване и публикуване на информация за проекта, както и разпространяване на резултатите извън групата на участниците в проекта. Осигуряват се и възможности за раздаване на права за редактиране и превеждане на потребителите или дори на цели подгрупи.

Като съществено ограничение при тази категория системи може да се изтъкне, че много често при тях не е възможно публикуване на информация, резултати или общуване извън групата членове на проекта. Общата среда е защитена и е невъзможен обмен на информация с масовата публика или със заинтересувани лица, които не са участници в проекта. Освен това, липсва архив с информацията за комуникациите, файловете или електронното съдържание, след разпускане на групата. Следователно групата няма възможност да продължи процеса на сътрудничество в бъдеще, без да възстанови всичко от самото начало. Би било добре, всеки участник да разполага с едни и същи възможности за обмен на данни (bandwidth), за гладкото протичане на комуникацията, както и повишаване на сигурността, защото при пропадане на връзката има опасност от загуба на всички файлове, данни за осъществената комуникация в групата и др. Други слабости са: не са предвидени средства за уведомяване за настъпила активност от някой член на групата; системата обикновено може да се ползва само от личния компютър, а не от произволен; достъпът за групата е невъзможен чрез използването на web-browser, клиентската софтуерна част е специфична за различните операционни системи, което от своя страна означава, че потребителят трябва да може сам да инсталира или осъвременява (update) програмата; възможни са проблеми с използването на продукта заради различни ограничения на защитните програми (firewalls) и др.

2.2.2. Съществуващи решения

В тази част са разгледани особеностите на някои от най-популярните системи за сътрудничество и съвместна работа.

2.2.2.1. Groove

Програмният продукт за колаборативна работа, Groove Virtual Office първоначално е разработен и поддържан от Groove Networks [26].

Най-новата версия, Microsoft Office Groove 2007 [16] се предлага на пазара вече като съвместна разработка на Groove Networks и Microsoft. Тя е усъвършенстван вариант на Groove Virtual Office и дава възможност за ефективна съвместна работа на различни колективи от хора, в рамките на общо работно пространство, независимо от това къде се намират, кога и с кого се налага да работят.

2.2.2.2. MS Netmeeting

MS NetMeeting [20] предлага цялостно решение за осъществяване на конферирane между много потребители чрез Интернет, на текстов chat, файлов трансфер, ползване на т.нар. „бяла дъска", както и на аудио и видео данни от точка до точка (вж. Фиг. 2.5).

Фигура 2.5. MS NetMeeting [20]

2.3. Софтуерни решения за управление на проекти

Основното предназначение на софтуерните решения за управление на проекти обхваща разпределяне на задачите между участниците в проекта, както и осигуряване на групово работно пространство и на неговите основни функции, които обикновено включват споделяне на файлове, общ календар и дискусионни форуми.

2.3.1. Функционалност, приложение, ограничения

В предлаганата от системите за управление на проекти обща работна среда, всеки от екипа участници в проекта има възможност да представи отчета за изпълнение на своята работа и финансов отчет („progress and cost reports"),

на базата на които средата автоматично генерира обобщен отчет за състоянието на дейностите по проекта. По този начин, координаторът/ ръководителят разполага с актуална информация за статуса на проекта към даден момент от времето. В допълнение към работната среда с частен достъп, само за членовете на колектива, проектът разполага и със среда с публичен достъп, чрез която да взаимодейства, предава и публикува информация за проекта, както и да разпространява неговите резултати за външни лица. Използването на система за управление на проекти, обаче може да доведе до губене на ценно време, както за ръководителя на проекта, така и за участниците в него, ако се пренебрегне факта, че някои от членовете е възможно да участват в няколко проекта едновременно, че проектът работи във взаимодействие със своята работна среда, а не независимо от нея и че голяма част от повтарящите се задачи по управление на проекта трябва да се автоматизират, колкото е възможно в по-голяма степен. Не бива да се забравя, например, че даден участник (ръководителят или друг член) може да ползва една и съща идентификация като потребител на системата, за всички проекти, в които е ангажиран и така, да си осигури възможност, лесно да се справи със задълженията си по тях. Той ще може да използва повторно съдържанието, разработено по един проект за нуждите на други бъдещи проекти, да спести пари, като плати лиценз само за един потребител, а всъщност – да участва в множество проекти.

Този тип софтуерни решения се развиват по посока към осигуряване на цялостни услуги за интерактивно разпространение на бюджета и отчетите по проекта, както и за планиране, управление на проекта, което спестява време и автоматизира задачите. Стремежът е да се развият услуги за ежедневно известяване и напомняне по въпроси свързани с проектните дейности, както и да се даде възможност, на посочен URL (Universal Resource Locator) по избор, колаборативната среда да е достъпна и за нейни членове, и за външни лица.

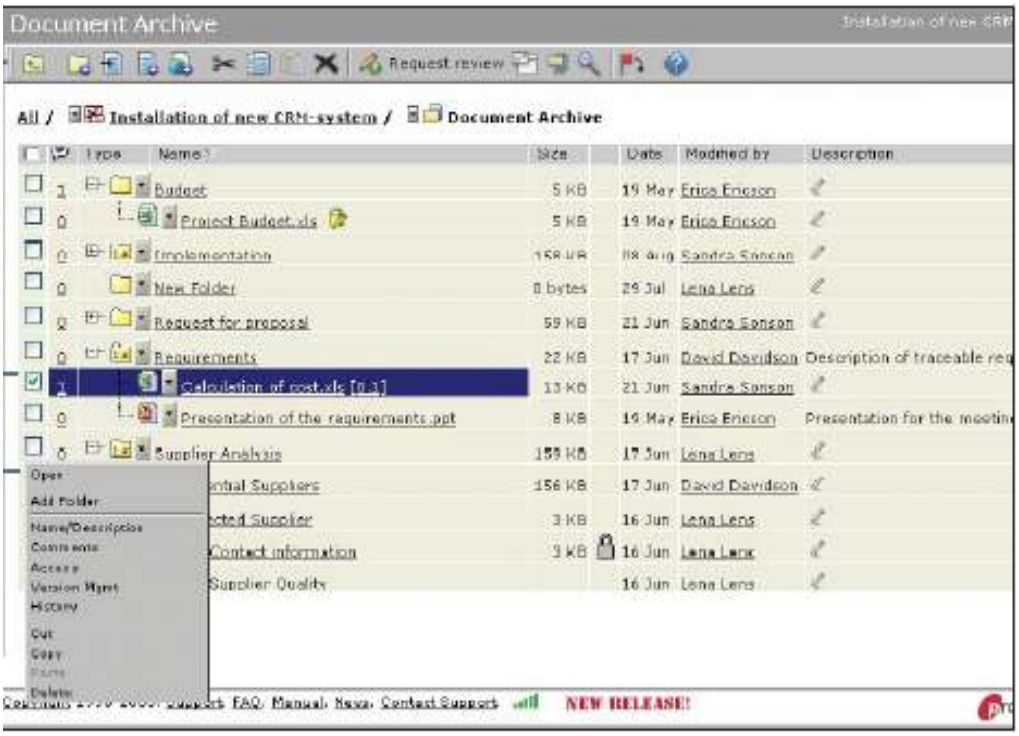
Често срещани ограничения при системите за управление на проекти са: невъзможност за комуникация и публикуване на информация и резултати от проекта извън неговия колектив; оскъден набор от средства за публикуване на съдържание (обикновено се предлага средство за взаимодействие, подобно на форум с линеен обмен на информация); прости инструменти за разпределяне на задачите, без отчитане или напомняне на крайния им срок и др.

2.3.2. Съществуващи решения

В настоящия раздел са разгледани особеностите на някои известни системи за управление на проекти.

2.3.2.1. ProjectPlace

Projectplace [12] е web-базиран софтуер за управление на проекти, който е достъпен като абонаментна услуга, поддържана от специален шведски доставчик (Projectplace International AB). Услугата е стартирана на адрес www.projectplace.com през септември 1998, като една от първите платени web-базирани услуги.



Фигура 2.6. Projectplace: архивиране на документи [34]

Projectplace представлява комплект от програми, осигуряващи инструменти за управление на проекти по подобие на Microsoft Project. Projectplace модулите включват: архивиране на документи (вж. Фиг. 2.6), управление на тиражи, планиране и проследяване на проекти, управление на срещи, портали на проекти, форум за контакти и взимане на решения.

Услугата се предлага на шест езика (шведски, норвежки, холандски, английски, немски и френски).

2.3.2.2. ERoom

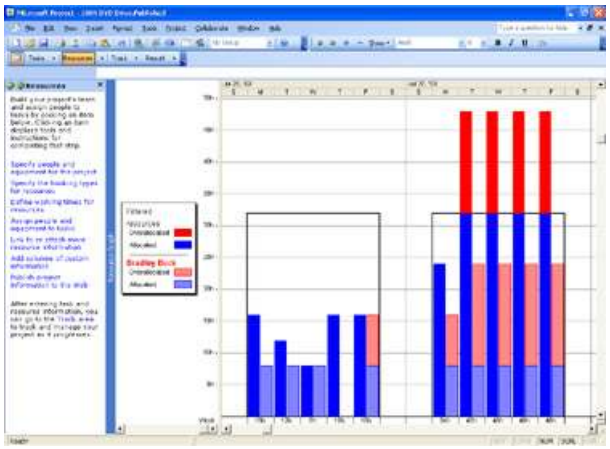
EMC eRoom [5] осигурява web-базирано работно пространство, което може лесно да се разгръща и настройва и е удобно за използване от разпределени колективи за съвместна работа. С негова помощ, всеки работещ по определен проект, колектив от хора, които може да се намират на различни краища на света, би могъл да увеличи и подобри производството и доставката на продуктите или услугите си, да оптимизира колаборативния бизнес процес, да разшири иновациите, да рационализира вземането на решения, лесно да администрира отчитането на проекта и да ползва удобни средства за форматиране и отпечатване.

2.3.2.3. QuickPlace

QuickPlace [10] е софтуер, осигуряващ web-базирано споделено работно пространство за сътрудничество в реално време между географски разпръснати участници. Използвайки QuickPlace, колеги или бизнес партньори могат да комуникират без забавяне, онлайн, в рамките на структурирано работно пространство, специално създадено за целта. Това дава възможност за хората да работят заедно много по-лесно и евтино, а в някои случаи, дори прави възможно сътрудничество, което без него не би била възможна. QuickPlace е достъпна на четиринадест езика и на пет различни платформи.

2.3.2.4. MS Project

Microsoft Project [18] е програма за управление на проекти, разработена и продавана от корпорацията Microsoft.



Фигура 2.7. Microsoft Project: проследяване на проекта [18]

Microsoft Project е създаден, в помощ на ръководители на проекти или ръководни лица от бизнеса и планирането за улесняване разработката на планове, разпределянето на ресурсите по задачите, проследяването на развитието и изпълнението на проекта (вж. Фиг. 2.7), анализа на обема на работа и разпространяването на данните по проекта. По този начин може да се осъществява поголям контрол върху изпълнението на проекта, да се централизират ресурсите, да се автоматизират процесите и да се подобри взаимодействието на колектива, работещ по проекта. Microsoft Project създава разписания на критичните пътища, които могат да бъдат съставяни, като се отчитат използваните ресурси.

2.4. Софтуерни решения за представяне

В класическия смисъл, системите за представяне главно са предназначени за презентиране с помощта на слайдове (slides), за множество поканени участници, в реално време, придружено от гласа на презентиращия.

2.4.1. Функционалност, приложение, ограничения

Системите за представяне поддържат обикновено само синхронна комуникация, т.е. комуникация в реално време, която не позволява да се направи справка или връщане назад, след като презентацията е свършила. Причина за това е, че осъществяването на представяне с потоково предаване на звук и видео в реално време до десетки или стотици участници изисква изключителна качество на връзката, както и стабилен сървър.

Възможностите на съвременните софтуерни среди за презентация позволяват на участниците достъп до конкретно представяне, както в реално време, така и във всеки един момент в бъдещето. Средствата за представяне, вече се съчетават с други синхронни (моментални съобщения, chat и др.) и асинхронни (форум, email, споделяне на файлове и др.) комуникационни услуги, което позволява да се съхраняват данни за обмена на мнения между участниците по време на презентацията. Тези данни са достъпно за бъдещи справки и преглед, дори от други хора. Някои системи поддържат работа с представяния в различни формати, като участници, които не могат да разглеждат такъв формат, могат да прочетат информация за него и да изтеглят (download) необходимата за визуализирането му програма. Понякога дори има средства за преобразуване от формата на MS PowerPoint в HTML (HyperText Markup Language) [31] формат.

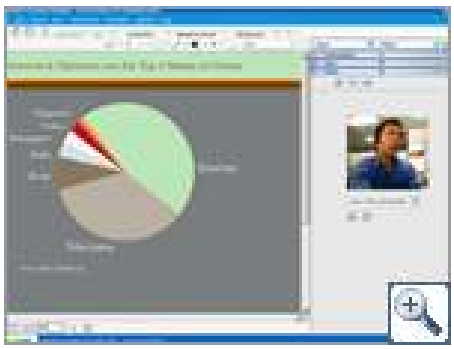
Типични ограничения по отношение на този род системи са: липса на поддръжка за мрежово взимодействие, както и за съвместно публикуване на съдържание; средата не разпознава участниците като група и не им предоставя възможност да общуват един с друг, след края на представянето; често презентациите са под формата на покани за участие, изискващи от участниците да се сдобият с парола; презентиращият онлайн е необходимо едновременно да изпълнява поне две роли на лица, добре запознати с темата на презентацията – едната е на представящия, а другата е на този, който ръководи дискусията между участниците и отговаря на техните въпросите; презентацията обикновено е достъпна само за ограничено време, след като говорещият приключи; дискусията в реално време, както и въпросите от участниците рядко се записват.

2.4.2. Съществуващи решения

Две от най-популярните системи за представяне са Webex и Placeware.

2.4.2.1. Webex

WebEx Communications Inc. [28], е компания, която предлага услуги за онлайн срещи, web-конферииране, и видео-конферииране с цел представяне на информация или обмен на мнения и обсъждане между участниците (вж. Фиг. 2.8). Нейните продукти включват "Meeting Center", "Training Center", "Event Center", "Support Center", "Sales Center" и др.



Фигура 2.8. WebEx услуги [28]

2.4.2.2. Placeware

PlaceWare е мултимедийна платформа за web-базирана комуникация и сътрудничество. Клиентите от бизнеса и промишлеността предпочитат PlaceWare web-конферирането поради неговата надеждност, гъвкавост, повишена сигурност и не на последно място, възможността за достъп чрез web-

browser. Системата поддържа всички типове web-базирано сътрудничество, от мащабни срещи с хиляди участници, до малки работни срещи, представяния и сесии за е-обучение. Основана през 1996, компанията PlaceWare сега е филиал на корпорацията Microsoft и новото наименование на продукта им е Live Meeting [17] (вж. Фиг. 2.9).

Фигура 2.9. Live Meeting [17]

2.5. Софтуерни решения за проучване и анкета

Програмните средства за проучване и анкета се съсредоточават върху възможността за създаване на въпросник или анкетна форма, която да се разпространява сред определена целева група хора, като след попълване от тяхна страна, могат да се видят подробно техните отговори, както и обобщение на резултатите.

2.5.1. Функционалност, приложение, ограничения

По-старите софтуерни решения от този тип, най-често имат постоянна база от въпроси, не пазят отговорите, дадени преди това от участника, лишени са от средства за комуникация, сътрудничество и работа в мрежа. При съвременните системи за проучване, обаче се поддържат дори по няколко начина за визуализация на въпросите, подробен или обобщен преглед на отговорите, както

и възможност за многократно попълване на анкетата от даден участник, като е допустимо да даде различни отговори всеки път, в зависимост от аспекта, в който отговаря, а дори и в един и същи аспект (което е особено полезно за проверка дали запитвания е променил гледната си точка). Те поддържат голям набор от типове въпроси, като например, със свободен отговор, с множествен избор, попълване на шаблон (матричен), с фиксиран текстов или числов отговор и др.

Този тип системи имат едно вторично положително качество: проучванията често се прилагат върху целева група с общи интереси. Съществува голяма вероятност членовете на тази група да проявят интерес да общуват помежду си. По този начин, участниците са още по-склонни да отговарят на проучването, за да се запознаят с хора с подобни интереси.

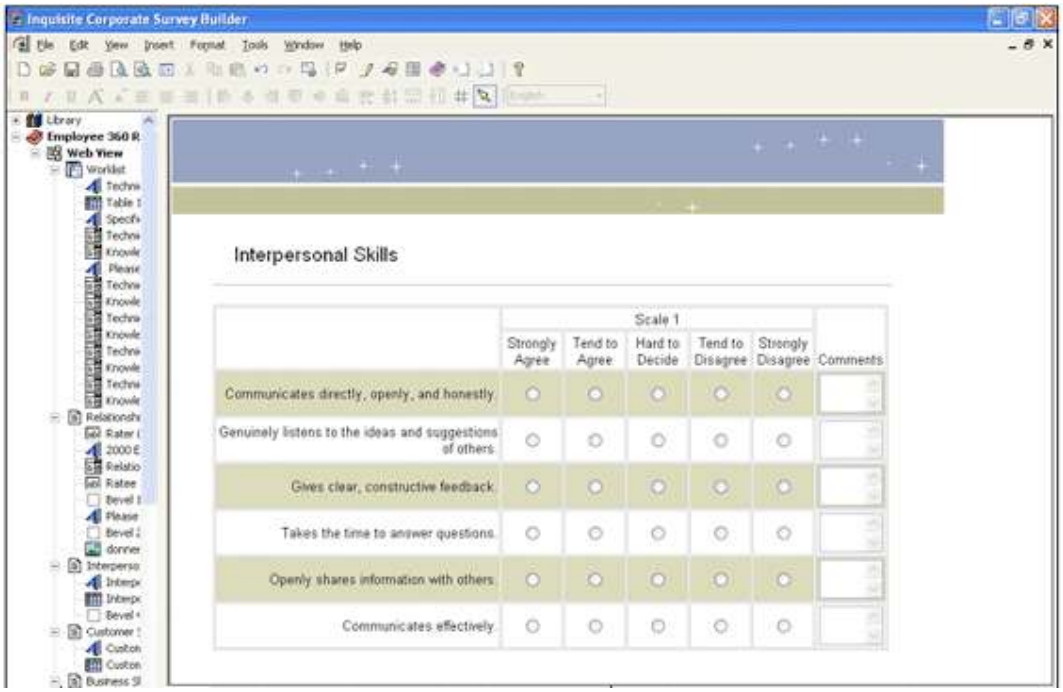
Характерна слабост при системи за провеждането на анкети е, че не всички от тях допускат формулирането на въпроси, изискващи свободен или шаблонен отговор. Те не поддържат автоматично и в реално време адаптиране на проучването или предлагане на подходяща помощ, в зависимост от отговорите (възможно и в предишни проучвания) и изискванията на анкетирания. Рядко е възможно да се продължи попълването на анкета, започната в минал момент от времето. Сериозно ограничение е и невъзможността за взаимодействие и комуникация между анкетираните.

2.5.2. Съществуващи решения

Характеристиките на някои от най-популярните системи за проучване и анкета са разгледани по-долу.

2.5.2.1. Boomerang

Boomerang [25] е системата за работа с анкети и въпросници. Boomerang е web-базирана система за разработване на email анкети и въпросници, които могат да се създават онлайн и да се редактират в реално време в много различни формати. Така разработените въпросници могат да се използват в разнообразни сфери, например като проучвания на пазара, състояние на продажбите, анкетиране на персонала, оценка на обучението, оценка на персонала, проучване на клиентите или доставчиците и много други приложения в критични бизнес области.



Фигура 2.10. Системата Inquisite [11]

2.5.2.2. Inquisite

Inquisite [11] е лесна за употреба софтуерна система за провеждане на web-базирани проучвания, която предлага за потребителите-неспециалисти възможност лесно да управляват свои web-базирани проучвания и да контролират цялостния процес на тяхното провеждане. С Inquisite е възможно за проучването да се избере специфична целева група, да се включат отдалечени работници или клиенти в реално време и моментално да се визуализира неговия резултат в подходяща форма и възможно на много нива. Много по-лесни за попълване от хартиените проучвания и по-малко скучни от телефонните анкети, мощните възможности на Inquisite позволяват създаването и провеждането на проучвания с много високо качество (вж. Фиг. 2.10).

3. Анализ на проблема. Проектиране на решението

Конкретната проблемна област на настоящата дипломна работа е областта на планиране на проекти и управление на проекти в сферата на висшето образование и научните изследвания, които се разработват на колективна основа и на принципите на самоорганизирането и самоуправлението (вж. Гл. 1).

В тази глава се прави анализ на потребностите [1, 9] по отношение на подходящ софтуер, с приложение в по-горе дефинираната проблемна област и се формулират идейният проект, както и случаите на употреба на конкретната програмна система, която се предлага като решение на поставените в Гл1. задачи.

3.1. Анализ на изискванията

Целта на анализа е теоретична обосновка на предлаганото решение, т.е. да бъдат определени изискванията по отношение на проектирането и разработването на web-базирана многоезикова платформа за оперативно управление на проекти, така че тя да предлага достатъчно високо ниво на функционалност, надеждност и взаимосвързаност на компонентите.

Проучването на тази относително нова проблемна област представлява значително предизвикателство. За уточняване на изискванията към разработката настоящият анализ се базира на систематизация на характеристиките на разглежданите проекти (вж. Гл. 1), както и на основните групи участници в тях (наречени тук, в най-общ смисъл, потребители).

3.1.1. Характеристики на проектите, финансирани по договори

Проектите, които са от интерес за дипломната работа са такива, които са финансирани по договори и са разработвани от не особено голям колектив.

Разглежданият тип проекти има голям брой отличителни характеристики [14, 22] в сравнение с други типове проекти, които биха могли да се разработват, например в областта на производството на материални, софтуерни и др. продукти, доставките, услугите, обучението и много други бизнес области. Техните особености засягат доста различни аспекти на организацията на проекта [15, 23], включително партньори, формална страна, управление, комуникация.

Хората от колектива, работещи по проекти, финансирани по договори, не са просто участници в проекта. По точно е те да бъдат наречени партньори, работещи по общи задачи и споделящи обща отговорност. Типични характеристики на **партньорите по проекта** са:

- сравнително голям брой;
- от различни държави;от разнородни организации (висши учебни заведения, изследователски институции, фирми, неправителствени организации);
- с интереси в различни области (научни, приложни, образователни);
- с разнороден опит и цели и др.

По отношение на **формалната страна** на проектите, които се разработват на колективна основа и на принципите на самоорганизирането и самоуправлението, те се характеризира със строго определени изисквания, фиксирани от договор с финансиращата организация, договор между партньорите, и др. Изискванията засягат:

- срокове;
- продължителност;
- изпълнение;
- отчитане.

Управлението на проектите също се подчинява на строго определена специфика:

- при управление на дейностите партньорите са равноправни и относително независими;
- решенията се взимат след обсъждане;
- има един партньор с по-специална функция – координатор на проекта (организира координацията на съвместните дейности);
- финансовото управление не е централизирано – всеки партньор отговаря за разпределението и изразходването на своята част от бюджета на проекта.

Особено сериозни са изискванията към **средствата за комуникация** между партньорите и отделните участници. В този случай задължително се изисква постоянна комуникация и обмен на информация със средства разнообразни по**:**

- тип (синхронни и асинхронни);

- технологична база (web-базирани, електронни, мобилни) и
- предназначение (информационни, сътрудничество, дискусии, конференции).

3.1.2. Характеристики на потребителите

Основната група потребители за проекти, финансирани по договори, са колективи, които са сформирани за осъществяване на определен проект и за които е характерно [33], че *колективът* е:

- по-скоро виртуален;
- самоорганизиращ се;
- самоуправляващ се;

На членовете на колектива, сформиран за съвместна работа, може да бъде приписана специална *роля*, като например:

- координатор на целия проект;
- локален координатор по проект от страна на някоя от партньорските организации;
- ръководител на работна група, сформирана за изпълнение на конкретна задача от работния план на проекта;
- контактно лице от страна на партньор по проекта;
- член на работна група и т.н.

Много често членовете, работещи по даден проект могат също да участват и в други проекти или да координират повече от един проект.

Допълнителна група потребители са лица**,** организации, специалисти, групи, за които са предназначени резултатите от проектите или имат друг общ интерес към тях (търсят партньори, интересуват се от информацията, използване на продуктите и т.н.).

3.1.3. Общи изисквания към платформата

Тази част от дипломната работа е посветена на уточняване на изискванията към разработката, които могат да бъдат изведени на базата на по-горе извършения анализ.

Изискванията, на които трябва да отговаря целевата web-базирана многоезикова платформа за оперативно управление на проекти могат да бъдат сис-

тематизирани, според следните съществени за нейните потребители измерения [24]: функционалност, интерфейс, достъпност, навременност, администриране по отношение на потребителите, сигурност и отвореност.

Най-общо, **функционалността** на системата е необходимо да предлага богати възможности и услуги, но от гледна точка на основната характеристика на съвременните проекти – възможност за колективна работа, комуникация и взимане на решения.

По отношение на **интерфейса**, разбира се, се поставят стандартните в това отношение изисквания, но в случая е необходимо да се постави сериозен акцент върху ползването на системата и на поддръжката, която тя осигурява за своите потребители, а не толкова върху функционирането като цяло. Това означава, че интерфейсът трябва да бъде:

- опростен;
- адаптивен;
- интуитивен;
- близък до този на други познати средства, т.е. стандартизиран в определена степен;
- да осигурява лесно се работи с приложението (лесен за използване).

Особено важно изискване към разработката се налага във връзка с нейната **достъпност.** Разработваната система освен, че трябва да е достъпна *в класическия смисъл*, т.е. по всяко време, за всички потребители и т.н., тя трябва да осигурява и достъп, с помощта на *разнообразен интерфейс*, като например, за синхронен и асинхронен Интернет достъп, чрез мобилни устройства и др. Интерфейсът трябва в максимална степен да улеснява използването на наличната информация и колективно съдържание.

Изискването за **навременност** означава предоставяне за потребителите на всяка необходима и важна информация в реално време. Това включва най-вече на базата на различни електронни и мобилни технологии да се осигурят богати средства за:

- уведомяване;
- напомняне;
- цялостно и структурирано представяне на цялата налична релевантна информация.

Възможностите на системата за **администриране по отношение на потребителите** си трябва да осигурява голяма степен на свобода. Сформирането на колективи трябва да бъде почти напълно нерегулирано, добавянето на нови потребители да бъде напълно безпроблемно и на основата на свободен избор за използваните възможности и услуги.

Сигурността налага изисквания с цел защита на данните и ресурсите на отделните потребители, както и на общото колективно съдържание и информация. Това означава, че разработваната система трябва да предлага и за установяване на автентичността и (autentication) на потребителя, и за разпределяне на права (authorization). Средствата за установяване на автентичността е процес при който приложението разпознава идентичността на потребителя и нивото му на достъп, а тези за разпределянето на правата разпознават потребителя по отношение на правата му за ползване на едни или други ресурси на програмата.

Изискването за **отвореност** е насочено главна към подхода при проектиране на системата и е свързано с възможностите за нейната лесна поддръжка и адаптивност според изискванията на потребителя. В тази връзка, системата за управлвние на проекти е добре да предлага и съвместимост с други подходящи софтуерни средства.

3.2. Проект на решението

На основата на направения в предишния раздел анализ на потребностите по отношение на подходящ софтуер, с приложение при планиране и управление на проекти, тук се предлага идеен проект на конкретна програмна система, наречена W-EPROM (Web-Environment for PROject Management).

Системата W-EPROM е web-базирана и предоставя възможности, както за планиране и реализация на проекти, с отчитане на зависимостите между времевите, човешките и информационните ресурси, така и за представяне и отчитане на проектите, чрез разнообразни средства за комуникация, организация, публикуване и споделяне на различни типове информация. Тя може да бъде и удобно средство за съвместна работа и обмен на данни посредством Интернет за разнообразни колективи от хора с общи интереси.

3.2.1. Основни понятия

В терминологията на W-EPROM, различните колективи, техните членове, предлаганите услуги и средства, както и информационните ресурси, които се обменят, се наричат **обекти**. Всеки обект има определени характеристики, част от които са отличителни по отношение на неговата физическа същност (напр. име, размер и др.), а останалите – по отношение на действията, които могат да се извършват с него (напр. визуализация, изтриване и др.) по указание на потребителя или системата. Тези два вида характеристики се наричат съответно **свойства** и **операции/ функции**.

3.2.1.1. Видове обекти

Според своята **функционалност** обектите се делят на два основни вида – активни и пасивни. *Активните обекти* са такива, които могат да въздействат върху други (да ги създават, да променят някои техни характеристики и т.н.), за разлика от *пасивните обекти*. Начините, по които активните обекти могат да въздействат върху други, се наричат техни *функции***.**

От гледна точка на **структурата**, както активните, така и пасивните обекти могат да бъдат *контейнери* и *прости*, според това могат ли да съдържат други обекти или не. Контейнерите в общия случай включват, както прости, така и други контейнерни обекти. Примери за прости обекти от системата са файл (документ) и потребител, а на съставни – папка и потребителска група.

В **семантично** отношение обектите на W-EPROM могат да се отнесат към следните видове (от тях само обектите ползватели са активни, а всички останали са пасивни):

- *Ползватели* – това са всички активни обекти, които се явяват крайни потребители на системата;
- *Информационни* – осигуряват поддръжката и обмена на информационни ресурси за нуждите на ползвателите;
- *Поддържащи* – подпомагат обектите ползватели При палниране, управление, взимане на решения или ползване на услугите на системата;
- *Комуникационни* – предоставят различни комуникационни услуги.

В Таблица 4.1 са систематизирани всички обекти на W-EPROM , според терминологията на системата и вида, на който принадлежат.

		Прости	Контейнери
Ак-тив-	*Ползва-тели*	гост	потребителска група
		потребител	група по интереси
			проект
Пасивни	*Информа-ционни*	документ	папка
	Под-държащи	контакт	бележник с контакти
		паметка	календар
			списък паметки
		профил	
		задача	работен план на проект
			списък задачи
			списък членове на група
			списък проекти
			списък групи по интереси
	Комуника-ционни	съобщение	входяща кутия
			изходяща кутия

Таблица 3.1. Основни обекти на W-EPROM според техния вид

Освен това обектите са взаимосвързани помежду си, като тези **връзки** могат да бъдат от някой от следните:

- "*част-от*" – някои от обектите могат да съществуват само като част от друг съставен обект (напр. контактът е винаги част от бележник с контакти). Има и случаи, в които тази връзка е възможна, но не задължителна (напр. потребителят може да е член, т.е. част от група по интереси, но не винаги); Връзката се пази в обекта- контейнер.
- "*отнесен-към*" – означава, че обектът е съпоставен на друг конкретен обект, като описва някои негови характеристики, пояснява го и др. и по този начин улеснява неговата поддръжка и управление (напр. профилът отразява различни характеристики на проявление, на обекта, с който е свързан;
- "*принадлежи-на*" – в смисъл, че обектът-**собственост** има подчинена роля по отношение на друг обект, наречен **собственик** (напр. потребителят може да има собствени документи, календар и т.н.);
- "*споделен-с*" – някои обекти могат да бъдат споделяни за общо ползване с други (напр. папка може да е споделена между всички потребители). Връзката се пази в обекта, с който се споделя.

Обектът, с който се споделят общи ресурси получава т. нар. **право на достъп** до споделения обект. Правата за достъп могат да бъдат например: за четене, запис, изтриване, преместване, модификация и др.

Могат да бъдат **споделени** само следните пасивни обекти: документ, папка. А споделянето може да бъде направено от активен обект, който или е собственик на пасивния, или поне има такова право на достъп до него, което позволява споделяне.

Ресурси, които не са споделени се наричат **частни**, а такива, до които е позволен напълно свободен достъп се наричат **публични**.

3.2.1.2. Работен прозорец и работно пространство

Всеки обект от системата представлява логическа цялост от присъщите му характеристики, функции, операции и взаимосвързаните по определен начин с него обекти.

С цел осигуряване на унифицирано и единно представяне на обектите, както и възможност за тяхното управление, на всеки обект от системата се съпоставят едно или няколко визуални представяния, чрез които те участват в различните видове интерфейс, осигуряващи достъпа до функциите и услугите на W-EPROM . Някои от тези представяния служат за визуализация на обекта в т. нар. ”разгънато” състояние. Те заемат правоъгълна област от екрана, която обединява, както визуализация на съдържанието или подобектите на обекта, така и средства за достъп до свързаните с тях функции, услуги или операции. Според възприетата терминология в подобни софтуерни системи, всяко такова представяне, съответстващо на активен обект ще наричаме **работно пространство**, а на пасивен обект – **работен прозорец**. Освен в ”разгънато” състояние обектът може да се изобразява и в “минимизирано” състояние, за нуждите на което му се съпоставя и **икона**.

3.2.1.3. Същност и характеристики на основните обекти

Всеки обект има характеризиращи свойства, или т. нар. **Метаданни** като: Име, Описание, Подсказка, Вид, Икона, Собственик, Дати на създаване и последно редактиране и др.

Освен това, обектът включва свойства **за неговите съдържателни данни**, които най-общо казано определят неговата стойност (или състояние). Например, съдържанието на съобщението е някакъв текст.

Към съдържателните данни на обект се отнасят още свойства, които отразяват неговите взаимовръзки с други обекти (част-от, характеристика-на, принадлежи-на, споделен-с). Или още, **всеки обект-контейнер** съдържа списък с обектите, които са "част-от" него. а **обект, с право на достъп до** споделени общи ресурси също разполага със списък на обектите "споделени-с" него заедно със съответните му "права за достъп" до тях.

3.2.1.3.1. Пасивни обекти

Документ в смисъл на W-EPROM е всеки файл, който участва като ресурс в системата (обикновено е зареден от някой потребител). Документът може да съдържа различен тип данни (текстови, графични, звукови, видео и т.н.) в подходящ файлов формат. Обектите-документи имат уникалната характеристика *MIME тип*, която определя файловия им формат.

Папката е съставен обект, чийто смисъл е подобен на този възприет в ГПИ (GUI). В този смисъл тя е средство за организация и управление на всички информационни обекти, които съдържа. Папката може да включва документи, други папки

Контакт съдържа контактна информация за хора, които не е задължително да са потребители на W-EPROM.

Бележникът с контакти съдържа контактите на даден потребител.

Пametката е обект, който винаги е част-от календар и позволява в календара да се включи напомняне за някакво планирано задължение, като частна среща, работна среща, конференция, лекция и др. За тази цел се указва време, дата, статус на активност, предмет и описание.

Списъкът с пametки съдържа всички пametки за даден приод от време и има възможност за персонално или споделено ползване и сортиране на елементите му по даден критерий.

Календарът подпомага планирането на времето, за различни периоди от време (година, месец, седмица, ден), чрез включване на пametки и възможност

за персонално или споделено ползване. Различното при него е интуитивното му графично представяне по месеци.

Профилът служи главно за задаване на детайлни данни за даден обект.

Задачата е винаги част от работен на проект. Освен това тя се характеризира с формулировка и собственик, състояние на възложеност, евентуално кой е изпълнителят й, какъв е срокът й, статус(не започната, в процес, завършена, чакаща). Всички задачи, на които индивидуален ползвател е собственик или изпълнител се включват в неговия **списък задачи** (специален вид папка).

Работен план е обект, който е винаги “отнесен-към” конкретен проект и включва всички задачи, дефинирани в него. Той представлява специална папка.

Списъкът от членове на група е папка, която съдържа списък от активни обекти, с или без URL-връзки към тях и поддържа функции за работа с този списък. Тези активни обекти са членове на ползвател-контейнер.

Списъкът от проекти – папка, която съдържа списък от проекти в системата, с или без URL-връзки към тях и поддържа функции за работа с този списък.

Списъкът от групи по интереси – папка, която съдържа списък от групи по интереси в системата, с или без URL-връзки към тях и поддържа функции за работа с този списък.

Съобщението служи главно за комуникация между членовете на системата. То може да бъде както текстово, така и системно (може да бъде изпращане/ приемане/ отказване на покана за присъединяване към проект или група по интереси или на заявка за възлагане на задача.

Входящата кутия съдържа всички получени текстови или системни съобщения и поддържа функции за работа с тях

Изходящата кутия съдържа всички изпратени текстови или системни съобщения и поддържа функции за работа с тях

3.2.1.3.2. Активни обекти

Едни от най важните характеристики на активните обекти са роля и абонамент:

Ролята е средство за определяне на "**правата за достъп**" на активния обект до различни други обекти, т.е. е множество от позволените му действия. Ако активният обект има няколко роли спрямо другия, то това множество е обединение от позволените действия за всяка от тях. Основните правила, които определят как ролите се интерпретират в системата са:

- всяка роля се свързва с определена папка (обикновена или от специален вид) и важи за всички обекти от папката;
- макар че приписване на роля може да става едновременно за цяла група потребители (напр. на всички членове на споделена папка), то в крайна сметка, ролята се свързва с отделен потребител;
- ако папката е частна, то единствено на обекта-собственик може да бъде приписана роля, а в случай, че е споделена, то роли спрямо нея (възможно различни) имат всички обекти, с които тя е споделена (вж. 3.1.1);
- приписването на роля става или автоматично още при регистриране на потребителя в системата (вж. 3.2.1), или се дефинира от собственика на споделена папка задължително при привличане на нови нейни членове, или може да се направи по желание (но от потребител, който има такова право);
- правата за достъп определени от ролята (или ролите), която някой потребител има спрямо дадена папка се наследяват (и важат) и за всички новосъздадени подпапки, ако и докато на потребителя не се припише нова роля;
- има предварително дефинирани роли (като регистриран потребител, координатор на проект, член на проект и др.), но потребителите на системата могат да дефинират и свои;
- има една привилигирована роля, която включва всички права за достъп до всички обекти – "администратор на системата".

Абонаментът служи за определяне на това, кои от незадължителните услуги, средства и функции на W-EPROM , ползвателят желае да бъдат включени към неговото работно пространство. Множеството от незадължителни възможности е различно за отделните видове активни обекти, като за всеки от тях има

предварително дефинирани нива на абонамент (стандартен, разширен и пълен). Правила, които определят как абонаментите се интерпретират в системата са много сходни с тези, които важат за ролите.

Услуги, средства и функции на системата, които могат да участват в абонамента на индивидуален или групов ползвател са (някои от поддържаните като подфункции на изброените също могат да са предмет на абонамент, в този случай абонамента се извършва на нива):

- функции: свободно/регулирано включване на членове (за ползватели-контейнери), свободно споделяне на ресурси, достъп до групите по интереси, поддръжка на история (за събитията с пасивен обект), оценяване и др.;
- средства: лично работно пространство, календар, бележник с контакти, работен план, списък задачи и др.;
- услуги: всички услуги за уведомяване, оповестяване и напомняне и комуникация (вж. 3.2.3.2) и др.

Гост е всеки нерегистриран в системата индивидуален ползвател. Гостът може да ползва само публичните ресурси на системата чрез общодостъпното работно пространство. Той може да разглежда поддържащите обекти контейнери - списъците с потребители, проекти и групи по интереси на системата.

Потребител на W-EPROM е всеки регистриран (вж. 3.2.3.1) в системата индивидуален ползвател. W-EPROM потребителят се характеризира със задължителна информация за *идентификация***,** допълнителна *персонална информация***,** *роля* и *абонамент*. Потребителят е автоматично абониран за ползване на лично работно пространство и на него се съпоставя *основна папка* на потребителя.

Потребителска група е съвкупност от потребители на обикновена или от специален вид папка, която се формира или на базата на свободно, или на регулирано (по покана на собственика на папката или друг с такова право) включване на *членове*. Това позволява ползването на общо работно пространство от членовете на групата (съвместни ресурси, услуги и др.) и групово абониране или назначаване на роли. Папката, около която се формира групата се нарича *основна папка* на работното пространство на групата. Групата може да има

подгрупи. Всички потребители на W-EPROM също формират една потребителска група със свободно включване на членове.

Група по интереси е специален вид потребителска група, която допълнително се характеризира с *тематика*, от която се интересува групата. Тя може да е свободна за включване на нови *членове* или да изисква предварителното одобрение от менажера на групата (полурегулирана). Групата по интереси се различава от останалите ползватели-контейнери по това, че неговата основна папка е видима за всички, но само регистрирани потребители имат достъп до нейното съдържание.

Проект е вид потребителска група, чиито членове съставляват колектива за осъществяване на определен проект. Поради това основната папка на проекта подпомага някои специфични за такива колективи дейности, като представяне на проекта (визитка, заглавие, основна цел, цели) или събиране и разпространение на получените резултати. Най-съществена, обаче е възможността за времево планиране и управление на проекта (чрез работен план, задачи и задаване на времеви условия и зависимости между тях).

3.2.2. Интерфейс

W-EPROM предлага един Web-базиран интерфейс. **Web-порталът** поддържа пълната функционалност на W-EPROM, разгледана по-долу.

Основно понятие при Web-интерфейса е работното пространство. При стартиране на Web-портала на W-EPROM за клиента се визуализира общодостъпното работно пространство на системата, което по подразбиране е свързано със свободната потребителска група (отворена за всички клиенти) и съответна основна папка. Ако клиентът е регистриран в системата потребител и осъществи оторизиран достъп, то той получава възможност да работи в своето специализирано работно.

Работното пространство винаги има сходна обща структура и функционалност, осигуряваща удобен достъп до функциите, услугите на системата и поддържаните информационни ресурси (както собствените, така и споделени обекти. Неговата Web-страницата винаги включва заглавна част (лого, главно меню на системата, навигационна лента, лента с бутони за бърз достъп и др.) и

съдържателна част. Съдържателната част изобразява съдържанието на текущо избран пасивен обект, т.е. неговия работен прозорец.

Работният прозорец, от своя страна, също включва две части – заглавна (меню, лента с бутони за бърз достъп и др.) и съдържателна. **Работният прозорец** на съставните пасивни обекти има таблична структура. Съдържателната им част се състои от поле за навигация и списък с компоненти, който е обогатен с информационни и структурни икони и икони за индикация. Работата с обекта-контейнер е улеснена, чрез бутони за достъп и операции и др.

3.2.3. Функционалност

Функционалността на системата е изградена на базата на съвременните Интернет технологии.

3.2.3.1. Обща функционалност

Попълването на W-EPROM с разнообразни ресурси и установяването на отношенията между тях се поддържа от единни средства за **създаване и инициализиране** на обекти.

Създаването на активен обект става в следствие на лична регистрация в системата.

Пасивен обект може да бъде включен (създаден, изтеглен, зареден, въведен) в средата автоматично, едновременно със създаването на друг обект или от някой активен обект, който вече е част от нея. Например, при регистриране на нов потребител се създава съответно работно пространство.

В някои случаи (най-често при активни обекти) при включване на нов обект се изисква преминаване през допълнителна процедура. Например, при покана на нов член на група се изпраща съобщение до него и само,ако той приеме да бъде член на групата, се добавя към колектива.

За всеки обект се поддържа съответен обект **профил**, който съдържа цялата информация за идентификацията, настройките, евентуално ролята, абонамента и др. за този обект. При създаване на нов обект, по-голямата част от данните в неговия профил получават стойности по подразбиране. Но те винаги могат да бъдат редактирани по-късно, един от начините, за което е достъп до профила.

W-EPROM поддържа и съответни единни средства за **изключване (унищожаване) на обекти** с цел съхраняване на нейната цялостност и интегрираност между ресурсите.

3.2.3.2. Функционалност по отношение на индивидуалните и групови ползватели

Активните обекти в W-EPROM са именно тези, благодарение на които и за чиито нужди, се поддържа развитието на нейните ресурси. Ето защо, функционалността на системата се разглежда именно по отношение на индивидуалните и групови ползватели. W-EPROM осигурява спектър от функции и услуги, които в общия случай са елементи на различни по тип платформи, като например платформи за взаимодействие и комуникация, за колективна работа, за публикуване на информация и др. W-EPROM подпомага работата на индивидуалните и групови ползватели (при това според степента на тяхната организираност) в следните основни направления:

1. Свобода при самоорганизиране на колектива
2. Поддръжка на информационните ресурси (Представяне на съдържанието)
3. Средства за комуникация и обсъждане
4. Сътрудничество и съвместна работа
5. Планиране и управление
6. Осигуряване на качеството

Функционалността на W-EPROM в изброените направления (указани с номера) за различните категории ползватели са систематизирани в Таблица 4.2, като са посочени обектите, операциите и средствата, чрез които се осъществява.

Направления	Функции	W-EPROM обекти/операции/услуги	Потребител	Група		Проект
				Потребителска	по интереси	
1.	***1.1.** Степен на регулиране*					
		регистрация, профил	✓	✓	✓	✓
		роля, права на достъп	✓			
	***1.2.** Избор на средства и услуги:*					
		абонамент, лично работно пространство, бележник с контакти, входяща кутия, изходяща кутия	✓			
		абонамент, споделено работно пространство, календар, списък членове на група, главна папка	✓	✓	✓	✓
		списък със задачи	✓			✓
2.	***2.1.** Изтегляне, зареждане, въвеждане, създаване на ресурси*					
		качване на файлове, създаване на папки	✓	✓	✓	✓
		тематика			✓	✓
		общо представяне				✓
	***2.2.** Редактиране на ресурси*					
	• *промяна, изриване*	работен прозорец	✓	✓	✓	✓
	2.3. Поддръжка (операции) на ресурси:					
	• сортиране, странициране за обекти-контейнери	работен прозорец	✓	✓	✓	✓
	2.4. Индикация за ресурс в реално време:					
	• за състоянието (ативен, неактивен, нестартиран, в процес, чакащ, завършен)	икони за състоянието	✓	✓	✓	✓
	• на събитията (нов, препратен, прочетен за съобщения)	икони за събитията	✓			
	2.5. Защита и сигурност:					

Нап-равле-ния	Функции	W-EPROM обекти/операции/услуги	Пот-ребите-л	Група		Проект
				Потребител-ска	по интереси	
	• контрол на достъпа	права на достъп	✓			
	• заключване на ресурси	работен прозорец	✓	✓	✓	✓
3.	***3.1.** Комуникация*					
	• изпращане на съобщения	съобщение	✓	✓	✓	✓
4.	***4.1.** Споделяне*					
		файлове, папки,	✓			
	***4.2.** Ползване на съвместни ресурси*					
		списък членове на група		✓	✓	✓
		календар, списък паметки			✓	✓
		списък задачи, съобщения	✓	✓	✓	✓
5.	***5.1.** Планиране на работата*					
		задача	✓	✓	✓	✓
		работен план				✓
	***5.2.** Изпълнение и управление на работата*					
		списък задачи, състояние и характерис-тики на задачата	✓	✓	✓	✓
		състояние и характеристики на работен план				✓
	5.3. Уведомление и напомняне					
	• ежедневен отчет или директно уведомле-ние за събитията	съобщение		✓	✓	✓
	• напомняне за ангажимент	съобщение, паметка	✓	✓	✓	✓
	• повторно изпращане на документ	съобщение	✓	✓	✓	✓
6.	***6.1.** Оценяване на документи*					
		работно пространство		✓	✓	✓
	***6.2.** Отчет на задача*					
		задача, списък задачи		✓	✓	✓

Таблица 3.2. Функционалност на W-EPROM

3.3. Дефиниране на актьорите и случаите на употреба за системата

Базирайки се на събраните и систематизирани бизнес изисквания (вж. 3.1) и предложения идеен проект (вж. 3.2), съвсем естествено можем да преминем към дефинирането на актьорите на системата и оформянето на случаите на употреба.

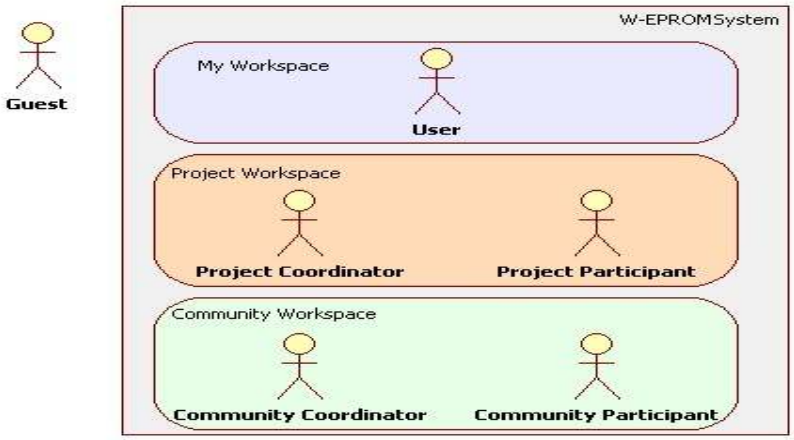
3.3.1. Актьори

Актьор на системата е всичко от външния свят, което си взаимодейства с нея. В нашия случай това са потребителите. Определяме 6 (шест) типа актьори, според следните два критерия:

- дали са гости или са регистрирани потребители на W-EPROM;
- работното пространство, в което се намират, когато са в системата и ролята, която играят в него.

На фиг. 3.1 са илюстрирани актьорите както следва:

- актьор **Guest**- всеки потребител, който е гост на системата;
- актьор **User**- всеки регистриран потребител, който се намира в личното си работно пространство;
- актьор **Project Participant**- всеки регистриран потребител, който се намира в работното пространство на проект, в който членува с роля „участник";
- актьор **Project Coordinator**- всеки регистриран потребител, който се намира в работното пространство на проект, в който членува с роля „координатор";
- актьор **Community Participant**- всеки регистриран потребител, който се намира в работното пространство на група по интереси, в която членува с роля „участник";
- актьор **Community Coordinator**- всеки регистриран потребител, който се намира в работното пространство на група по интереси, в която членува с роля „координатор".

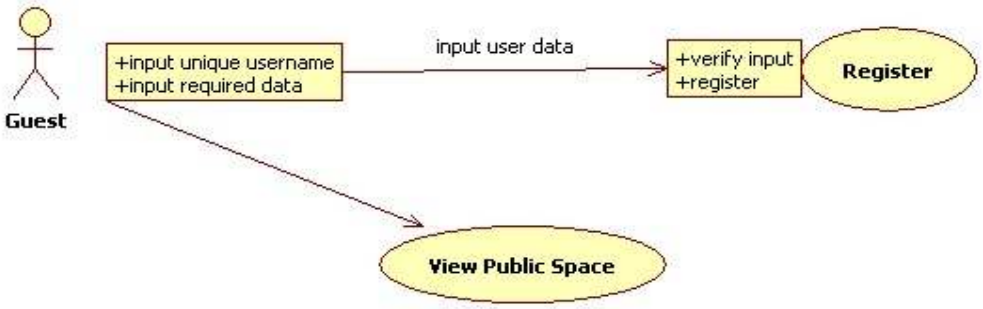


Фигура 3.1. Актьори в W-EPROM

3.3.2. Случаи на употреба

При така дефинираните актьори, нека да проследим и случаите на употреба, в които те участват (Фиг. 3.2).

3.3.2.1. Случаи на употреба за актьор Guest



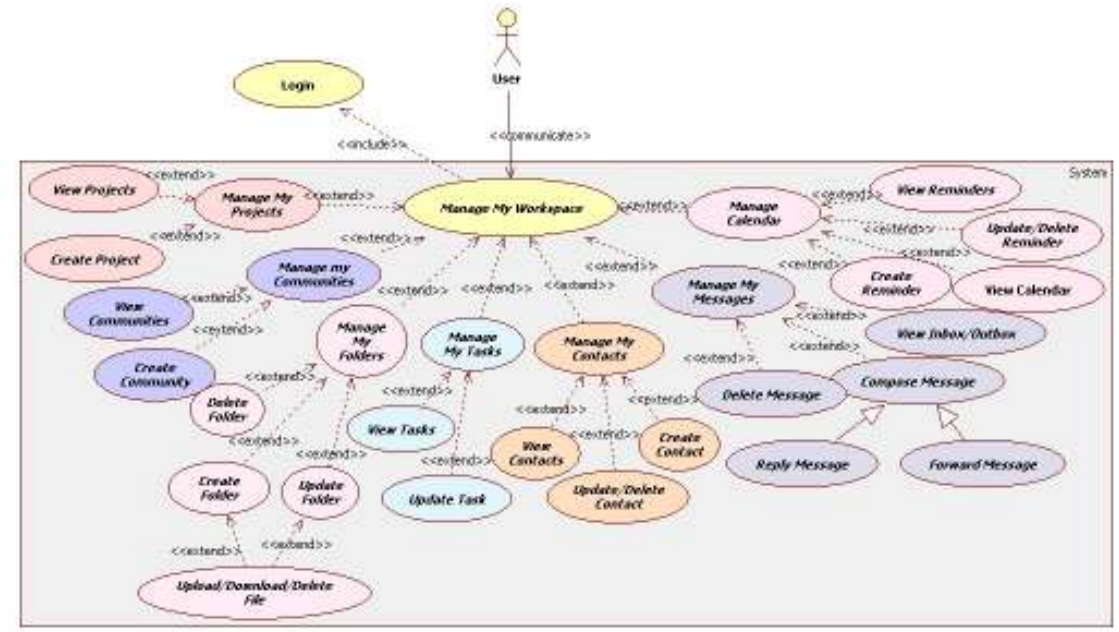
Фигура 3.2. Диаграма на случаите на употреба за актьор Guest

Актьор с тази роля може единствено да разгледа публичното пространство на W-EPROM и да се регистрира. При регистрация, гостът въвежда някой задължителни данни, между които е и потребителското му име (прякор), с което ще влиза в системата. Тя поддържа списък от уникални имена, по които идентифицира потребителите си. Затова преди да регистрира всеки гост, системата проверява дали въведеното от него име вече не съществува. Освен потребителското име, гостът трябва да въведе и друга задължителна информация като парола, фамилия, електронен адрес; пожелателни са данни като име, телефон и коментар за себе си.

3.3.2.2. Случаи на употреба за актьор User

Актьорът User има привилегията да управлява своето работно пространство. Разбира се, преди да влезе в него, той трябва да е бил идентифициран от W-EPROM като неин член чрез логин (Фиг. 3.3). В работното си пространство всеки регистриран потребител може да работи със своите

- проекти- създаване на нов проект, работа със списъка с проекти;
- групи по интереси- създаване на нова група, работа със списъка с групи по интереси;
- папки и документи- създаване, редакция и изтриване на папка, качване, сваляне, изтриване на файл;
- задачи- редактиране на задача, работа със списъка със задачи;
- контакти- създаване, редакция и изтриване на контакт, разглеждане на бележника с контакти;
- съобщения- работа с входяща и изходяща кутия, създаване на ново съобщение, отговаряне, препращане и изтриване на съобщение;
- календар- създаване, редакция и изтриване на паметка, работа с календара.

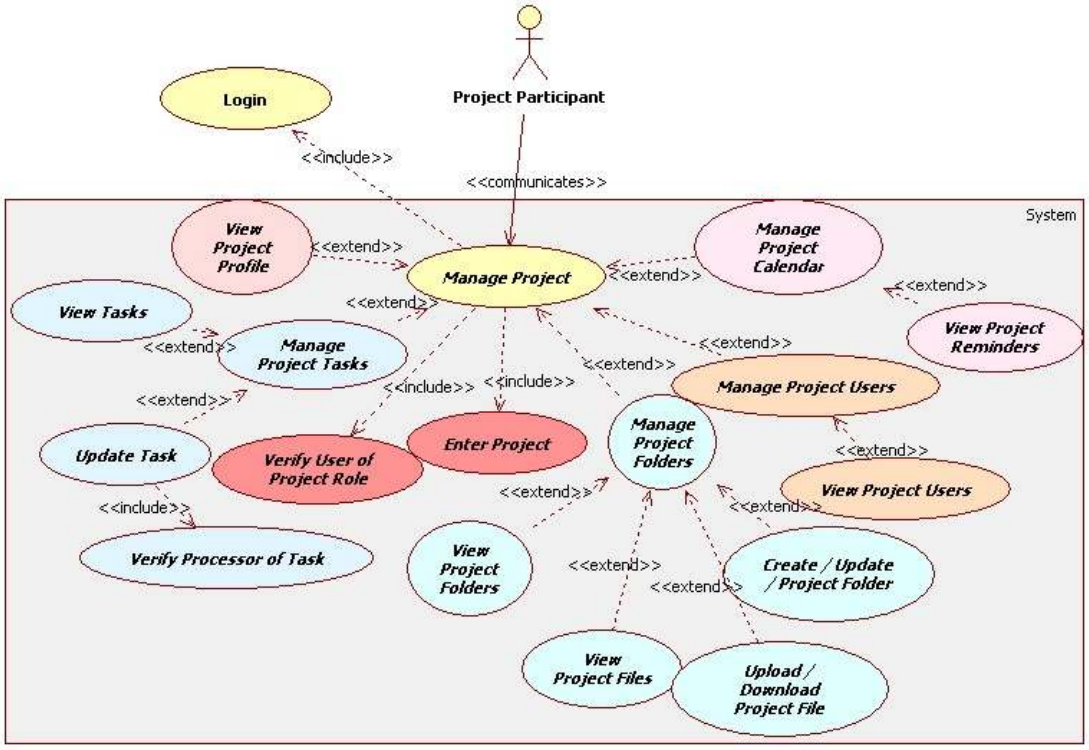


Фигура 3.3. Диаграма на случаите на употреба за актьор User

3.3.2.3. Случаи на употреба за актьор Project Participant

Актьорът Project Participant има възможността да използва работното пространство на проекта, в който членува (Фиг. 3.4). Той се допуска до него, обаче, само след успешен логин в W-EPROM. В работното пространство на проекта участникът има ограничени права. Той може да

- разглежда проектния профил;
- работи с папките и документите на проекта, което включва разглеждане, създаване и редакция на папка, разглеждане, качване и сваляне на файл;
- редактира задачите, на които е изпълнител, да работи със списъка със задачи;
- разглежда проектния календар и паметки;
- работи със списъка от членове на проекта.

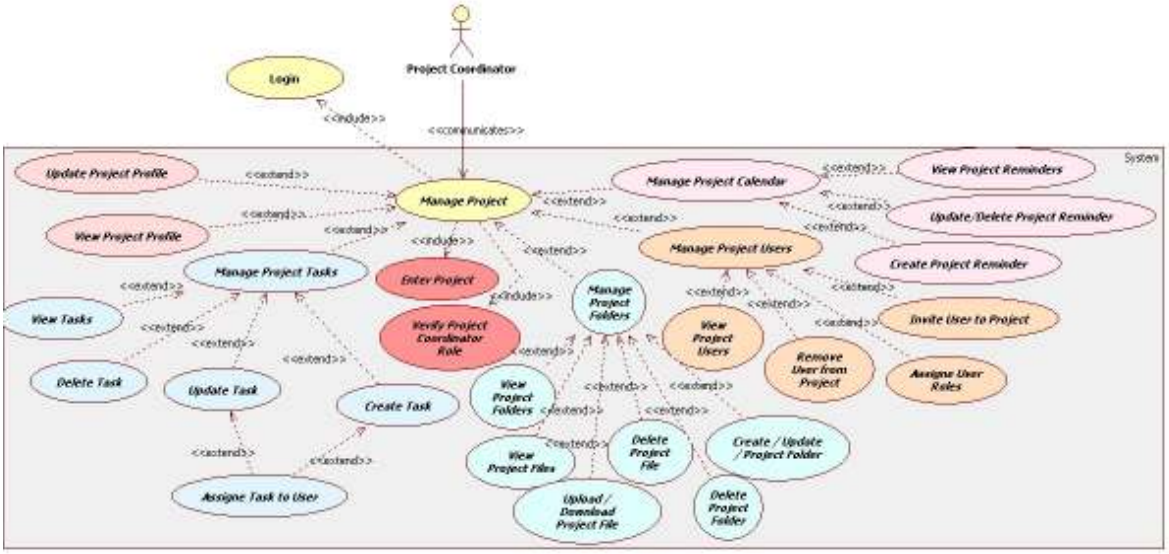


Фигура 3.4. Диаграма на случаите на употреба за актьор Project Participant

3.3.2.4. Случаи на употреба за актьор Project Coordinator

Актьорът Project Coordinator има привилегията да управлява работното пространство на проекта, на който е координатор (Фиг. 3.5). Той има достъп до него само след успешен логин в W-EPROM. В работното пространство на проекта всеки координатор може да

- редактира и разглежда проектния профил;
- работи с папките и документите на проекта, което включва разглеждане, създаване, редакция и изтриване на папка, разглеждане, качване, сваляне и изтриване на файл;
- създава задачи, да възлага, редактира и изтрива задачите, на които е собственик, да работи със списъка със задачи;
- създава, редактира и изтрива пametките на проекта, да работи със списъка от пametки и проектния календар;
- кани за членство нови потребители в проекта, да редактира ролите на членовете на проекта, да изтрива член на проект, да работи със списъка от членове на проекта.

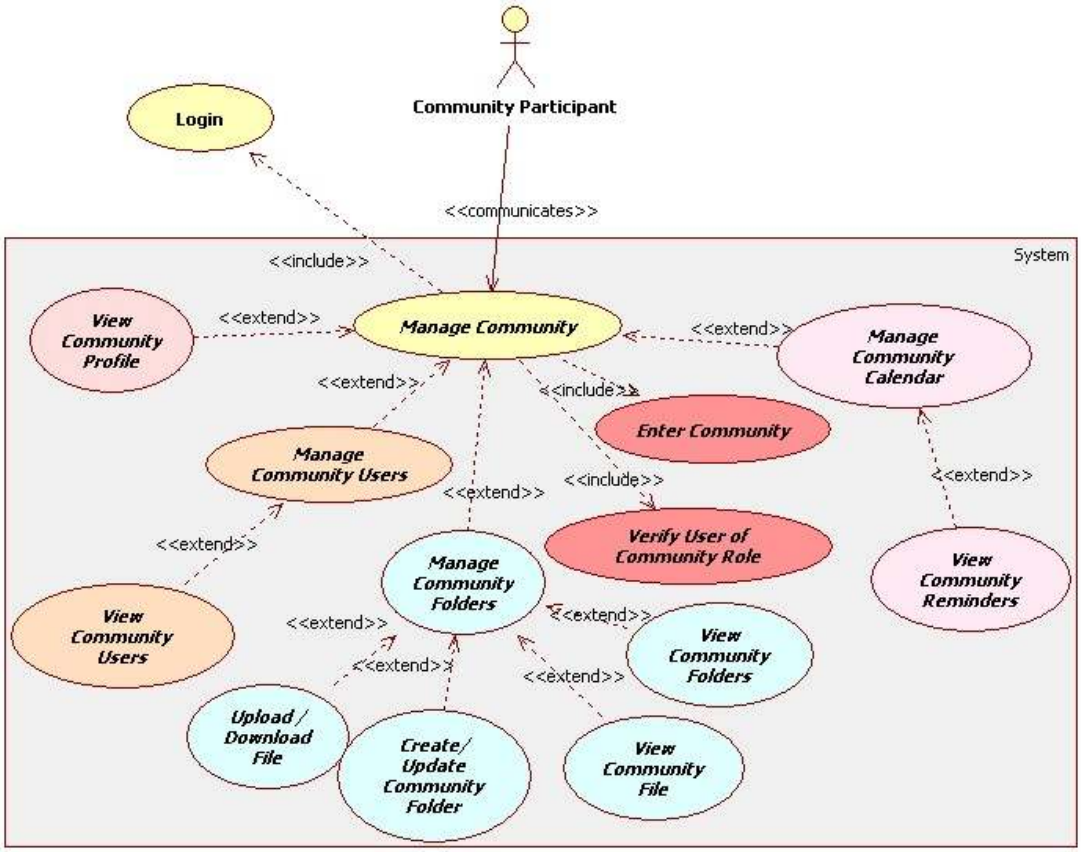


Фигура 3.5. Диаграма на случаите на употреба за актьор Project Coordinator

3.3.2.5. Случаи на употреба за актьор Community Participant

Актьорът Community Participant има достъп до работното пространство на групата по интереси, в която членува (Фиг. 3.4), но отново само след успешен логин в системата. В работното пространство на групата участникът има ограничени права. Той може да

- разглежда профила на групата по интереси;
- работи с папките и документите на групата, което включва разглеждане, създаване и редакция на папка, разглеждане, качване и сваляне на файл;
- разглежда календара и пametките на групата;
- работи със списъка от членове на групата.

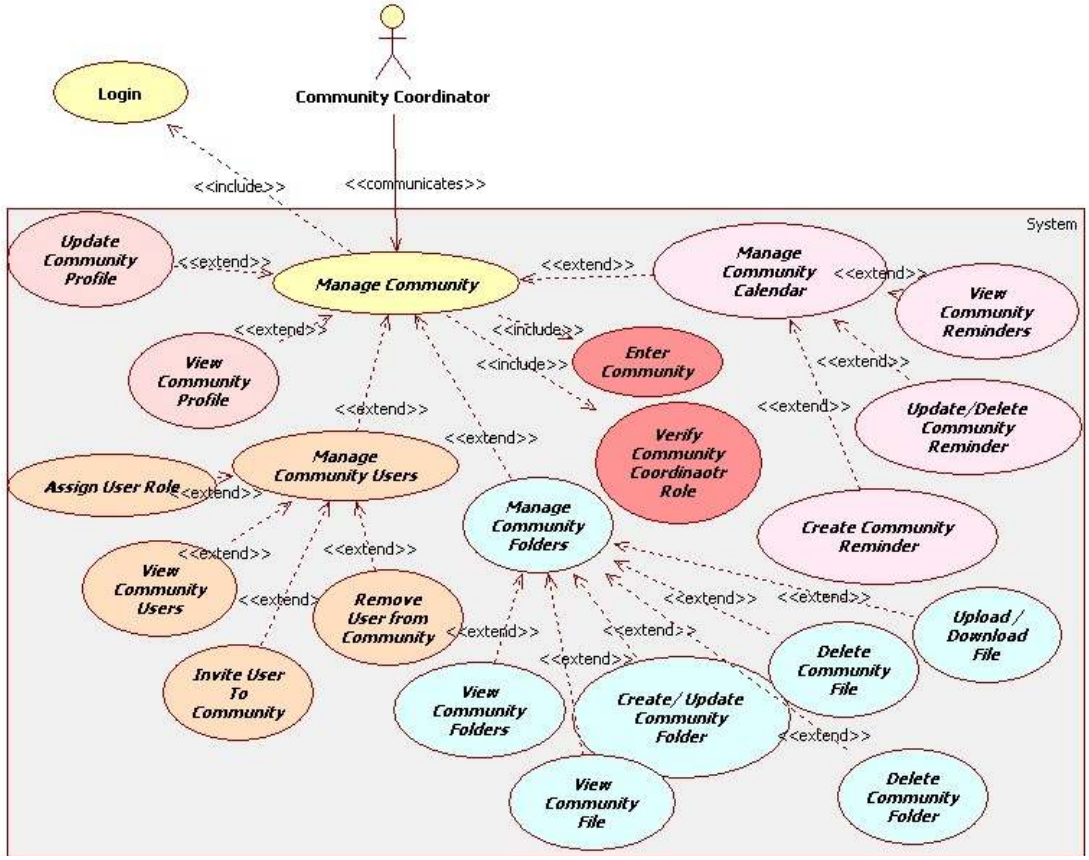


Фигура 3.6. Диаграма на случаите на употреба за актьор Community Participant

3.3.2.6. Случаи на употреба за актьор Community Coordinator

Актьорът Community Coordinator е оторизиран да управлява работното пространство на групата по интереси, на която е координатор (Фиг. 3.5). Той има достъп до него само след успешна идентификация от системата. В работното пространство на групата по интереси всеки координатор може да

- редактира и разглежда профила на групата по интереси;
- работи с папките и документите на групата, което включва разглеждане, създаване, редакция и изтриване на папка, разглеждане, качване, сваляне и изтриване на файл;
- създава, редактира и изтрива пametките на групата, да работи със списъка от пametки и груповия календар;
- кани за членство нови потребители в групата, да редактира ролите на членовете й, да изтрива член на групата, да работи със списъка от членове на групата.



Фигура 3.7. Диаграма на случаите на употреба за актьор Community Coordinator

4. Избор на технологични платформи за реализацията на проекта

Класически подход при разработване на Web-приложения е отговорностите за общата функционалност на системата да се разделят на три слоя: презентационен, приложен (на бизнес логиката) и слой база данни. Всеки от тях има грижата за определена страна приложението и не бива да припокрива функционалността си с останалите слоеве. Всеки слой е изолиран от останалите, но позволява комуникация с тях. За разбиране на тази трислойна архитектура, много подходяща е метафората формулирана от Зелдман [32]: „Ако вашето web- приложение беше филм, то Презентационният слой щеше да е неговият сценарист и артистичен директор, Приложният слой щеше да е неговият отговорник по специалните ефекти и слоят база данни – неговият директор на продукция". Или с други думи:

- Презентационният слой отговаря за получаването на входящите HTTP заявки и сформирането на HTML отговори.
- Приложният слой отговаря за цялата бизнес логика;
- Слоят база данни отговаря за достъпа до базата данни и сигурното съхраняване на данните.

4.1. Избор на База Данни – MySQL

За реализацията на слоят база данни е избрана най-популярната в света база данни с отворен код – MySQL [44]. Тя е безплатна и до момента предлага богата функционалност сравнима с тази на най-добрите комерсиални бази от данни (Oracle [45], Microsoft SQL Server [46], IBM DB2 Server [47]).

През последниет десет години общността за отворения код (*Open Source Community*) видимо подобри качеството на своя софтуер до такава степен, че осмисли инициативата за създаване на безплатен код. Като резултат много търговски преприятия показаха интерес към миграция от патентен комерсиален софтуер към софтуер с отворен код. Например забелязваме, че световният бизнес обикновено използа Linux [48] операционната система, Perl [49] езикът за програмиране, Apach Web сървърът [50] и двете водещи бази от данни с отворен код - PostgreSQL [51] и MySQL.

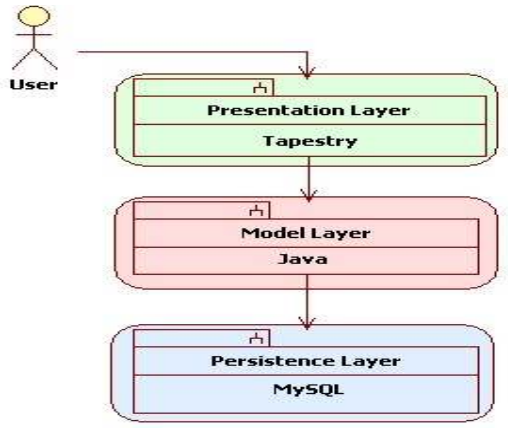
Въпросът за избор между PostgreSQL и MySQL базите от данни често опира до въпросът за производителността. Но скоростта не е всичко когато въпросът е да се избере подходящо решение като се вземат предвид предлаганите възможности. И двете базии данни предлагат добра стабилност, гъвкавост и производителност. MySQL има възможности, който PostgreSQL не предлага и обратното.

Защо е избрана MySQL база данни ?

- Безплатна е.
- Предлага по-малка мрежова натовареност.
- Бърза е и доста добре оптимизирана.
- Предлага гъвкав интерфейс към множество формати данни.
- Лесна за изучаване.
- Предлага благонадеждно архивиране.
- Има добра поддръжка.

5. Описание на реализацията

W-EPROM системата по същество е уеб приложение, написано на Java програмен език в Eclipse [40] интегрираната среда за разработка. Освен Java програмния език, за изграждането й са използвани езикът за работа с база данни SQL [35], дескриптивните маркиращи езици (*descriptive mark-up languages*) XML и HTML, както и езикът за навигация в граф от обекти OGNL (*Object-Graph Navigation Language*) [36]. В допълнение, приложението е разработено и с помощта на HiveMind платформата [37], Tapestry component web-framework [8], JDBC (*Java Database Connectivity*) [38] и CSS технологията [39]. W-EPROM приложението има трислойна логическа архитектура (фиг. 5.1)



Фигура 5.1. Трислойна логическа архитектура

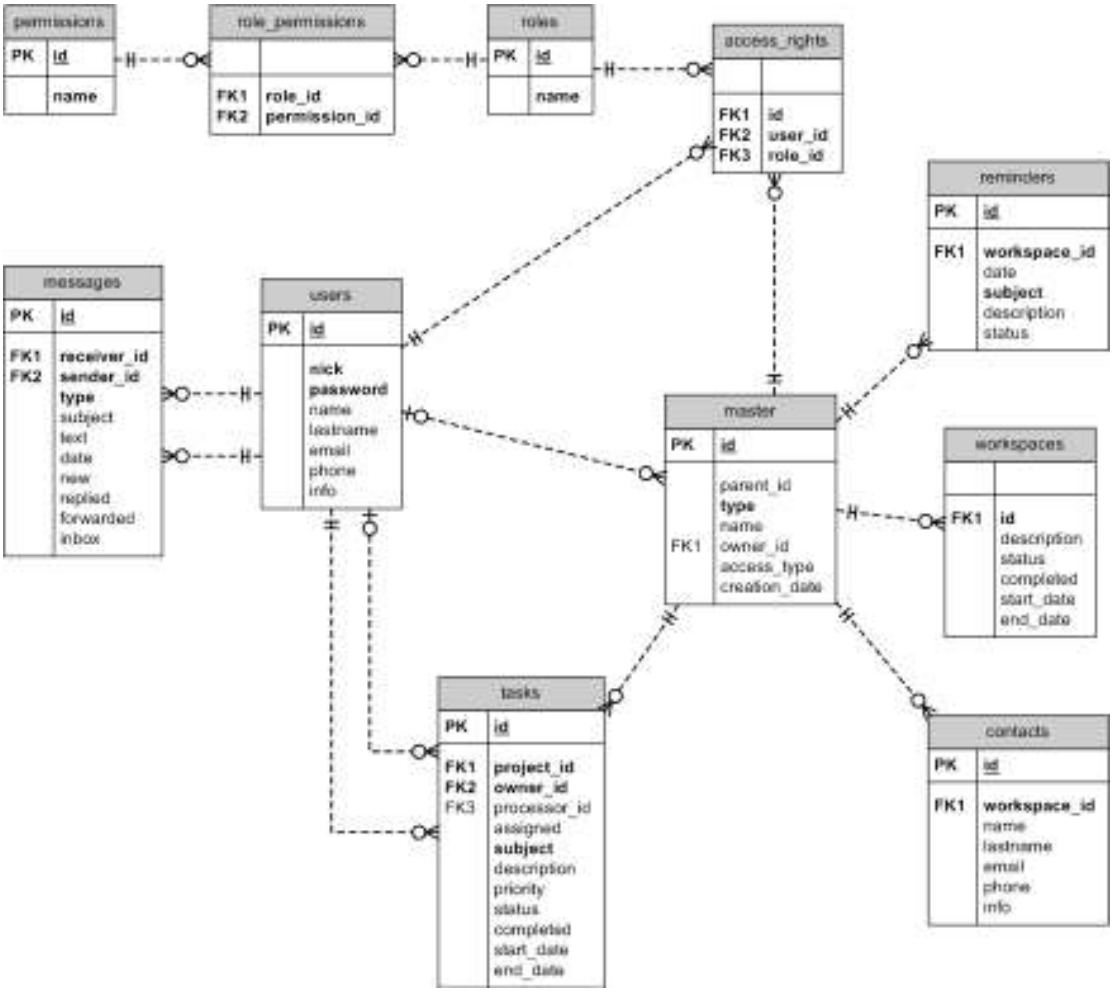
Постоянният слой използва MySQL базата данни. Приложният слой е написан на Java, а слоят Представяне е изграден чрез рамката за уеб приложения Tapestry. Разделянето на достъпа до данните, бизнес логиката, и логиката на потребителския интерфейс в три отделни подсистеми на приложението има много предимства които включват:

- лесен за поддръжка и оптимизация код;
- чиста декомпозиция на функционалността, благодарение на която в процеса на разработка концентрирането върху една или друга част от приложението ставаше лесно.
- достъпът до данни е съсредоточен на едно място, което прави създаването и поддръжката на приложението лесно.

Следвайки логиката на архитектурата на системата W-EPROM, в тази глава подробно е разгледана реализацията на слоят база данни и слоят бизнес логика.

5.1. Слой база данни

W-EPROM използва MySQL базата, за да осигури съхранението на своите данни. Физическата схема на базата данни беше създадена след внимателно анализиране на изискванията към системата и бяха взети предвид и условията в която тя ще се използва.



Фигура 5.2. UML-диаграма на модела на базата данни

Базата данни (БД) от разглеждания слой на системата е проектирана с единадесет таблици, съдържащи данните съответно за йерархията на работните пространства, директорийте и документите (**master**), потребителите (**users**),

правата за достъп (**permissions**), ролите (**roles**), релацията между правата за достъп и ролите (**role_permissions**), ролите на потребителите в отделните работни пространства (**access_rights**), допълнителна информация за работните пространства (**workspaces**), съобщенията (**messages**), задачите (**tasks**), напомнянията (**reminders**) и контактите (**contacts**). Моделът на БД е представен чрез диаграма на таблиците и релациите между тях (Фиг. 5.2).

В следващите подраздели е дадено описание на таблиците на БД, както и предназначението на техните *колони* (домени), които определят смисъла на данните в отделните полета на таблиците.

5.1.1. Таблица master

Това е основната таблица (Табл. 5.1) в модела на базата данни за W-EPROM. Тя описва йерархична структура от данни, подобна на тази при файловите системи и се състои от директории и файлове, като в допълнение директориите могат да бъдат няколко типа. Описание на възможните типове обекти е дадено по-долу за колоната type.

Име на колона	Тип	NULL	PK
id	Int (11)	-	+
parent_id	Int (11)	+	
type	Tinyint (4)	-	
name	Varchar (255)	+	
owner_id	Int (11)	+	
access_type	Tinyint (4)	+	
size	Bigint(20)	+	
creation_date	Datetime	+	

Таблица 5.1. Наименование и тип на домените за таблица master

Описанието на колоните е, както следва:

- id – Уникален идентификатор на обекта.
- parent_id – Идентификатор на обекта-предшественик (контейнер). Стойност NULL в тази колона означава, че обектът няма предшественик, т.е. той е корен на дървото. Реферира ред от тази таблицата (*self-join*). Това поле на таблицата именно прави възможно описанието на йерархичната структура.
- type – Тип на обекта. Възможните типове обекти са:
  MASTER_TYPE_USERWORKSPACE – Потребителско пространство.

MASTER_TYPE_PROJECT – Проект.

MASTER_TYPE_COMMUNITY – Група по интереси.

MASTER_TYPE_FOLDER – Директория за документи.

MASTER_TYPE_FILE –Документ (Файл).

- name – Име на обекта.
- owner_id – Идентификатор на притежателя на обекта. Реферира ред от таблицата users.
- access_type – Тип на достъп. Възможните типове са :

  ACCESS_TYPE_PUBLIC – Публичен достъп. Използва се за обозначаване на документите, достъпни за всички потребители на системата, както и за нерегистрирани (анонимни) потребители.

  ACCESS_TYPE_PRIVATE – Частен достъп. Използва се за обозначаване на документите в потребителското пространство.

  ACCESS_TYPE_SHARED – Споделен достъп. Използва се за обозначаване на документите в проектите или групите по интереси.

  ACCESS_TYPE_INHERITED – Наследен достъп или достъп по подразбиране. За да се избегне изричното обознаване на типа на достъп до всеки обект, се използа типът на достъп до обекта предшественик (контейнер). Ако неговият тип на достъп е отново наследен, то се обръщаме към неговия предшественик и така нагоре по йерархията, докато стигнем до различен от наследен достъп.
- creation_date – Дата и време на създаване на обекта.
- size – Големината на документа (файла).

5.1.2. Таблица users

Таблицата (Табл. 5.2) съдържа информация за всички реистрирани в системата W-EPROM потребители.

Име на колона	Тип	NULL	PK
id	int(11)	-	+
nick	varchar(10)	-	
password	varchar(10)	-	
name	varchar(30)	+	
lastname	varchar(30)	+	
email	varchar(20)	+	
phone	varchar(25)	+	
info	varchar(255)	+	

Таблица 5.2. Наименование и тип на домените за таблица users

Описанието на колоните е, както следва:

- id – Уникален идентификатор на потребителя.
- nick – Уникален псевдоним на потребителя (името под което се логва във W-EPROM системата).
- password – Парола на потребителя.
- name – Име на потребителя.
- lastname – Последното име на потребителя.
- email – E-mail адрес на потребителя.
- phone – Телефон на потребителя.
- Info – Допълнителна информация за потребителя.

5.1.3. Таблица permissions

Таблица (Табл. 5.3), описваща всички права за достъп във системата W-EPROM. Например правото за модернизиране на данните за даден проект е предефинирано и носи името "*project.update*".

Име на колона	Тип	NULL	PK
id	int(11)	-	+
name	varchar(30)	-	

Таблица 5.3. Наименование и тип на домените за таблица permissions

Описанието на колоните е, както следва:

- id – Уникален идентификатор на право за достъп.
- name – Уникално име на право за достъп.

5.1.4. Таблица roles

Таблица (Табл. 5.4), описваща всички роли в системата W-EPROM.

Име на колона	Тип	NULL	PK
id	int(11)	-	+
name	varchar(30)	-	

Таблица 5.4. Наименование и тип на домените за таблица roles

В системата W-EPROM има първоначално дефинирани следните роли:

- Administrator
- CommunityCoordinator
- CommunityParticipant

- Guest
- ProjectCoordinator
- ProjectParticipant
- User (Member)

При нужда от въвеждане на нова роля се прави нов запис във тази таблица с името и идентификатора й. В допълнение трябва да се опишат правата за достъп, които принадлежат на тази роля в следващата таблица **role_permissions**.

Описанието на колоните е, както следва:

- id – Уникален идентификатор на ролята.
- name – Уникално име на ролята.

5.1.5. Таблица role_permissions

Тази таблица (Табл. 5.5) описва релацията между правата за достъп от таблицата **permissions** и ролите в таблицата **roles**, т.е. коя роля какви права за достъп включва в себе си.

Име на колона	Тип	NULL	PK
role_id	int(11)	-	
permission_id	int(11)	-	

Таблица 5.5. Име и тип на домените за таблица role_permissions

Описанието на колоните е, както следва:

- role_id – Уникален идентификатор на ролята. Реферира ред от таблицата roles (*foreign key*).
- permission_id – Уникално име на право за достъп. Реферира ред от таблицата permissions (*foreign key*).

5.1.6. Таблица access_rights

Тази таблица (Табл. 5.6) описва ролите, с които потребителите участват в проектите и групите по интереси.

Име на колона	Тип	NULL	PK
id	int(11)	-	
user_id	int(11)	-	
role_id	int(11)	-	

Таблица 5.6. Наименование и тип на домените за таблица access_rights

Описанието на колоните е, както следва:

- id – Уникален идентификатор на обекта (проект / група по интереси). Реферира ред от таблицата master.
- user_id – Идентификатор на потребителя. Реферира ред от таблицата users.
- role_id – Уникален идентификатор на ролята. Реферира ред от таблицата roles (foreign key).

5.1.7. Таблица workspaces

Тази таблица (Табл. 5.7) предоставя допълнителна информация за някои от типовете обекти, които принадлежат на таблицата **master**. По специално се използва само за проект и група по интереси.

Име на колона	Тип	NULL	PK
id	Int (11)	-	
description	varchar(255)	+	
status	Tinyint (4)	+	
completed	Tinyint (4)	+	
start_date	Date	+	
end_date	Date	+	

Таблица 5.7. Наименование и тип на домените за таблица workspaces

Описанието на колоните е, както следва:

- id – Уникален идентификатор на обекта (проект / група по интереси). Реферира ред от таблицата master.
- description – Описание на обекта.
- status – Статус на проект, може да е един от изброените:
  NOTSTARTED – Проектът не е стартиран.
  INPROCESS – Проектът е в процес на изпълнение.
  WAITING – Проектът е в процес на на изчакване.
  COMPLETED - Проектът е завършен.
- completed – Съдържа степента на завършеност на проекта – 0% – 100%.
- start_date – Начална дата.
- end_date – Крайна дата.

5.1.8. Таблица messages

Всеки ред от таблицата (Табл. 5.8) описва едно съобщение. Потребителите могат да изпращат помежду си текстови съобщения (TEXT) и системни съобщения. Системните съобщения биват:

- PROJECT_INVITATION - Покана за участие в проект.
- COMMUNITY_INVITATION - Покана за участие в група по интереси.
- TASK_ASSIGNMENT - Заявка за възлагане на задача.

Възможните отговори на системно съобщение са два – приел(а) или отказал(а), като при отговор системата автоматично изпраща текстово съобщение-рапорт обратно до изпращача.

Име на колона	Тип	NULL	PK
id	Int (11)	-	+
receiver_id	Int (11)	-	
sender_id	Int (11)	-	
type	Tinyint (4)	-	
subject	varchar(128)	+	
text	varchar(786)	+	
date	Datetime	+	
new	Boolean	+	
replied	Boolean	+	
forwarded	Boolean	+	
inbox	Boolean	+	

Таблица 5.8. Наименование и тип на домените за таблица messages

Описанието на колоните е, както следва:

- id – Уникален идентификатор на съобщението.
- receiver_id – Идентификатор на потребителя-получател. Реферира ред от таблицата users.
- sender_id – Идентификатор на потребителя-изпращач. Реферира ред от таблицата users.
- type – Тип на съобщението. По-горе са изброени четирите възможни типа съобщения.
- subject – Предмет на съобщението.
- text – Текстово съдържание на съобщението. При системните съобщения тази колона може да съдържа списък от идентификатори на проекти, групи по интереси или задачи.
- date – Дата и време на изпращане на съобщението.

- new – При стойност TRUE - имаме ново съобщение (непрочетено), а при стойност FALSE - старо съобщение (прочетено).
- replied – При стойност TRUE - имаме съобщение, на което е било отговорено, а при стойност FALSE - не.
- forwarded – При стойност TRUE - имаме съобщение, което е било препратено, а при стойност FALSE - не.
- inbox – За всяко изпратено съобщение имаме по едно копие за изпращача (FALSE) и по едно за получателя (TRUE).

5.1.9. Таблица tasks

Всеки ред от таблицата (Табл. 5.9) описва задача от проект.

Име на колона	Тип	NULL	PK
id	Int (11)	-	+
project_id	Int (11)	-	
owner_id	Int (11)	-	
processor_id	Int (11)	+	
assigned	Boolean	+	
subject	Varchar(128)	-	
description	Varchar(786)	+	
priority	Tinyint (4)	+	
status	Tinyint (4)	+	
completed	Tinyint (4)	+	
start_date	Date	+	
end_date	Date	+	

Таблица 5.9. Наименование и тип на домените за таблица tasks

Описанието на колоните е, както следва:

- id – Уникален идентификатор на задача.
- project_id – Уникален идентификатор на проект. Реферира ред от таблицата master.
- owner_id – Идентификатор на потребителя-създател на задачата. Реферира ред от таблицата users.
- processor_id – Идентификатор на потребителя, отговорен за изпълнение на задачата. Реферира ред от таблицата users.
- assigned – При TRUE задачата има назначен изпълнител, а при FALSE – няма такъв.
- subject – Предмет на задачата.
- description – Описание на задачата.

- priority – Приоритет на задачата. Може да бъде една от следните стойности:

  LOW – Нисък.

  MEDIUM – Среден.

  HIGH – Висок.

  VERYHIGH – Много висок.
- status – Състояние на задачата. Може да е едно от изброените:

  NOTSTARTED – Изпълнението на задачата не е започнало.

  INPROCESS – Задачата е в процес на изпълнение.

  WAITING – Задачата е в процес на изчакване.

  COMPLETED – Изпълнението на задачата е завършено.
- completed – Съдържа степента на завършеност на задачата – 0% – 100%.
- start_date – Начална дата.
- end_date – Крайна дата.

В зависимост от това дали задачата има назначен потребител за изпълнението ѝ, имаме няколко възможни състояния, кодирани в две от колоните на таблицата assigned и processor_id:

assigned	**processor_id**	Значение
TRUE / FALSE	NULL	Задачата няма назначен изпълнител.
FALSE	xxxxxxxx	Изпратено е системно съобщение - заявка за възлагане на задача до потребител с идентификатор xxxxxxxx. Потребителят все още не е отговорил на съобщението.
TRUE	xxxxxxxx	Задачата има назначен за изпълнител потребител с идентификатор xxxxxxxx

Таблица 5.10. Състояние на задачите от таблица tasks

5.1.10. Таблица reminders

Всеки ред от таблицата (Табл. 5.11) представлява паметка.

Име на колона	Тип	NULL	PK
id	Int (11)	-	+
workspace_id	Int (11)	-	
date	Datetime	+	
subject	Varchar(40)	-	
description	Varchar(256)	+	
status	Tinyint (4)	+	

Таблица 5.11. Наименование и тип на домените за таблица reminders

Описанието на колоните е, както следва:

- id – Уникален идентификатор на паметка.
- workspace_id – Уникален идентификатор на работно пространство (проект / група по интереси / лично пространство). Реферира ред от таблицата master.
- date – Дата и час на паметката.
- subject – Предмет на паметката.
- description – Описание на паметката.
- status – Статус на паметката, може да един от изброените:

  ON – Паметката е активна.

  OFF – Паметката е неактивна.

5.1.11. Таблица contacts

Всеки ред от таблицата (Табл. 5.12) представлява контактна информация.

Име на колона	Тип	NULL	PK
id	Int (11)	-	+
workspace_id	Int (11)	-	
name	Varchar(30)	+	
lastname	Varchar(30)	+	
email	Varchar(20)	+	
phone	Varchar(25)	+	
info	Varchar(256)	+	

Таблица 5.12. Наименование и тип на домените за таблица contacts

Описанието на колоните е, както следва:

- id – Уникален идентификатор на паметка.

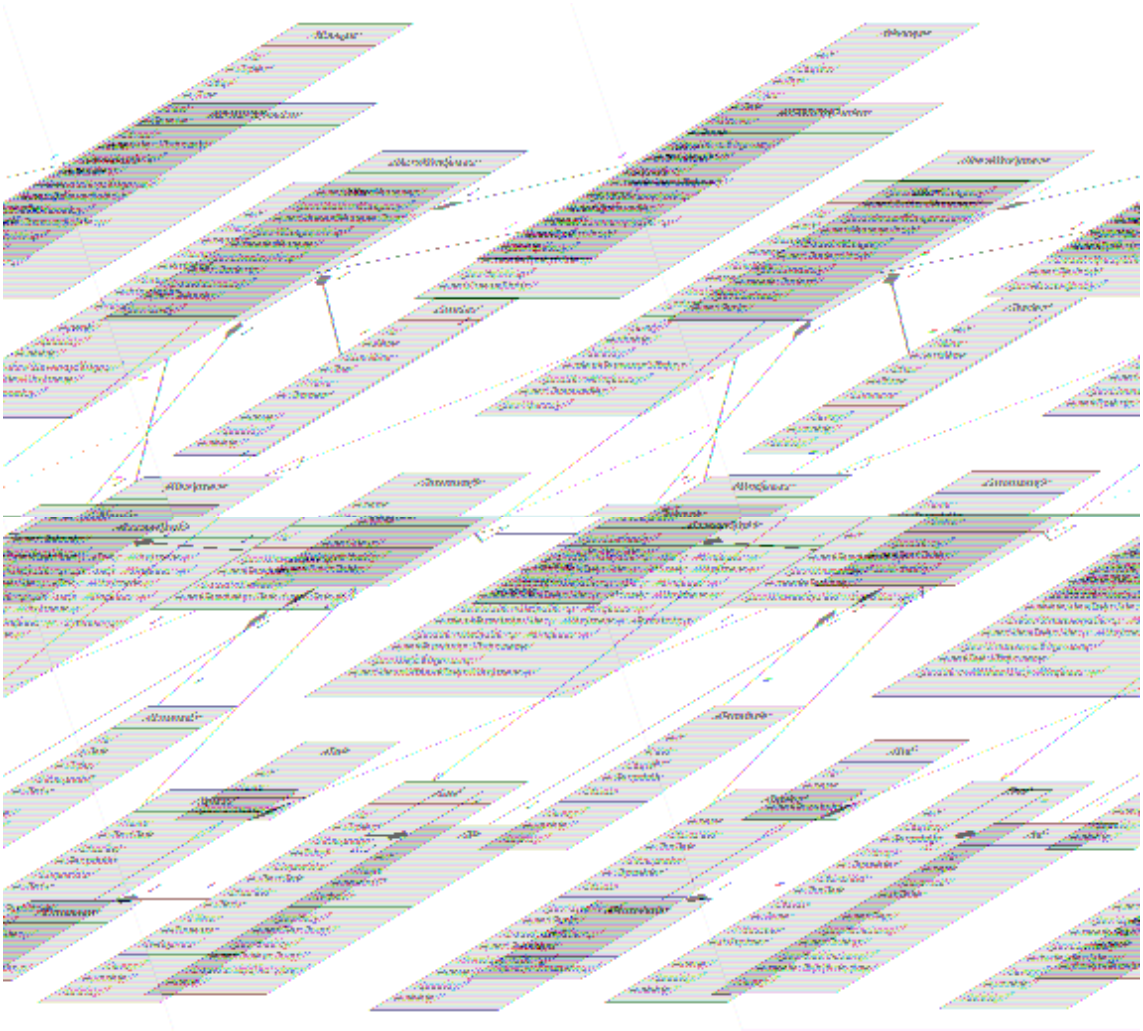
- workspace_id – Уникален идентификатор на работно пространство (проект / група по интереси / лично пространство). Реферира ред от таблицата master.
- name – Име на контактното лице.
- lastname – Последно име на контактното лице.
- email – E-mail адрес на контактното лице.
- phone – Телефон на контактното лице.
- info – Допълнителна информация за контактното лице.

5.2. Слой бизнес логика

Този слой на системата W-EPROM определя представянето на данните и операциите за всички обекти и субекти, играещи роля за функциите на системата на логическо ниво.

5.2.1. Проект на модела на данните

Проектът на модела на данните, или както още ще ги наричаме - обектите от домейна, дефинира типа на данните, които се използват във всички слоеве на web-приложението. Те са образ на обектите от реалността, като напирмер съобщение, задача, документ и т.н.



Фигура 5.3. UML клас-диаграма на модела на данните

Това е направено с цел унифициране на предаваните данни между слоевете. Т.е. едни и същи обекти се използват от презентационния слой при визу-

ализиране и промяна на аттрибутите им, след това същите се обработват от слоя бизнес логика и накрая се записват в базата данни. На Фиг. 5.3 е илюстрирана UML клас-диаграма на модела на обектите от домейна, а по-долу има подробно описание на всеки един от тях.

5.2.1.1. Описание на класовете и интерфейсите

Следващото изложение представлява описание на класовете и интерфейсите на обектите от домейна и на най-съществените техни характеристики.

Трябва да отбележим, че те споделят общ подход за създаване, обновяване и изтриване на обекти в базата данни.

За създаване имаме *createX()* метод, където *X* е името на съответния клас. Този метод създава нова инстанция, която е временна и можем да работим с нея и чак когато извикаме методът *save()* тя се запазва в базата данни. Обикновено всяка инстанция има образ в точно един ред от някоя от таблиците на базата данни. Ето и пример за създаване на нов потребител:

```
User admin = wepromSystem.createUser("admin");
admin.setPassword("admin");
admin.setName("James");
admin.setLastName("Gosling");
admin.setEmail("james@weprom.com");
admin.save();
```

Ако сме заредили обект от базата можем да променим някои от неговите атрибути, след което да актуализираме това в базата данни посредством метод *update()*:

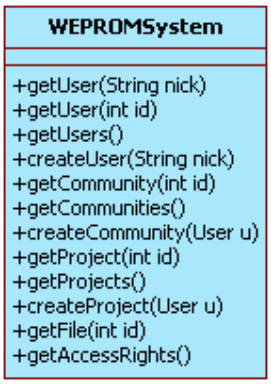
```
admin.setPhone("+3592123456789");
admin.setInformation("One of the creators of the Java programming language.");
admin.update();
```

Съответно ако имаме зареден вече обект можем да премахнем неговия образ от базата данни чрез метода *delete()*:

```
admin.delete();
```

Тук е важно да отбележим, че въпреки, че презентационният слой работи директно с обектите от домейна не се допуска директното им запазване в потребителската сесия. За целта се използва метод - *getID()* и съответно се запазват само уникалните идентификатори на обектите.

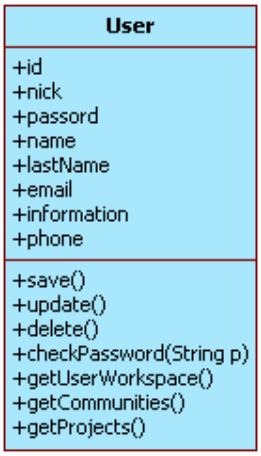
Клас WEPROMSystem



WEPROMSystem е базовият интерфейс в слоя бизнес логика. Той няма собствен образ в базата данни, а спомага за достъпването на всички останали обекти от домейна директно или индиректно. Чрез него можем да заредим всички потребители (*User* обекти), потребител по даден идентификатор и да създадем нов потребителски обект. Аналогично имаме същите методи за проекти и групи по интереси, с тази разлика, че потребител можем да заредим и по псевдоним (*nick name*). Това е нужно за логин сценария, защото задаваме псевдонима, а и няма как да знаем идентификаторът си.

Чрез метода *getAccessRights()* достъпваме обекта за управление на контрол на достъпа, който е единствен за системата (*singleton*).

Клас User

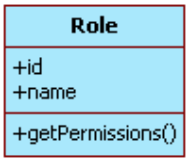


Този интерфейс представлява даден потребител на системата W-EPROM. Освен стандартните методи имаме метод за проверка на паролата *checkPassword(String)* нужен за логин сценария, както и методи за достъпване на проектите (*getProjects()*) и групите по интереси (*getCommunities()*), в които потребителят участва.

Клас Workspace

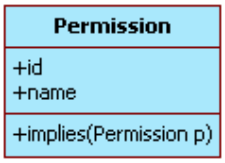
Базов интерфейс за всички типове работни пространства – потребителско пространство (*UserWorkspace*), проект (*Project*) и група по интереси (*Community*). Методът *getUsers()* дава достъп до всички участници в съответното работно пространство, като за *UserWorkspace* имаме точно един фиксиран потребител. Методът *getRootFolder()* предоставя инстанция на главната директория за документи на съответното работно пространство.

Клас Role



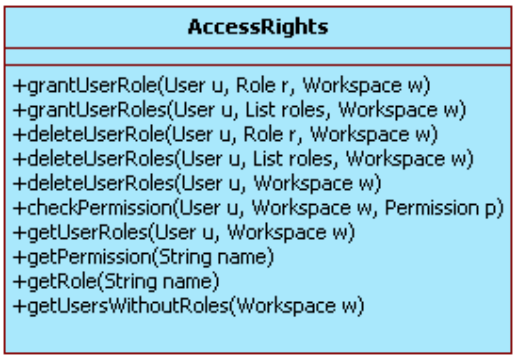
Класът представя дадена роля в системата W-EPROM. Всяка роля има име, уникален идентификатор и множество от права за достъп, достъпни чрез метода *getPermissions()*. Ето и някои от предефинираните имена на роли: *Administrator*, *ProjectParticipant*, *ProjectCoordinator*, *CommunityParticipant*, *CommunityCoordinator*.

Клас Permission



Класът представя дадено право за достъп. Всяко такова се състои от име и уникален идентификатор. Методът *implies(Permission)* се използва за проверка дали дадено право „включва" друго във себе си. Например правото за достъп с име *ALL* включва в себе си всички останали възможни права във системата и затова присъства само във администраторската роля с име *Administrator*.

Клас AccessRights



Този интерфейс служи за управление на контрола на достъп. Методите *grantUserRole(…)* и *grantUserRoles(…)* се използват при добавяне на потреби-

тел във проект или група по интереси или при обновяване на вече съществуващите му роли там. Съответно имаме методи за изтриване на една или повече роли на потребител в някое работно пространство - *deleteUserRole(…, Role role,…)* и *deleteUserRoles(…,List roles,…)*. Последният метод за изтриване на роли, при който не се специфицират никакви роли - *deleteUserRoles(User u, Workspace w)* по подразбиране изтрива всички роли на потребителя *u* във работното пространство *w*. Т.е. така премахваме участието на даден потребител в дадено работно пространство.

Методът *checkPermission(…)* се използва за проверка дали даден потребител, във дадено работно пространство има дадено право за достъп.

Имаме метод за намиране на роля по име – *getRole(String name)*, и за намиране на право за достъп по име – *getPermission(String name)*.

Методът *getUserRoles(…)* връща всички роли, с които даден потребител участва в дадено работно пространство.

Методът *getUsersWithoutRoles(Workspace w)* връща всички потребители, които нямат участие в дадено работно пространство. Той е подходящ за сценария, при който избираме нови участници за някое работно пространство.

Клас UserWorkspace

Този интерфейс представя потребителско работно пространство. Всеки регистриран потребител автоматично получава такова. То предоставя методи за работа със съобщения – прочитане на входящата (*getInboxMessages()*) и изходящата кутия (*getOutboxMessages()*), получаване броя на непрочетените съобщения (*getUnreadMessageCount()*) и създаване на ново съобщение (*createMessage()*). Методи за работа с контакти, достъп до календар (*getCalendar()*) и

достъп до всички задачи възложени на потребителя в различните проекти (*getTasks()*).

Клас Project

Този интерфейс представя потребителско пространство от тип проект. Всеки проект е съставен от произволен брой задачи и съответно имаме методи за достъп до тях – *getTask(int id)* за намиране на задача по уникален идентификатор, *getTasks()* за намиране на всички задачи и *createTask(User u)* за създаване на задача в проекта. Всеки проект има календар и той може да бъде достъпен чрез метода *getCalendar()*.

По-голям интерес представлява методът *inviteUser(User u)*, който изпраща системно съобщение-покана за участие в проекта на което потребителят може да отговори със съгласие или несъгласие и съответно да бъде добавен или не в проекта.

Клас Community

Този интерфейс представя потребителско пространство тип група по интереси.

Освен стандарните методи, подобно на проекта и тук имаме календар и метод за достъпването му - *getCalendar()*, също и метод за покана на нови участници - *inviteUser(User u)*.

Клас Message

Този интерфейс представя съобщение. По общо имаме два типа съобщения – текстови и системни. Текстовите съобщения (*TEXT*) се използват за комуникация между потребителите на системата. Системните съобщения биват три вида и се изполват съответно за:

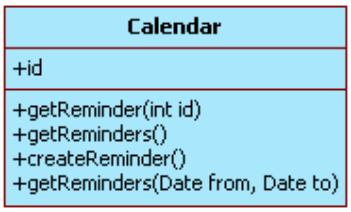
- *PROJECT_INVITATION* - Покана за участие в проект.
- *COMMUNITY_INVITATION* - Покана за участие в група по интереси.
- *TASK_ASSIGNMENT* - Заявка за възлагане на задача.

На всеки един тип системно съобщение потребителят може да отговори със съгласие (метод *accept()*) или несъгласие (метод *decline()*).

За изпращане на новосъздадено съобщение се използва методът *send()*. *Update()* методът се използва за обновяване на съобщението, като възможни за промяна са само атрибутите-флагове *New*, *Forwarded*, *Replied*.

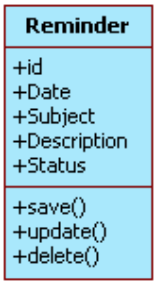
Важно да се отбележи също е, че отговор на системно съобщение може да се направи само един единствен път, при което системата отбелязва това като установява атрибутът *New* на съобщението със стойност TRUE.

Клас Calendar



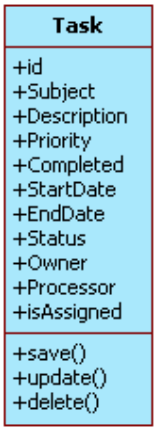
Calendar е интерфейсът предоставящ методи за достъп до пametките. Можем да търсим паметка(и) по уникален идентификатор чрез метод *getReminder(int id)* или по зададен период от време - *getReminders(Date from, Date to)*. Също можем да прочетем всички паметки (*getReminders()*) и да създадем нова такава (*createReminder()*).

Клас Reminder



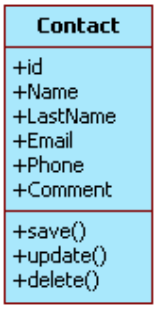
Интерфейс представящ една паметка. Паметката служи за напомняне на събития и принадлежи на точно един календар. Тя има за атрибути дата и час, тема и описание. Също допълнителна възможност дава атрибутът й състояние, който може да приема стойност „активно” (*ON*) и „неактивно” (*OFF*).

Клас Task



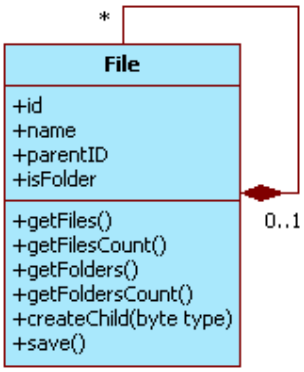
Този интерфейс представя задача от проект. Имаме само стандартните методи за запазване, обновяване и изтриване на обект в базата данни. Но зад някои от атрибутите се крие допълнителна логика. Важна роля играят *Processor* и *Assigned* атрибутите. Те определят дали задачата има назначен изпълнител или е изпратена заявка към потребител да изпълни задачата, но той все още не е отговорил. Заявката се изпраща автоматично при установяване на нова стойност на атрибута *Processor*.

Клас Contact



Този интефейс дефинира контактна информация. Налични са само стандартните методи.

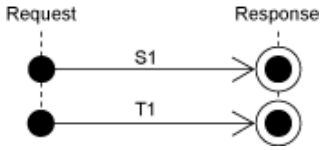
Клас File



Този интерфейс е абстракция за обекти от файловата система за документи на системата W-EPROM. Може да представлява файл или директория (подобно на *java.io.File* класа от стандартният програмен интерфейс на *Java*), като тази информация се пази в атрибута *isFolder*. При създаването на дъщерен обект чрез метода *createChild(byte type)* избираме типът - дали да е директория или файл, като самият метод е приложим само за директории, тъй като файлът представлява листо в дървото на йерархията и не може да съдържа дъщерени обекти. Имаме методи за достъп до всички дъщерни директории (*getFolders()*) и файлове (*getFiles()*), както и за намиране на техният брой съответно – *getFilesCount()* и *getFoldersCount()*.

5.2.2. Управление на цикъла заявка-отговор

В текущата имплементация за стратегията за управление на заявката (цикъл заявка-отговор) е избран модела “една сесия на една заявка” (*session-per-request*). Тук под сесия разбираме група от операции в базата данни. Това е най-широко използваният модел (Фиг 5.4).



Фигура 5.4. Модел за управление на заявката “една сесия на една заявка”

5.2.2.1. Стратегия за управление на заявката

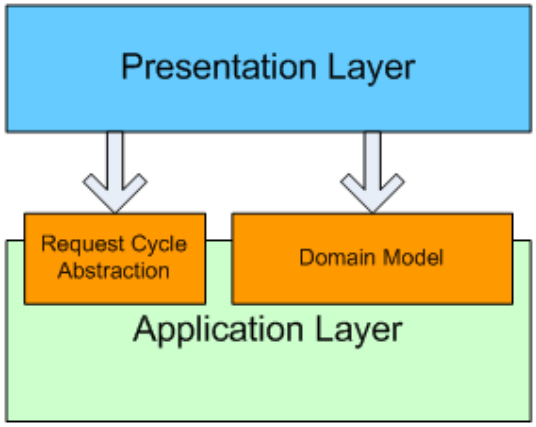
В рамките на всяка заявка всички необходими домейн обекти се презареждат наново от базата като най-често в презентационният слой се пазят само техните уникални идентификатори (ако е нужно). Също така в рамките на една

заявка може да имаме и множество транзакции в БД, като всяка от тях има точно определени граници – начало и край и освен това, при грешка, транзакцията се инвалидира (*roll-back*).

Връзката към базата данни се установява при първият опит за достъп до домейн обект(и) и се затваря автоматично в края на сесията.

5.2.2.2. Абстракция на цикъла заявка-отговор

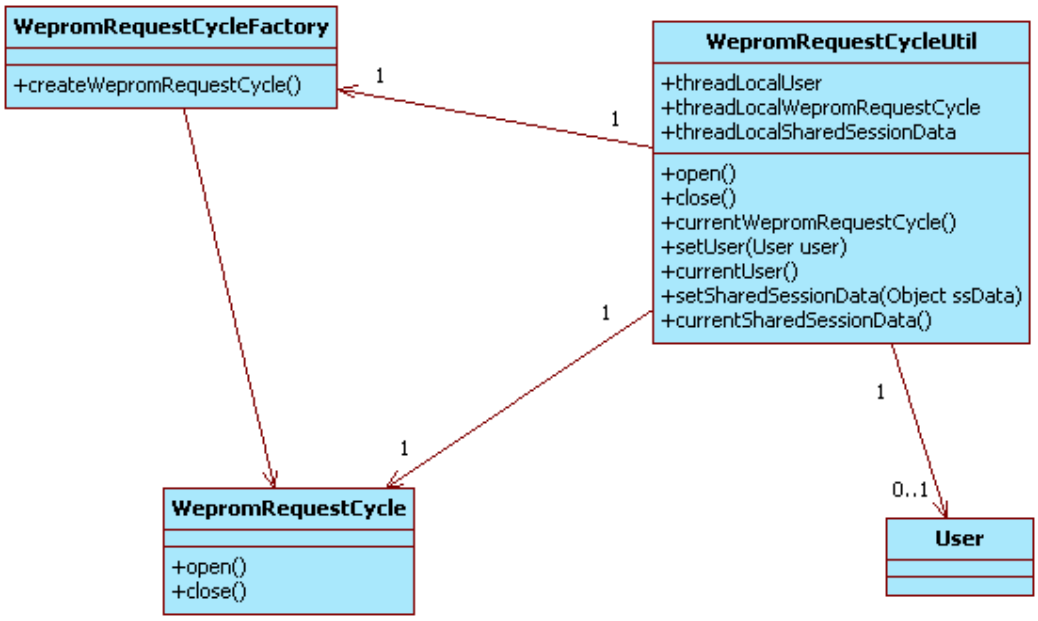
Създаваме абстракция на цикъла заявка-отговор, за да избегнем пряката зависимост на презентационния слой към имплементацията на слоя бизнес логика. Така презентационният слой има зависимост само към програмния интерфейс на обектите от домейна (*Domain Model*) и тази абстракция (*Request Cycle Abstraction*), виж Фиг. 5.5.



Фигура 5.5. Диаграма на зависимост между презентационния слой и слоят бизнес логика

По този начин замяната на имплементацията на слоя бизнес логика е лесна и може да стане прозрачно за презентационният слой. В допълнение тази абстракция с помоща на презентационният слой предоставя възможност на слоя бизнес логика да пази обекти в обхвата на една потребителска сесия.

Разгледаната абстракция на цикъла заявка-отговор е реализирана в *com.fmi.weprom.util* пакета на системата W-EPROM, чийто проект отново е представен чрез UML клас диаграма (Фиг. 5.6).



Фигура 5.6. UML клас-диаграма на *com.fmi.weprom.util* пакета

Моделът за абстракция на цикъла заявка-отговор включва:

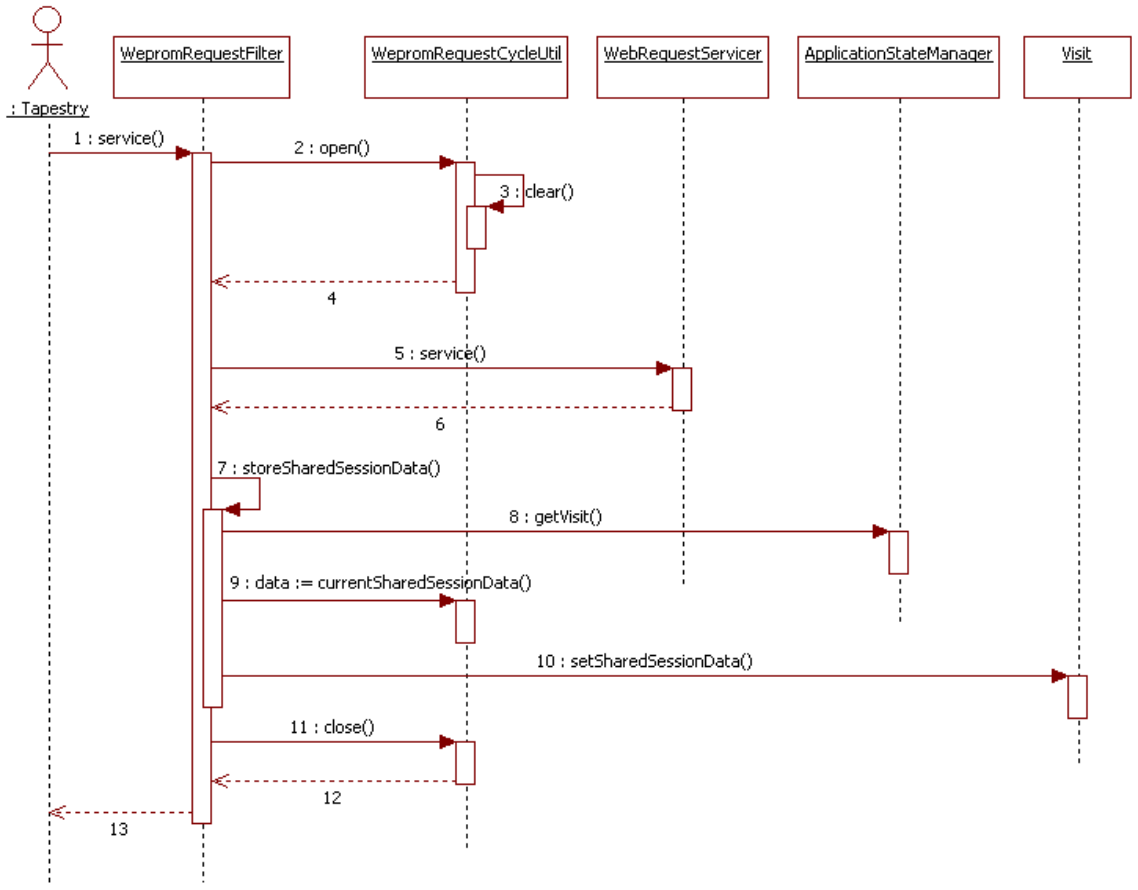
- *WepromRequestCycle* – интерфейс представляващ цикъла заявка-отговор. Всяка отделна имплементация на модела на данните може да предостави своя собствена реализация на този интерфейс и да имплементира своя конкретна логика за инициализация – в метода *open()* в началото на заявката и деинициализация в метода *close()* в края на заявката.
- *WepromRequestCycleFactory* – интерфейс представляващ фабрика за обекти от тип *WepromRequestCycle*. Съответно използваната имплементация трябва да реализира своя собствена фабрика за тези обекти. Това дава възможност да се абстрахираме от начина на конструиране на самите обекти.
- *WepromRequestCycleUtil* – помощен клас, за връзка между презентационният слой и абстракцията на цикъла заявка-отговор. Тук трябва да отбележим, че презентационният слой е този, който определя началото и края на заявката. При постъпване на нова заявка той извиква метода *open()* и в края на заявката извиква метода *close()*.

Факт е, че всяка заявка се обработва в собствена нишка и този помощен клас използва това за да прикрепи обекта представляващ самата заявката (*WepromRequestCycle*) към текущата нишка. Това е реализирано посредством специален механизъм в Java за използване на локални за нишката променливи – *Java thread-local variables* [41].

5.2.2.3. Жизнен цикъл на заявката

5.2.2.3.1. Връзка със заявката от презентационния слой

Двигателят на цикъла заявка-отговор, както вече пояснихме, е презентационният слой и тук илюстрираме начинът на пренасянето му в слоя бизнес логика използвайки горе-описаната абстракция чрез UML диаграма на последователностите (Фиг 5.7):



Фигура 5.7. UML диаграма на последователностите на връзката на заявката в слоя бизнес логика със тази в презентационният слой

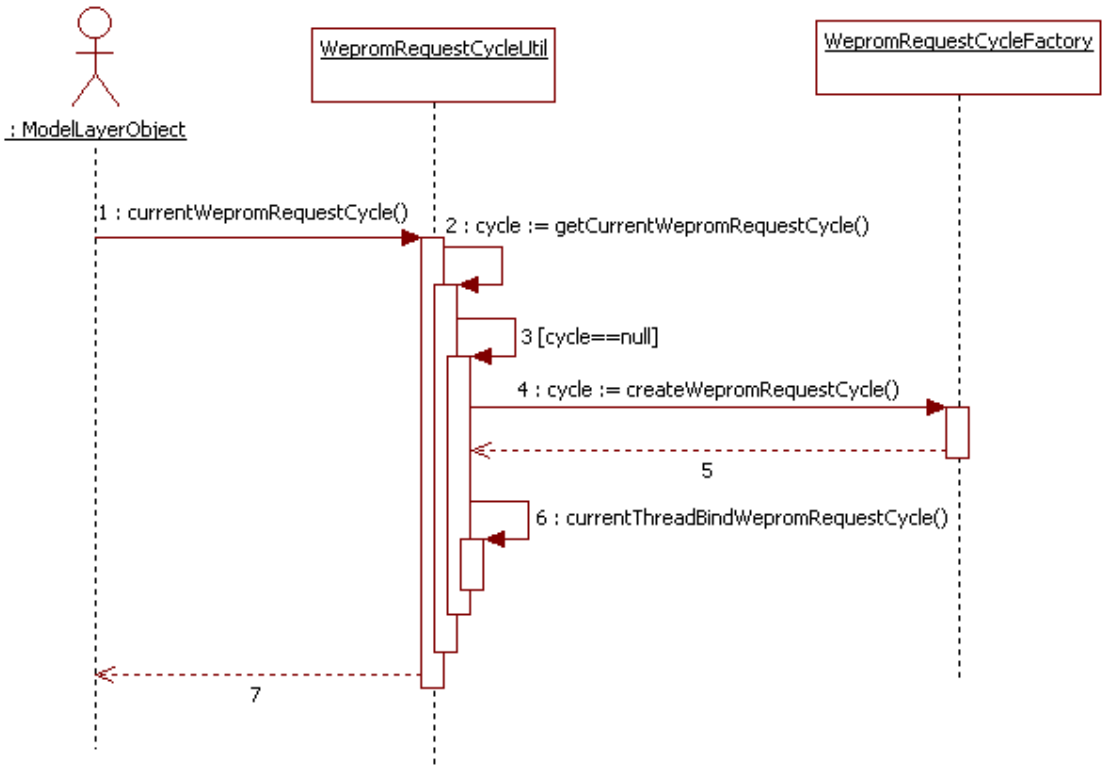
За осъществяване на връзката със заявката от презентационния слой се използва специален механизъм на Tapestry и по-точно на HiveMind платформата, а именно - дефинираме HiveMind сервиз *WebRequestServicerFilter*.

Този механизъм е подобен на сървлет филтър [42] и предоставя контрол преди и след обработване на текущата заявка от Tapestry (*5: service()*). Това, в случая се използва, за да се инициализират съответните компоненти от слоя бизнес логика (*2: open()*) и след приключване на заявката да се деинициализират (*11: close()*), както ще видим по-подробно по-долу.

Междувременно извикваме *WebRequestServicer* за да се обработи самата заявка в Tapestry рамката за приложения (*5: service()*).

5.2.2.3.2. Вземане на инстанция на текущия цикъл

Инстанция на обект представящ текущия цикъл се създава само при поискване (Фиг. 5.8). Т.е. ако имаме заявка в презентационния слой, която не изисква обект от домейна, то такава инстанция няма да бъде създадена.

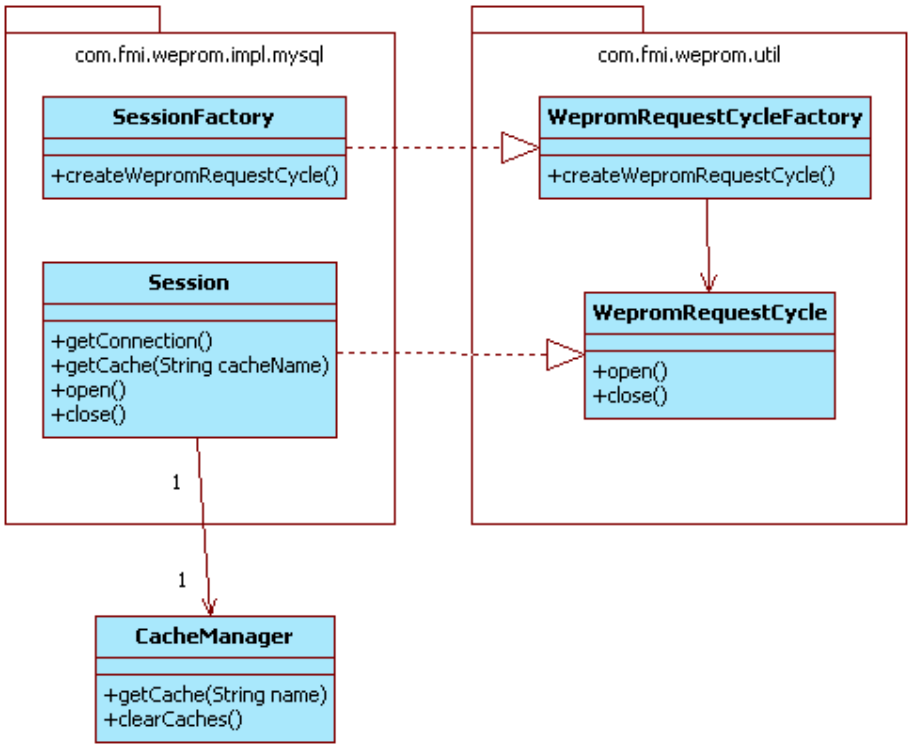


Фигура 5.8. UML диаграма на последователностите на вземане на инстанция на текущия цикъл

Инстанцията се създава посредством фабриката за обекти от тип *WepromRequestCycle*. Класът, който имплементира самата фабрика (от тип *WepromRequestCycleFactory*) е нужно да се зададе в специален за целта параметър. В текущият прототип стойността на паратеъра е:

“com.fmi.weprom.impl.mysql.SessionFactory”

Както споменахме по-рано всяка заявка се обработва в своя собствена нишка и тук, при имплементацията на абстракцията на цикъла заявка-отговор (вж. UML Клас-диаграмата от Фиг. 5.9), проверяваме ако няма прикрепена инстанция на текущата заявка[1], то я създаваме и съответно я прикрепяме към текущата нишка (*Java thread-local variables*).

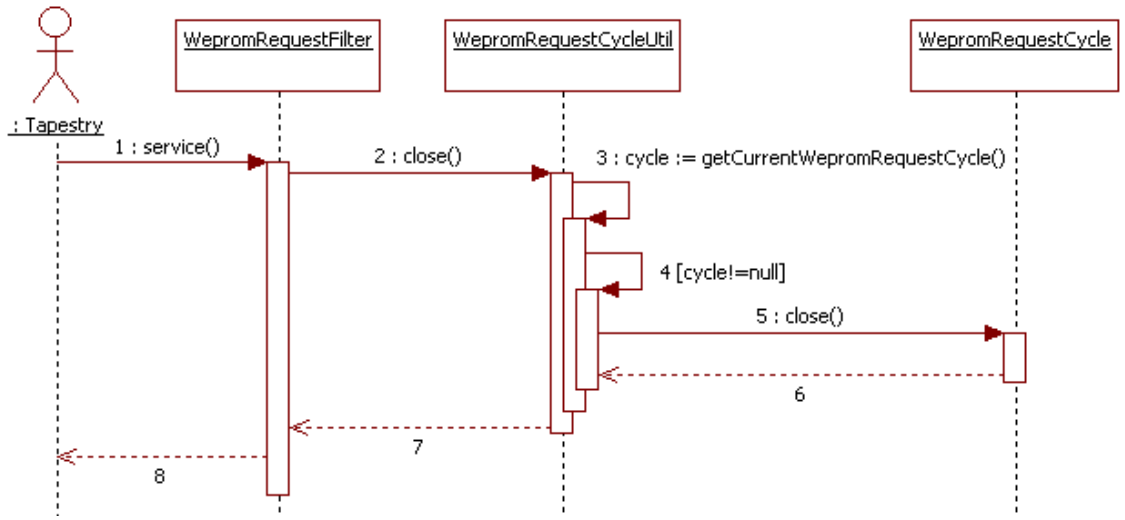


Фигура 5.9. UML Клас-диаграма на имплементацията на абстракцията на цикъла заявка-отговор

Виждаме, че класът който представлява заявката в слоя бизнес логика предоставя механизъм за получаване на връзка кам базата данни (метод *getConnection()*) и механизъм за достъпване до кеш за обектите от домейна (метод *getCache(…)*).

5.2.2.3.3. Автоматично затваряне на текущия цикъл

Деинициализацията на текущия цикъл се задейства от презентационния слой както видяхме по-горе. Методът *close()* (Фиг. 5.10) ще бъде извикан само ако в рамките на текущата заявка е била поискана поне веднъж инстанция на текущия цикъл. В противен случай затварянето на цикъла преминава без допълнителни стъпки.



Фигура 5.10. UML диаграма на последователностите на автоматично затваряне на текущия цикъл

Автоматичното затваряне на цикъла създава удобство за:

- Предоставяне на отворена връзка към базата данни от момента на първото й поискване ако има такова и автоматичното й затваряне в края на цикъла.
- Автоматично изчистване на кеша за обекти от домейна използван в обхвата на текущата заявка.

5.2.2.3.4. Предоставяне на възможност на слоя бизнес логика да запазва данни в рамките на една потребителска сесия

Както знаем потребителската сесия (*HttpSession*) е средство достъпно само за презентационния слой. В случай, че слоят бизнес логика има нужда да запазва данни в рамките на една потребителска сесия може да използва абст-

[1] Еквивалента на цикъла заявка-отговор в презентационният слой се нарича сесия в слоя биз-

ракцията от Фиг.5.6. и по-специално методите на класа *WepromRequestCycleUtil* - *setSharedSessionData(Object ssData)* и *getSharedSessionData()*. Тези методи използват локална за текущата нишка променлива за запазване на данните.

За да се осигури това удобство на слоя бизнес логика е нужно презентационният слой да взема текущите данни в края на всяка заявка и да ги запазва във *Visit* обекта, а в началото на заявката преди да започне обработването ѝ да ги прочита от *Visit* обекта и да ги установява отново чрез *setSharedSessionData(Object ssData)* методът. Първото е илюстрирано на диаграмата на последователностите от Фиг.5.7 - HiveMind сервиз *WebRequestServicerFilter* запазва данните в края на заявката (*7: storeSharedSessionData()*), а второто на диаграмата на последователностите от Фиг.5.12 (*11: - 15:*) по-долу.

5.2.3. Модел на контрола на достъпа, базиран на роли

Математически, изискванията към контрола на достъп, в повечето случаи, могат да бъдат изразени като релация *AC* (*Access Control*) между множество от потребители - *Users* и множество от права за достъп – *Permissions*:

$$AC \subseteq Users \times Permissions$$

Потребител *u* има право *p* тогава и само тогава, когато $(u, p) \in AC$. Като изключим техническият въпрос как да интегрираме тази релация в системите, така че даването на права да я запазва, по-голямо предизвикателство представлява това как по-ефективно да представим тази информация, тъй като ако директно запазваме всички двойки от вида (*u*, *p*) не води до добра производителност. При това този подход е доста „плосък” и не поддържа естествени абстракции като множества от права за достъп.

Моделът на контрол на достъпа базиран на роли *RBAC* (*Role Based Access Control*) [43] е решението на горните две ограничения. Основната идея на *RBAC* е да въведе множество от роли и да разложи релацията *AC* на други две релации *UA* (*User Assignment*) и *PA* (*User Assignment*):

$$UA \subseteq Users \times Roles, \qquad PA \subseteq Roles \times Permissions$$

Тогава релацията на контрола на достъп *AC* е композицията на горните две релации:

нес логика.

$$AC \subseteq PA \circ UA$$

С други думи *AC* дефинираме като:

$AC = \{(u, p) \in Users \times Permissions \mid \exists\ role \in Roles.\ (u, role) \in UA \wedge (role, p) \in UA\}$

Във W-EPROM системата горните две релации се пазят във таблици в базата данни както следва:

- Релацията *PA* се пази във таблицата role_permissions.
- Релацията *UA* се пази във таблицата access_rights. Като допълнение в тази таблица имаме и колона, която посочва за кое работно пространство се отнася тя. Т.е. потребителят има отделни роли във отделните проекти и групи по интереси и съответно има различни права на достъп във тях.

Самите роли и правата за достъп се пазят във отделни таблици в базата данни, съответно - roles и permissions. Също трябва да отбележим, че имаме предефинирани роли и права за достъп, но дизайнът е достатъчно гъвкав и позволява лесно разширяването им.

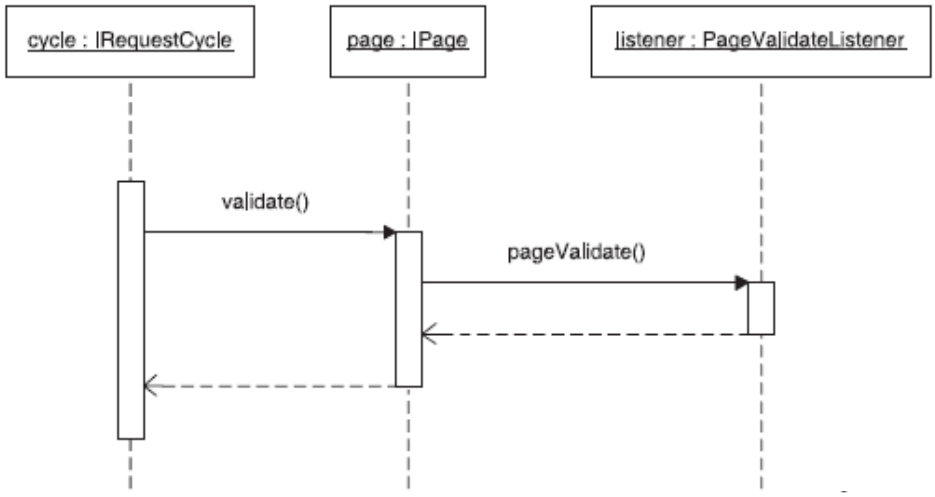
5.2.3.1. Реализация и управление на контрола на достъпа

Управлението на достъпа до обектите от домейна включва описаните в следващите подраздели механизми.

5.2.3.1.1. Прикрепяне на извикващия потребителски обект към текущата нишка

В Tapestry има механизъм за валидиране на достъпа (Фиг. 5.11) до всяка страница. За да се ползва това е достатъчно класът на страницата да имплементира специален интерфейс – *org.apache.tapestry.event. PageValidateListener*.

В прототипа всички страници от презентационният слой, които изискват логнат потребител, наследяват един и същи базов клас, който използва механизмът за валидиране – *workspace.web.pages.WEPROMBasePage*.

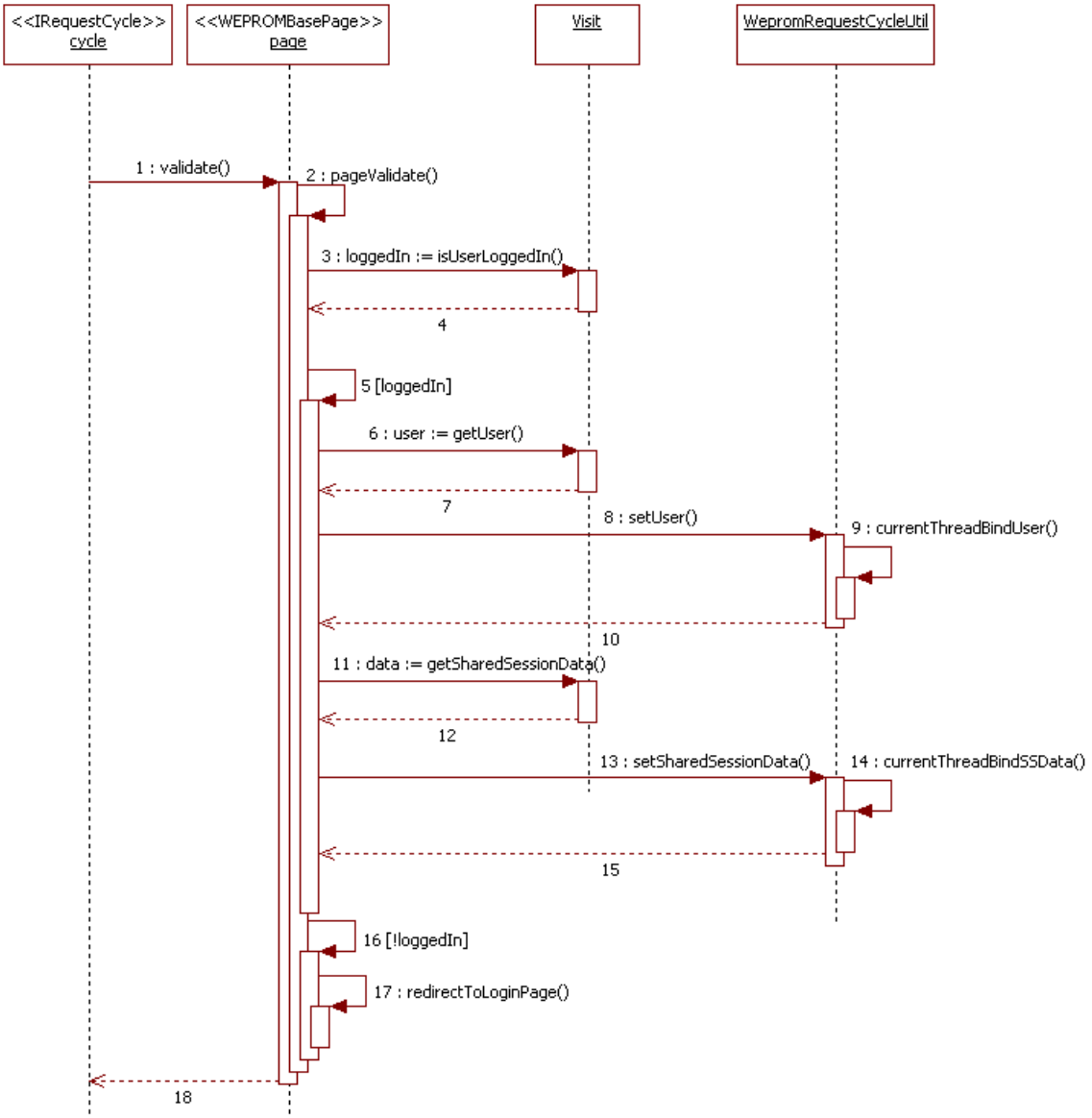


Фигура 5.11. UML Диаграма на последователностите на Tapestry механизма за валидиране на страниците

Следващата диаграма (Фиг.5.12) илюстрира как този механизъм е приложен за W-EPROM системата.

При самото валидиране се прави проверка дали има логнат потребител с помоща на *Visit* обекта (еквивалентът на *HttpSession* при Java сървлетите). Ако има такъв, то съответната му инстанция се извлича от *Visit* обекта и се прикрепя към текущата нишка. По този начин още преди да е започнала каквато и да е обработка на текущата заявка от Tapestry рамката за приложения сме обозначили (като локална за текущата нишка променлива) кой е извикващият потребител, т.е. кой потребител изпълнява заявката. Ако няма логнат потребител текущата обработка на заявката се прекратява и той бива пренасочен към страницата за логване (*redirect*).

В Tapestry страниците и компонентите имат достъп до *Visit* обекта и съответно могат да достъпят извикващият потребител от него. Така, че конструираният по-горе механизъм не е от голяма полза за презентационния слой. Слоят бизнес логика от своя страна няма достъп до нито един от обектите и класовете на презентационният слой. Това е така, понеже презентационният слой се намира над него, т.е. имаме зависимост само в едната посока (виж Фиг.5.5) и това е причината за създаването на горе-описаната конструкция, ползата от която е илюстрирана в следващата точка.



Фигура 5.12. UML Диаграма на последователностите на механизма за валидиране на страниците във W-ERPOM

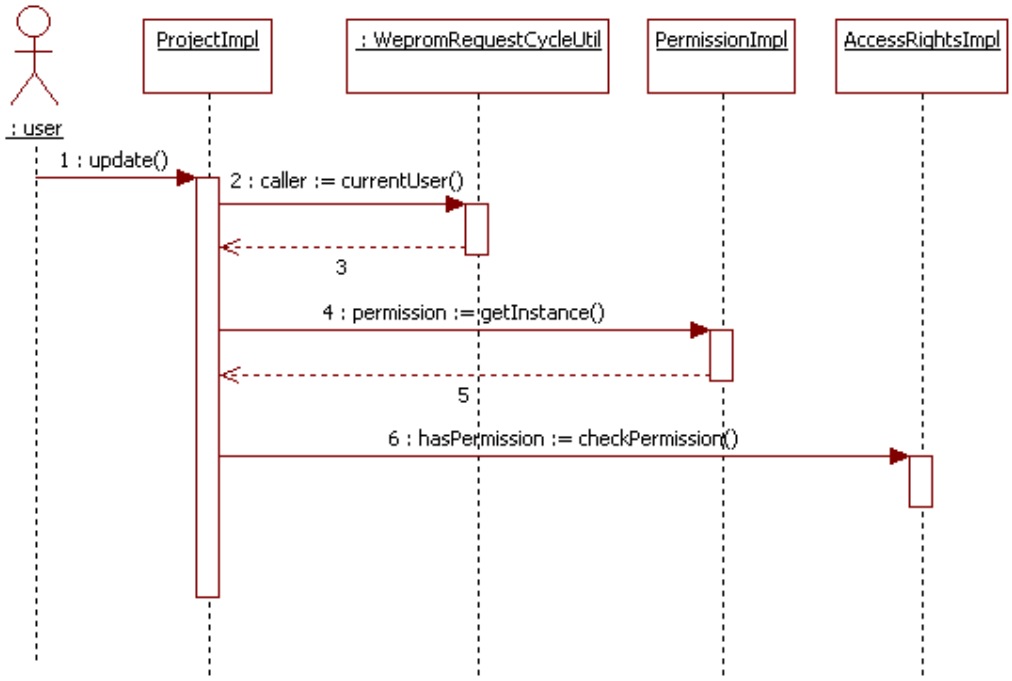
5.2.3.1.2. Проверяване правата за достъп при всяка операция с обекти от домейна

Имаме два основни типа сценарий, при които правим проверка на правата за достъп. Единият е при вземане на решение дали даден компонент от презентационния слой трябва да се визуализира или дали някоя от неговите операции е разрешена / забранена за текущият потребител.

Другият случай е при изпълнение на операция върху обект от домейна. Тук трябва да допълним, че първият вариант сам по себе си не е достатъчен, тъй

като дори и безгрешно реализиран дава възможност за неоторизирано извикване например на операция, която видимо е била забранена във графичния потребителски интерфейс. При web- приложенията това може лесно да се постигне чрез ръчно манипулиране на *URL*-адреса, т.е на параметрите на *HTTP GET* заявката. Това е и причината явно да проверяваме правата на достъп при операции с обекти от домейна във слоя бизнес логика.

На следващата диаграма (Фиг. 5.13) е илюстриран вторият случай, като за пример е взета операцията обновяване на проект (*project.update()*).



Фигура 5.13. UML Диаграма на последователностите за проверяване на правата за достъп

Методът *2: currentUser()* е статичен и това е удобство, защото може да бъде извикан от кой да е обект от модела на данните (инстанция на *ProjectImpl* класа в случая) без той да има нужда от специален указател. Това се дължи на идеята да прикрепим *User* обекта към текущата нишка обработваща текущата заявка.

След като имаме потребителския обект (*User*) и нужното право за достъп (*Permission*) можем да проверим дали то е удовлетворено в текущия контекст -

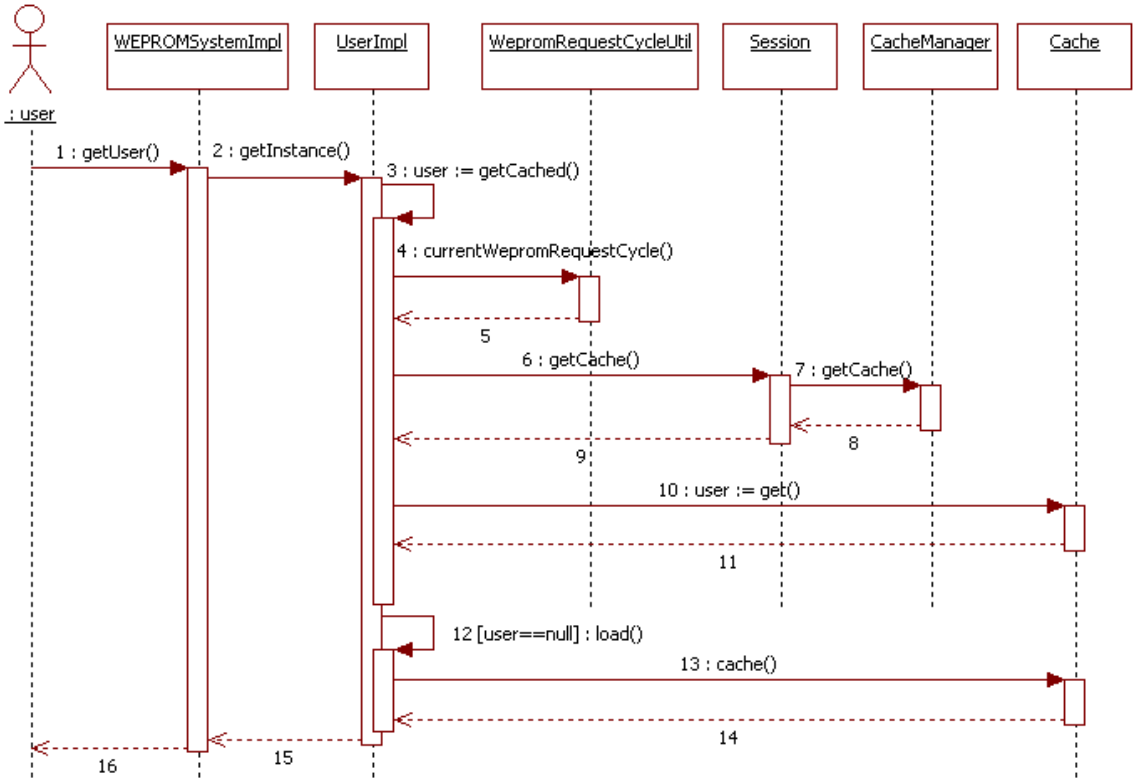
произволен проект или група по интереси и в зависимост от резултатът да изпълним или не желаната операция.

5.2.4. Кеширане на обектите от домейна

Кеширането на обектите от домейна по време на работа на системата се осъществява в два обхвата – в рамките на един цикъл заявка-отговор и в рамките на една потребителска сесия (*HttpSession*).

5.2.4.1. Кеш в рамките на един цикъл заявка-отговор

Кеширането на обектите от домейна в рамките на една заявка се използва за предотвратяване на дубликати на инстанции на обекти имащи един и същи образ в базата данни. Това в случая води и до едно друго удобство - минимизиране на броя на заявките към БД. С други думи при произволно обхождане на графа от обекти от домейна няма да имаме повторно зареждане на един и същи обект от БД за кой да е обект и съответното му дублиране като Java инстанция.



Фигура 5.14. UML диаграма на последователностите на начин на използване на кеш в рамките на една заявка

Това не изисква специален начин на използване на програмният интерфеис предоставен от слоя бизнес логика и е изцяло прозрачно за презентационният слой.

За целта имплементацията на абстракцията на цикъла на заявката (*Session* - Фиг. 5.9) предоставя метод за достъп до инстанция на обект от тип *CacheManager*. Той от своя страна предоставя раличен кеш обект за всеки отделен клас обекти от домейна. На Фиг. 5.14 чрез UML диаграма на последователностите е изобразено как се достъпва обект от тип *User* (потребител).

Първо се проверява дали вече имаме зареден обект от тип *User* със съответен уникален идентификатор в кеша (*3 : getCached(…)*). Ако това е така директно се връща кешираният обект. В противен случай се прави съответна заявка до БД и се конструира нов обект от тип *User* (*12 : load(…)*), който първо се кешира (*13 : cache()*) и тогава се връща като резултат.

За да завършим ще напомним, че на Фиг. 5.10 (*5 : close()*) е илюстрирано затварянето на цикъла заявка отговор, което в *Session* класа се използва за изчистване на кеша. Т.е. всяка следваща заявка започва и завършва с празен кеш.

5.2.4.2. Кеш в рамките на една потребителска сесия

Както по-горе видяхме абстракцията на цикъла заявка-отговор предоставя възможност на слоя бизнес логика да запазва обекти в рамките на една потребителска сесия.

Това се използва за кеширане на обекти от домейна, които не могат да се променят (*immutable objects*). Такива обекти са например ролите и правата за достъп. Това е допълнителна оптимизация за ускоряване на обработката на заявките.

5.2.5. Поведение на системата

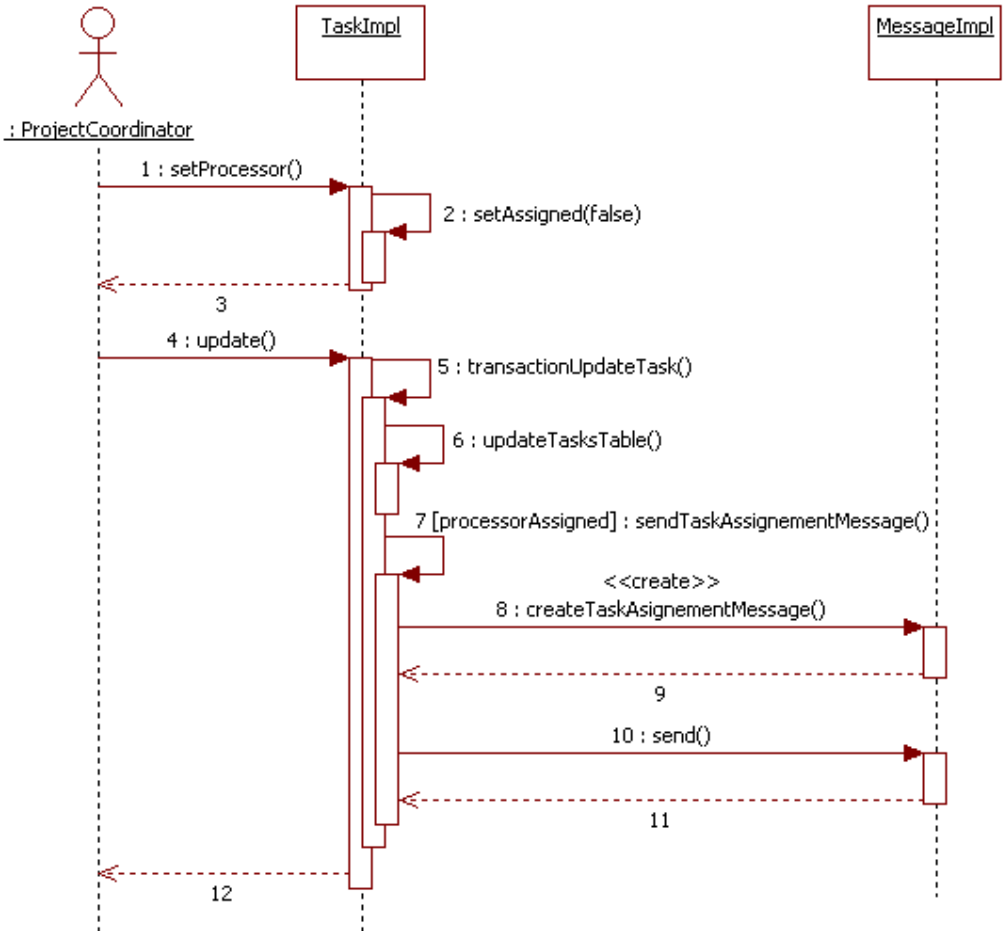
Слоят бизнес логика кодира поведението на системата за всеки един от случаите на употреба. Най-голям интерес представлява обработката на системните съобщения представено в следващите раздели.

5.2.5.1. Обработка на съобщенията

Съобщенията биват два вида – системни и текстови. Текстовите съобщения се използват за комуникация между потребителите подобно на email. Те не се нуждаят от специална обработка и просто се запазват в съответното потребителско пространство. Системните съобщения се използват за покана на потребител в проект или група по интереси, и за възлагане на задачи. Такива съобщения може да изпраща само координатор на проект или група по интереси. В следващите раздели това е разгледано по-подробно.

5.2.5.1.1. Възлагане на задача

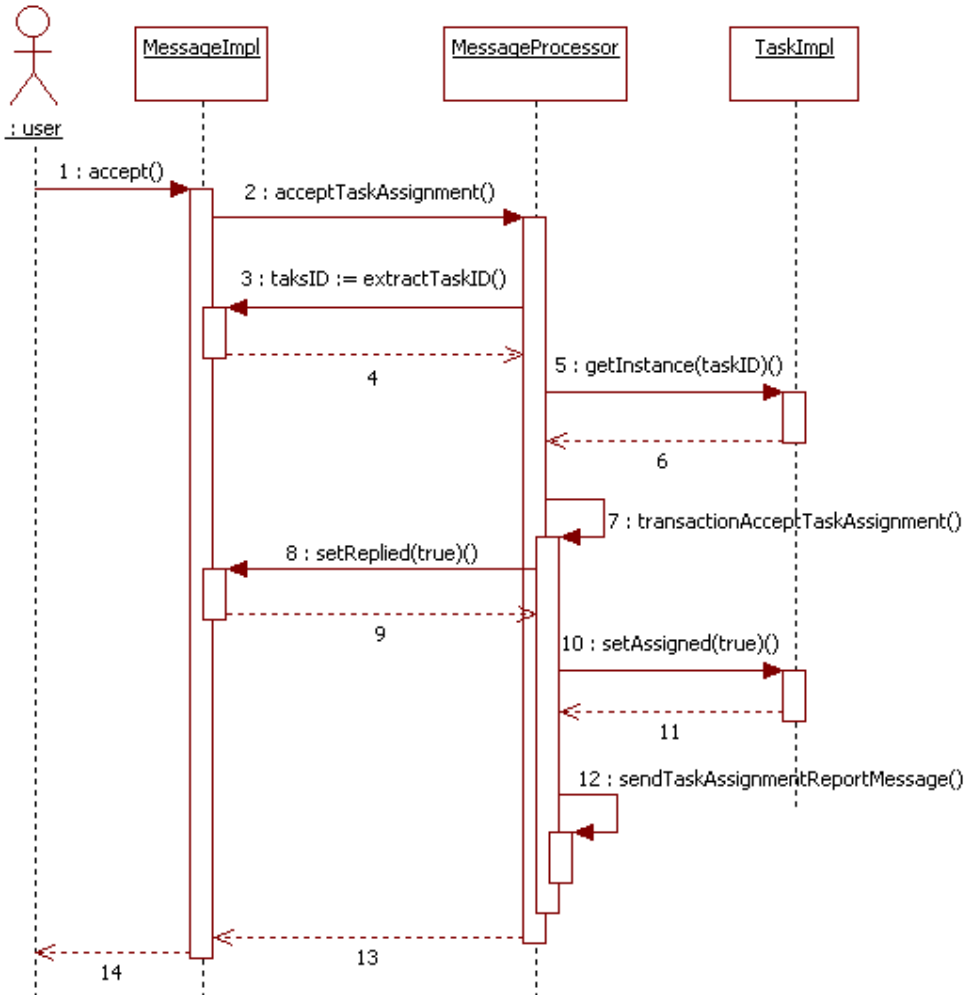
Когато координаторът на проект създаде нова задача, той може да изпрати заявка към някои от другите участници в проекта за да се заемат с



Фигура 5.15. UML диаграма на последователностите на възлагане на задача

нейното изпълнение. Това става просто ако се установи атрибутът *Processor* на задачата и тя се запази посредством *save()* метода или се обнови с *update()* метода ако вече е била създадена. На Фиг 5.15 е представен вторият вариант като UML диаграма на последователностите. При извикване на *4: update ()* започва транзакция за обновяване на задача *5: transactionUpdateTask ()*.

По време на транзакцията се обновява информацията във таблица tasks чрез *6: updateTasksTable()* след което ако имаме нов потребител назначен за изпълняването на задачата се създава ново системно съобщение *8: createTaskAssignmentMessage()* и се изпраща *10: send()*. Така приключва текущата транзакция *5:*.



Фигура 5.16. UML диаграма на последователностите на съгласие за обработка на задача

Потребителят назначен за изпълнение на задачата получава системно съобщение във входящата си кутия. В този случай презентационният слой позволява на потребителя да отговори със съгласие – бутон „Accept” или несъгласие – бутон „Decline” като съответно се извикват методите *accept()* или *decline()* на съобщението.

Да разгледаме случаят на съгласие от Фиг 5.16. Тук слоят бизнес логика обработва системното съобщението чрез метода *2: acceptTaskAssignment()* на специален за целта клас – *MessageProcessor* .

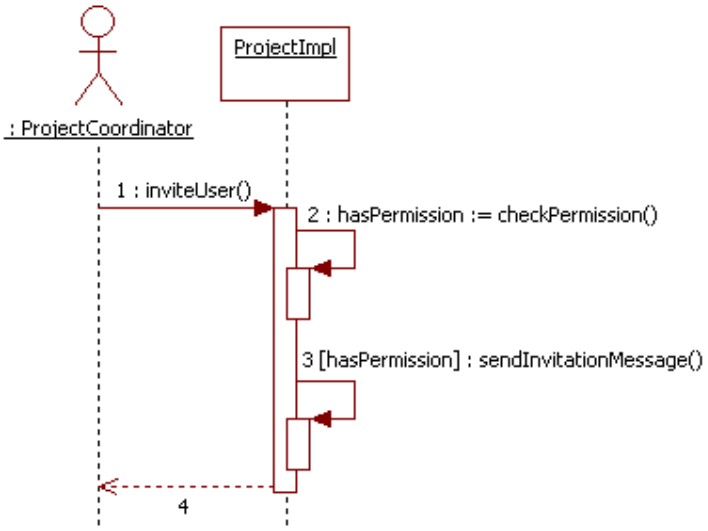
Първо се взема инстанция на съответната задача по нейния уникален идентификатор кодиран в самото съобщение – *5: getInstance(taskID)*, след което започва транзакция за приемане на задача – *7: transactionAcceptTaskAssignment()*. По време на транзакцията се правят три неща – първо се „вдига” флага *Replied* на съобщението (*8: setReplied(true)*) за да не може да се направи повторен отговор – *accept()* или *decline()*. Второ, „вдига” се флага *Assigned* на задачата (*10: setAssigned(true)*) за да се обозначи, че назначеният потребител е приел задачата за изпълнение. И трето, изпраща се автоматично текстово съобщение от името на изпълнителя на задачата, че е приел задачата за изпълнение (*12: sendTaskAssignmentRepotrMessage()*).

По подобен начин се извършва и обработката при отказ за изпълнение на задача – *decline()*.

5.2.5.1.2. Покана за участие в проект

Всеки координатор на даден проект има право да кани нови потребители за участие в него. Това става посредством метод *1: inviteUser(User u)* (виж Фиг.5.17). Първо се проверяват правата на извикващият потребител и по-специално се проверява дали има право на достъп “*project.user.roles.edit*” (*2: checkPermission()*). Ако той има това право се създава и изпраща ново системно съобщение до потребител *u* (*3: sendInvitationMessage()*). При това потребител *u* го получава във входящата си кутия и може да отговори отново със съгласие или несъгласие.

Ще разгледаме по-интересният от двата отговора – този на съгласие (Фиг.5.18). Първо се взема инстанция на съответният проект по неговия уникален идентификатор кодиран в самото съобщение – *5: getInstance(projectID)*.

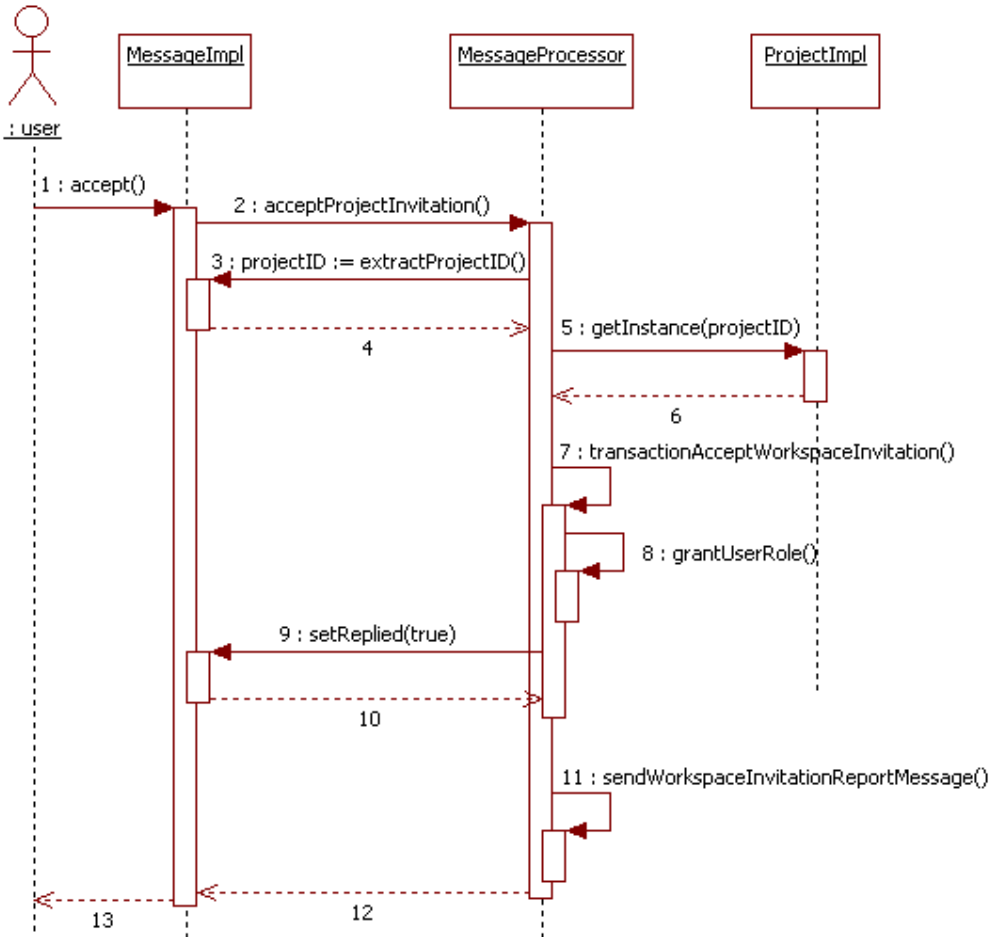


Фигура 5.17. UML диаграма на последователностите на поканване на потребител за участие в проект

След което започва транзакция *7: transactionAcceptWorkspaceInvitation()*. По време на транзакцията се дава роля *ProjectParticipant* на потребителя (*8: grantUserRole()*) и се „вдига" флага *Replied* на съобщението (9*: setReplied(true)*) за да не може да се направи повторен отговор – *accept()* или *decline()*.

След приключване на транзакцията се изпраща съобщение рапорт на изпращача на поканата за даване на информация дали той е приел или отказал. Тъй като това съобщение е просто информативно, то не е включено в транзакцията и при грешка може и да не бъде изпратено успешно, но това няма да попречи на работата на потребителите или на състоянието на системата.

При отказ на потребителя да участва в проекта съобщението се обработва по сходен начин, с тази разлика, че се пропуска стъпката при която му се дава роля, т.е. той няма да получи участие в проекта.



Фигура 5.18. UML диаграма на последователностите на приемане на покана за участие в проект

5.2.5.1.3. Покана за участие в група по интереси

Поканата за участие в група по интереси може да бъде изпратена само от някой от нейните координатори (потребителите с роля *CommunityCoordinator*) и се обработва аналогично на поканата за проект, с тази разлика че при съгласие се дава роля *CommunityParticipant* на потребителя.

5.2.6. Запазване на обектите от домейна в базата данни

Текущата имплементация е изградена на базата на *Java Database Connectivity (JDBC)* [38] – стандартният програмен интерфейс за връзка с бази данни в Java.

Най-общо всяка таблица от слоя база данни има съответен клас, който се грижи за прехвърлянето на инфомрацията от и в БД, който има три основни ме-

тода – *save()*, *update(), delete()* и *load()*. За създаване на временна инстанция имаме съответен *create()* метод, в клас който е в релация с дадения. Например методът за създаване на *Message* инстанция се намира в *UserWorkspace* класа. При създаване на нова инстанция нямаме операция с БД и атрибутите на обекта са празни. Затова обикновено след създаването установяваме желаните стойности на атрибутите и извикваме методът *save()*. Тогава се взема текущата за заявката връзка с базата данни и се изпълнява съответната SQL заявка. В някой случаи се налага да се изпълнят няколко последователни заявки в една обособена транзация. Например при създаване на нов проект – *project.save()* първо се добавя съответен ред в таблицата master (Таблица 5.1), автоматично се дава роля ProjectCoordinator на притежателя на проекта (ред в таблицата access_rights (Таблица 5.6)) и накрая се попълва допълнителната информация за проекта, като ред в таблицата workspaces (Таблица 5.7).

6. Тестване и внедряване

Често срещан подход [3] при тестване на софтуерни приложения е то да се извършва от независима група „тестери”, след като системата е разработена, но преди тя да бъде въведена в експлоатация. Друга практика също се прилага – тестването започва заедно с етапа на разработка и продължава до неговия край. Съществува и трети вид практика, според която тестови последователности се разработват едва, когато заявките за поддръжка рязко се повишат.

Като контраст на тези подходи, може да се спомене най-съвременният подход [30], който се налага все повече заедно с развитието на новите софтуерни технологии, наричани „софтуерни технологии, управлявани от тестването” (*test-driven software development*). При тези технологии, тестовите сценарии се съставят предварително от програмиста и след разработването на кода, той се подлага на тестване, като последователностите постоянно се актуализират.

Именно технологията „управлявана от тестването” бе приложена при разработването и тестването на W-EPROM.

В завършващ етап от реализацията на всяка част от системата, на базата на случаите на употреба, описани в Гл. 3 на дипломната работа бяха дефинирани многобройни тестове за всяка една от функционалностите на системата. Чрез тези тестови последователности, системата бе подложена на тестване, получените резултати от проверката на всички случаи бяха оценени и използвани, за подобряване на качеството ѝ.

С цел да бъдат обхванати всички случаи за проверка на правилното функциониране и качеството на системата, бяха разработени три групи тестови сценарии – за т. нар. „тестване от тип черна кутия”, „тестване от тип бяла кутия”, „тестване от тип сива кутия”. Те от своя страна отразяват съответно три гледни точки:

- “тестване от тип черна кутия” – външните качества на тествания софтуерен модул;
- „тестване от тип бяла кутия” – вътрешните качества;
- „тестване от тип сива кутия” – междинен подход при тестване, при който е позволено да се манипулира директно с тествания модул и който се прилага например за тестване на бази данни и заявки към тях.

Като допълнение, при създаването на едно от приложенията – „Ръководство за потребителя” отново бяха реализирани множество тестови случаи, засягащи:

- валидацията на входните данни;
- идентификацията на потребителите;
- оторизацията на потребителските права;
- цялостното коректно функциониране на услугите на системата и правилното управление на данните в нея.

По този начин последователно бяха осъществени първите три нива от етапа на тестване, а именно, тестване на компонентите, тестване на интеграцията на компонентите и цялостно тестване на функциите и услугите на системата W-EPROM. Така в процеса на разработката, всяка нейна следваща версия бе подобрявана.

За осъществяване на четвъртото ниво от етапа на тестване – тестване за одобрение от потребителя, системата W-EPROM е внедрена за пробно ползване на локален сървър без публичен достъп в Пловдивския Университет „Паисий Хилендарски”. Правят се проби за нейното използване за нуждите по управлението и реализацията на два реални проекта:

- Национален проект към НФНИ, „Съвременни методи, средства и технологии в диагностиката, консултирането и обучението на лица с увреждания”, с участието на четири партньорски организации, между които ПУ "П.Хилендарски" и СУ "Св. Кл. Охридски" и
- Международен проект по Програма SOCRATES за образование и култура на ЕС, 116684-CP-1-2004-1-HU-MINERVA-M „e-Taster, кратки, безплатни, онлайн курсове, за международно разпространение на различни езици”, с участието на осем партньорски организации от различни страни, между които ПУ "П.Хилендарски".

До момента получените резултати и отзиви от това пробно внедряване са силно положителни.

W-EPROM, също така е достъпна за свободно разглеждане и тестване на адрес http://85.187.184.114:8080/weprom/app.

Очаква се системата W-EPROM да бъде внедрена и в Софийски Университет „Св. Климент Охридски”, Факултет по математика и информатика, катедра „Информационни технологии”, CIST (Centre of Information Society Technologies).

7. Заключение

В началото на работата по дипломния тезис желанията и плановете за реализация на една цялостна, гъвкава, многофункционална и удобна за работа web-базирана система за съвместно управление на проекти бяха много по-амбициозни, мащабни и смели. След един достатъчно дълъг период на усилена работа (прототипът надхвърля общо с [52] над 17000 реда Java програмен код), сблъсък с конкретни технически трудности и трезво оценяване на сложността и обема на планираните задачи, обаче, се установи, че тези планове надхвърляха в доста голяма степен обема на дипломна работа.

Въпреки това в представяната разработка са постигнати доста на брой и качествени резултати.

Основните **приноси** на дипломанта са:

- проучена е предметната област, свързана с разработката и е направен изчерпателен обзор и критичен анализ на нейното състояние;
- анализирани са специфичните нужди от софтуер за планиране и управление на проекти в областта на висшето образование и научните изследвания, които се разработват на колективна основа и на принципите на самоорганизирането;
- уточнени са общите изисквания към програмната система, обект на работата;
- предложен е идеен проект на конкретна програмна система, наречена W-EPROM;
- направено е описание на тълкованията на специализираните термини от речника на W-EPROM, т.е. дефиниран е смисъла на метафорите, с които тя си служи;
- проектирана е общата функционалност на системата на принципа на разпределяне на отговорностите по осъществяване на отделните функции между обектите на системата;
- уточнени са актьорите и са проектирани случаите на употреба (Use Case) на софтуерното приложение, въз основа на бизнес изискванията;

- проучени са съвременните подходи при разработване на многослойни Web-приложения и използваните технологии при реализация на слоя база данни и слоя бизнес логика;
- проектирана е трислойната логическа архитектура на W-EPROM;
- реализирани са слоят база данни и слоят бизнес логика на системата, което включва:
  - проект на физическата схема на базата данни ;
  - проект на модела на данните (обектите от домейна);
  - средства за управление на цикъла заявка-отговор;
  - проект на модела на контрола на достъпа;
  - средства за кеширане на обектите от домейна;
- проучени са съществуващите подходи за тестване на софтуерни приложения;
- избрана е подходяща технология, която се прилага при разработването и тестването на W-EPROM, а именно – „управлявана от тестването";
- осъществено е тестване на системата на три нива – тестване на компонентите, тестване на интеграцията на компонентите и цялостно тестване на функциите и услугите й;
- внедряване на W-EPROM за пробно ползване на локален сървър.

При реализацията на настоящата дипломна работа е постигната поставената основна цел. Разработената система за управление на проекти предлага голяма част от основните възможности, услуги и средства за цялостно управление, проследяване и отчитане на проекти, сформиране на проектен колектив, осъществяване на пълноценна комуникация и работа върху общо съдържание, обмен на мнения и идеи и сътрудничество. В допълнение на това, тя дава възможност да се сформират колективи и групи от хора с не толкова строга организационна структура, а които просто имат общи интереси и желаят да обсъждат разнообразни теми.

Въпреки това, поради широкия диапазон от приложни области, където W-EPROM би могла да бъде използвана и потребителски потребности, които би могла да покрие, системата може да бъде развивана в много направления.

Насоки за възможно развитие на работата:

- да се обогатят ролите, съответстващи на активните обекти на системата, например с потребителска група;
- да се разширят възможностите за оперативно планиране на проекти (някои от тях са описани в идейния проект на системата);
- да се обогати поддръжката на избор на средства и услуги за потребителите;
- да се допълнят операциите за публикуване и манипулиране със съдържанието (документи, папки);
- разработване на възможности за финансово планиране, проследяване и отчитане на проекти и задачи;
- осигуряване на допълнителни интерфейси за достъп до средата, например, чрез мобилни устройства и др.
- осигуряване на преносимост
- внедряване на W-EPROM за реално ползване.

8. Използвана литература

Основни източници:

1. Ajeebo: Web based software and tools directory (2006) How Project Management Software can help you better manage your projects. In http://www.ajeebo.com/index.php?page=Article:ViewPage&aid=1859.
2. Apache Software Foundation (2006) Wellcome to Tapestry. In http://tapestry.apache.org/.
3. Black R (2002) *Managing the Testing Process.* Second Edition: John Wiley and Sons, ISBN 0-471-22398-0.
4. Eagle M (2004) Wiring Your Web Application with Open Source Java: O'Reilly Media, Inc., http://www.onjava.com/pub/a/onjava/2004/04/07/wiringwebapps.html.
5. EMC Corporation (2006) EMC Corporation Web Site. In http://software.emc.com/.
6. Fraunhofer FIT and OrbiTeam Software GmbH (2006) BSCW, 1995-2005. In http://bscw.fit.fraunhofer.de/.
7. Groupee Nation Community (2006) EVE Home Page. In http://eve.groupee.com/.
8. Howard M Lewis Ship (2004) Tapestry in Action: Manning Publications Co, NY, ISBN: 1932394117.
9. Hughes B& Cotterell M (2005) Software Project Management Fourth Edition: McGraw-Hill.
10. IBM (2006) IBM Lotus QuickPlace. In http://www-142.ibm.com/software/sw-lotus/products/product3.nsf/wdocs/ltwhome/.
11. Inquisite Inc. (2006) Inquisite Inc. Web Site. In http://www.inquisite.com/.
12. Langham M (2004) ProjectPlace, Gets IT Documents In Focus. In Bloor Research: Sep. 14.
13. LukeW Interface Designs (2005) Web Application Solutions: A Designer's Guide. In http://www.lukew.com/ff/entry.asp?170.
14. Max Wideman R (2001) The Future of Project Management: AEW Services, Vancouver, Canada.
15. Max Wideman R (2006) Managing Successful Programmes: AEW Services, Vancouver, BC, Canada.
16. Microsoft Corporation (2005) Microsoft Web Site: Microsoft, Groove Networks to Combine Forces to Create Anytime, Anywhere Collaboration. In http://www.microsoft.com/presspass/features/2005/mar05/03-10GrooveQA.mspx.
17. Microsoft Corporation (2006) MS Live Meeting. In http://www.microsoft.com/uc/livemeeting/default.mspx.
18. Microsoft Corporation (2006) MS Project. In http://www.microsoft.com/office/project/prodinfo/default.mspx.

19. Microsoft Corporation (2006) Microsoft Web Site: MS Exchange. In http://www.microsoft.com/.
20. Microsoft Corporation (2006) Microsoft Web Site: MS NetMeeting. In http://www.microsoft.com/windows/NetMeeting/default.ASP.
21. Open Text Corporation (2006) FirstClass Division of Open Text Corporation. In http://www.centrinity.com/.
22. Salmons J Castellucci F (2000) Specification Writing for Web-based Project Planning Software. In http://sohodojo.com/techsig/project-planning-project.html.
23. Schrage M (2000) No More Teams! Mastering the Dynamics of Creative Collaboration: ISBN 0-385-47603-5, USA.
24. Scott B (2005) The Art of Project Management: O'Reilly Media.
25. Sigmer Technologies Ltd (2006) Boomerang System. In http://www.boomerang.uk.com/.
26. Stanhope P (2002) Get in the Groove: Building Tools and Peer-to-Peer Solutions with the Groove Platform: Hungry Minds, New York.
27. Web Crossing Inc. (2006) Web Crossing Home Page. In http://www.webcrossing.com.
28. WebEx Communications Inc. (2005) WebEx Press Release. In http://ir.webex.com/phoenix.zhtml?c=121018&p=irol-newsArticle&ID=709701&highlight=.
29. Wikipedia The Free Encyclopedia (2006) FirstClass. In http://en.wikipedia.org/wiki/FirstClass.
30. Wikipedia The Free Encyclopedia (2006) Software testing. In http://en.wikipedia.org/wiki/Software_testing.
31. World Wide Web Consortium (2006) HyperText Markup Language Home Page. In www.w3.org/MarkUp/.
32. Zeldman J(2006) Designing with Web Standards (Paperback): New Riders.
33. Zengobi, Inc. (2006) Curio User Profiles. In Creative Software for Creative Thinkers, http://www.zengobi.com/products/curio/profiles.html
34. http://files.projectplace.com/english/demo/document_management/intro_dm/uk_doar.html

Web- ръководства:

35. www.w3schools.com/sql/default.asp
36. http://www.ognl.org/
37. http://jakarta.apache.org/hivemind/
38. http://java.sun.com/javase/technologies/database/index.jsp
39. http://www.w3.org/Style/CSS/
40. http://www.eclipse.org/
41. http://www-128.ibm.com/developerworks/library/j-threads3.html

42. http://java.sun.com/products/servlet/Filters.html
43. David Basin, J¨urgen Doser, Torsten Lodderstedt;2005; Model Driven Security: from UML Models to Access Control Infrastructures
44. http://www.mysql.com/
45. http://www.oracle.com/
46. http://www.microsoft.com/sql/
47. http://www.ibm.com/db2
48. http://www.linux.org/
49. http://www.perl.org/
50. http://httpd.apache.org/
51. http://www.postgresql.org/

Дипломна работа:

52. Донева Н.(2006) „Уеб-базирана система за съвместно управление на проекти- презентационен слой”

9. Приложения

9.1. Ръководство на потребителя

9.1.1. Как да започнем?

Преди всичко, бихме искали да препоръчаме ползването на W-EPROM с Mozilla Firefox браузъра.

Когато за първи път достъпите началната страница на W-EPROM, Вие ще имате достъп до цялото публично пространство (Фиг. 9.1) на системата като “**гост**”.

Фигура 9.1. Публично пространство на W-EPROM

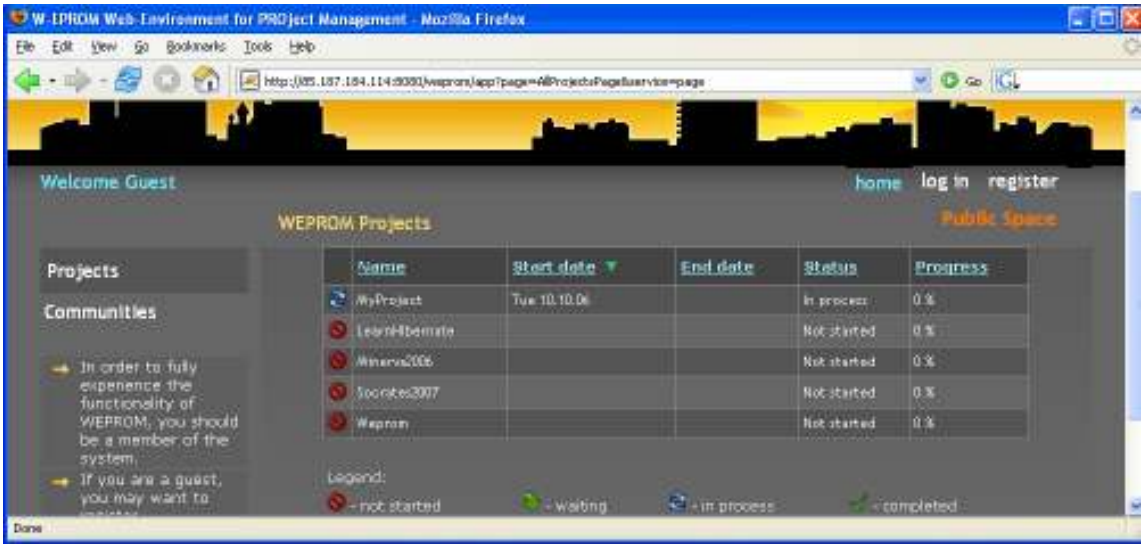
С тази роля можете единствено да разгледате:

а) Началната страница на системата (Фиг. 9.2);

б) Проектите в системата (Фиг. 9.3);

в) Групите по интереси[2] в системата (Фиг. 9.4).

Фигура 9.2. Начална страница на W-EPROM



Фигура 9.3. Проекти в системата W-EPROM

[2] **Група по интереси** е специален вид потребителска група, която допълнително се характеризира с *тематика*, от която се интересува групата. Включването на нови *членове* става само по покана на координатора на групата, а основна й папка е видима за всички, но само *членовете* й имат достъп до нейното съдържание.

Фигура 9.4. Групи по интереси в WEPROM

9.1.2. Регистрация

За да започнете пълноценно да използвате WEPROM, Вие трябва да се регистрирате в системата. След кратка регистрационна процедура (Фиг. 9.5), Вие получавате ролята “**потребител**”, който се идентифицира от:

- потребителско име (уникално);
- парола;
- име;
- фамилия;
- email адрес;
- телефон;
- допълнителен коментар.

Уверете се, че сте попълнили всички задължителни полета, обозначени със звездички (*) и съдържанието на Password и Retype password е еднакво.

След като попълните данните си, продължете с натискане на бутона Register (Фиг. 9.5).

Фигура 9.5. Регистрационна процедура

На екрана ще се появи утвърдително съобщение (Фиг. 9.6), ако данните са коректно въведени и потребителското име е уникално, т.е все още не е заето от друг потребител.

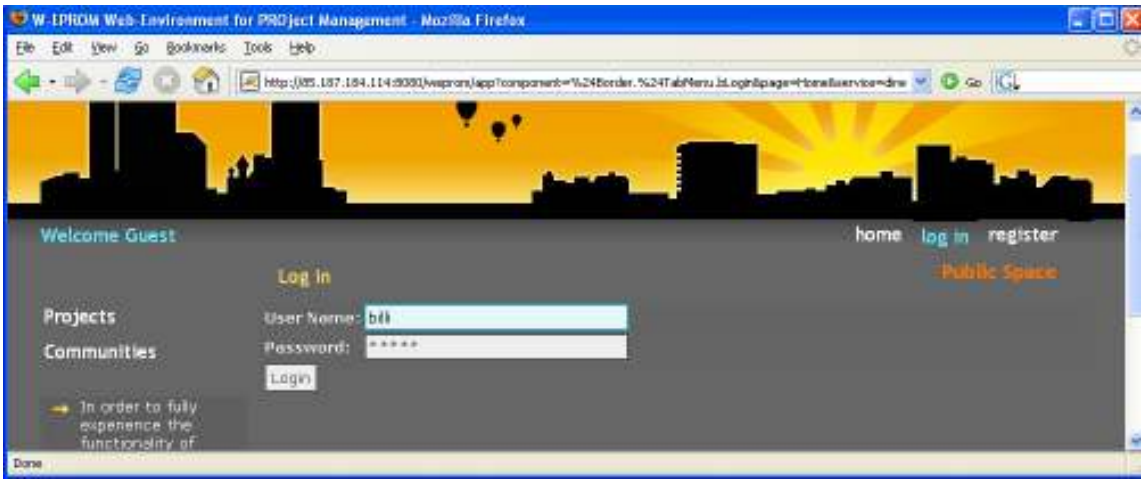
Фигура 9.6. Утвърдително съобщение при регистрация

Възможно е и да съществува вече такова потребителско име или някои от данните да са некоректно зададени. В такъв случай на екрана се извежда съобщение за грешка (Фиг. 9.7).

Фигура 9.7. Съобщение за грешка при регистрация

9.1.3. Влизане в системата WEPROM

Вие вече сте разпознаван от системата като регистриран потребител и следователно можете да влезете (да се „логнете”) в нея.



Фигура 9.8. Влизане в системата WEPROM

За това е нужно само да попълните потребителското име и паролата си и да натиснете бутона Login (Фиг. 9.8).

9.1.4. Моето потребителско пространство.

С получаването на роля „**потребител**” при регистрацията си, Вие придобивате следните **права**:

- притежаване и разглеждане на собствено работно пространство;
- създаване на проект;
- създаване на група по интереси;
- изпращане, получаване и управление на съобщения;
- управление на контакти;
- притежаване и разглеждане на собствен календар;
- управление на паметки;
- управление на документи и папки.

Имайки предвид горното, непосредствено след „логин”, Вие влизате в своето лично потребителско работно пространство („My workspace”). Вашето потребителско пространство (Фиг. 9.9) е достъпно само от Вас самите.

Личното работно пространство съдържа:

- всички проекти[3], на които сте член;
- всички групи по интереси, на които сте член;
- всички задачи, на които сте създател или изпълнител;
- Вашите съобщения, били те входящи/ изходящи, текстови/системни;
- Вашите контакти;
- личният Ви календар и паметки;
- потребителската Ви папка.

[3] **Проект** е вид потребителска група, чиито членове съставляват колектива за осъществяване на определен проект. Поради това основната папка на проекта подпомага някои специфични за такива колективи дейности, като представяне на проекта (визитка, заглавие). Най-съществена, обаче е възможността за времево планиране и управление на проекта (чрез работен план, работни пакети, задачи и задаване на времеви условия и зависимости между тях).

Фигура 9.9. Лично потребителско работно пространство

Единствено от вашето лично пространство, вие можете да достъпите другите работни пространства, на които сте член.

9.1.4.1. Проекти

Първоначално Вашият списък с проекти е празен (Фиг. 9.10). Проект се добавя към него, ако:

а) Вие сте създали нов, в който автоматично сте получили роля „**координатор на проект**” или

б) някой координатор на проект Ви е поканил за членство и Вие сте приели.

Фигура 9.10. Празен списък с проектите на потребителя

Роли в проект

Съществуват две потребителски роли във всеки проект – „**участник в проект**”, обозначен с иконата  и „**координатор на проект**”, обозначен с  . Всяка от тях се характеризира с определени права. Важно е да спомена отново, че при създаване на проект, Вие автоматично получавате и двете роли. Другата особеност е, че „**координатор на проект**” включва в себе си „**участник в проект**”, т.е. всички права, присъщи на *участника*, са преписани и на *координатора*.

Правата на **координатора** са:

- редактиране, изтриване на проект;
- редактиране на ролите на потребителите на проекта;
- създаване, редактиране, изтриване на паметка от календара на проекта;
- създаване, редактиране, изтриване на задача от работния пакет;
- назначаване на задачи;
- покана на нови членове;
- отстраняване на членове.

Правата на **участника** включват:

- разглеждане на проекта;
- разглеждане на паметка от календара на проекта;
- разглеждане на задача от работния пакет;
- редактиране, само ако участикът е изпълнител, на задача от работния пакет;
- създаване, разглеждане, изтриване на документ в проектна папка;
- създаване, разглеждане, изтриване на проектна папка.

Създаване на проект

Функцията се достига от лявото меню Projects/New Project, когато потребителят е „логнат”.

При създаването на нов проект (Фиг. 9.11), няколко характеристики могат да се зададат:

- име;
- описание;

- статус (нестартиран, в процес на изпълнение, процес на на изчакване, завършен);
- начална дата;
- крайна дата;
- степента на завършеност на проекта в проценти.

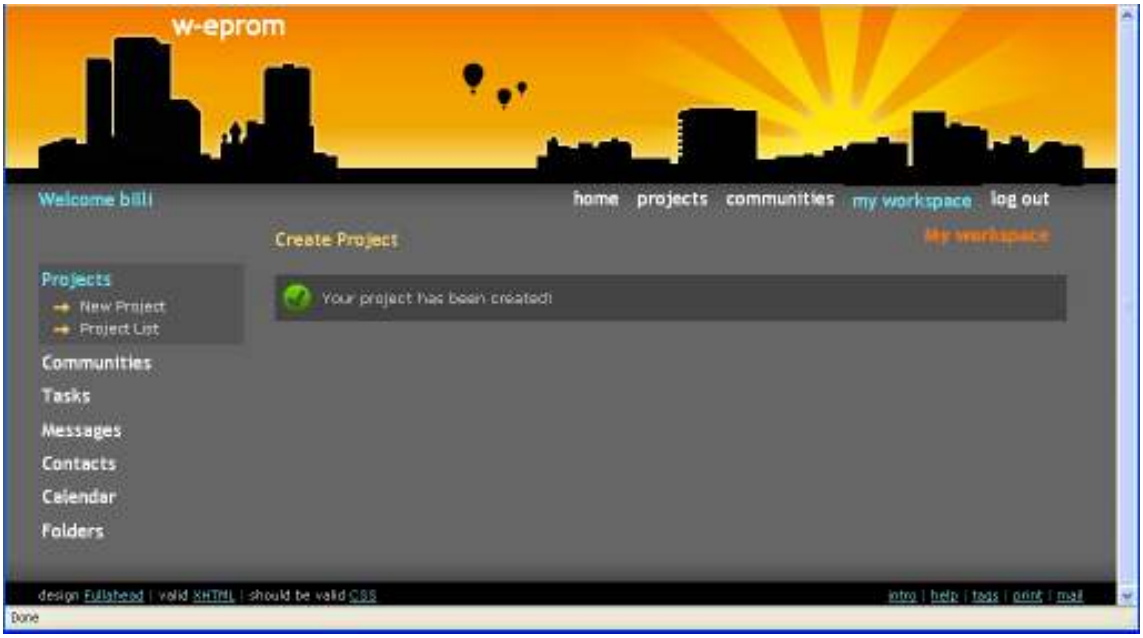
Не забравяйте да зададете име и начална дата на проекта.

След като въведете данните, натиснете бутона Create.

Фигура 9.11. Създаването на нов проект

Има възможност да сте дублирали името на проекта с вече съществуващо. Тогава на екрана се появява съобщение за грешка.

Ако транзакцията е успешна, на екрана се появява утвърдително съобщение (Фиг. 9.12).



Фигура 9.12. Утвърдително съобщение при създаване на проект

Когато се върнете на списъка с проекти, новият проект вече е изброен.

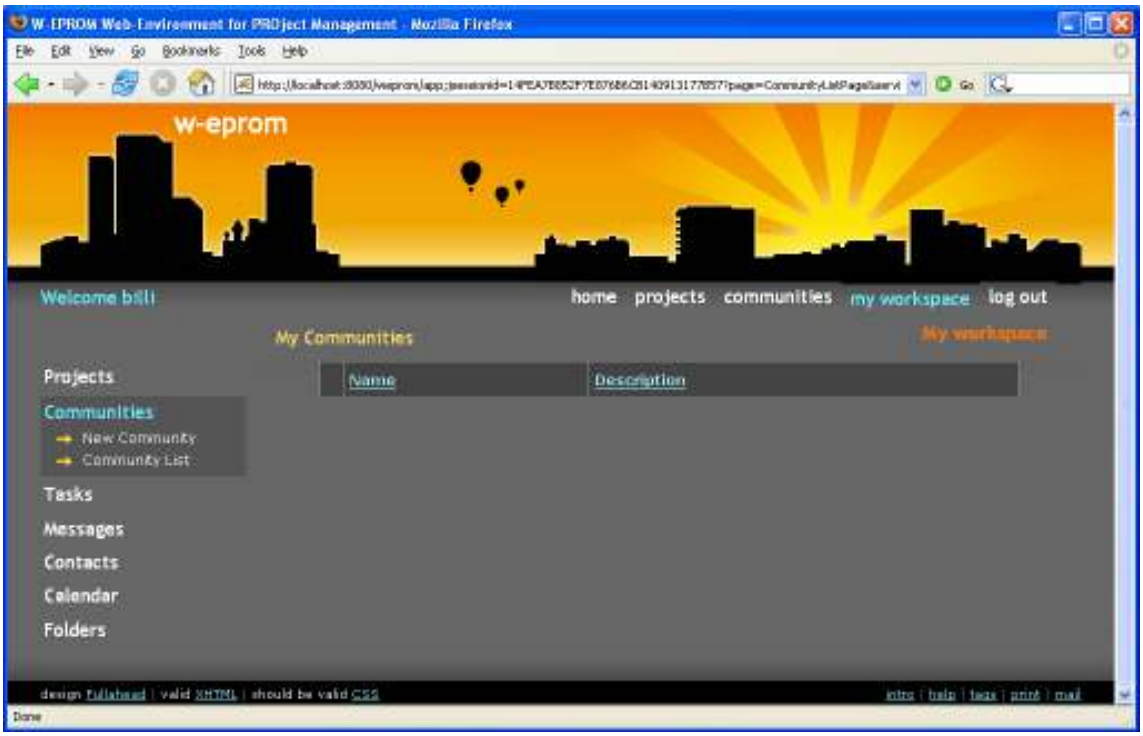
Фигура 9.13. Списък с проектите на потребителя

9.1.4.2. Групи по интереси

Аналогично на проектите, първоначално списъкът с групите по интереси е празен (Фиг. 9.14). Нова такава се добавя към него, ако:

а) Вие сте създали своя, в която автоматично сте получили роля „**координатор на група по интереси**”,

б) някой координатор на група по интереси Ви е поканил за членство и Вие сте приели.



Фигура 9.14. Празен списък с групите по интереси на потребителя

Роли в групата по интереси

Съществуват две потребителски роли във всеки проект- „**участник в група по интереси**”, обозначен с  и „**координатор на група по интереси**”, обозначен с  . Всяка от тях се характеризира с определени права. Важно е да споменем отново, че при създаване на група по интереси, Вие автоматично получавате и двете роли. Другата особеност е, че „**координатор на група по интереси**” включва в себе си „**участник в група по интереси**”, т.е. всички права, присъщи на *участника*, са преписани и на *координатора*.

Правата на координатора са:

- редактиране, изтриване на групата по интереси;
- редактиране на ролите на потребителите на групата по интереси;
- създаване, редактиране, изтриване на паметка от календара на групата по интереси;
- покана на нови членове;
- отстраняване на членове.

Права на участника:

- разглеждане на групата по интереси;
- разглеждане на паметка от календара на групата по интереси;
- създаване, разглеждане, изтриване на документ в проектна папка;
- създаване, разглеждане, изтриване на проектна папка.

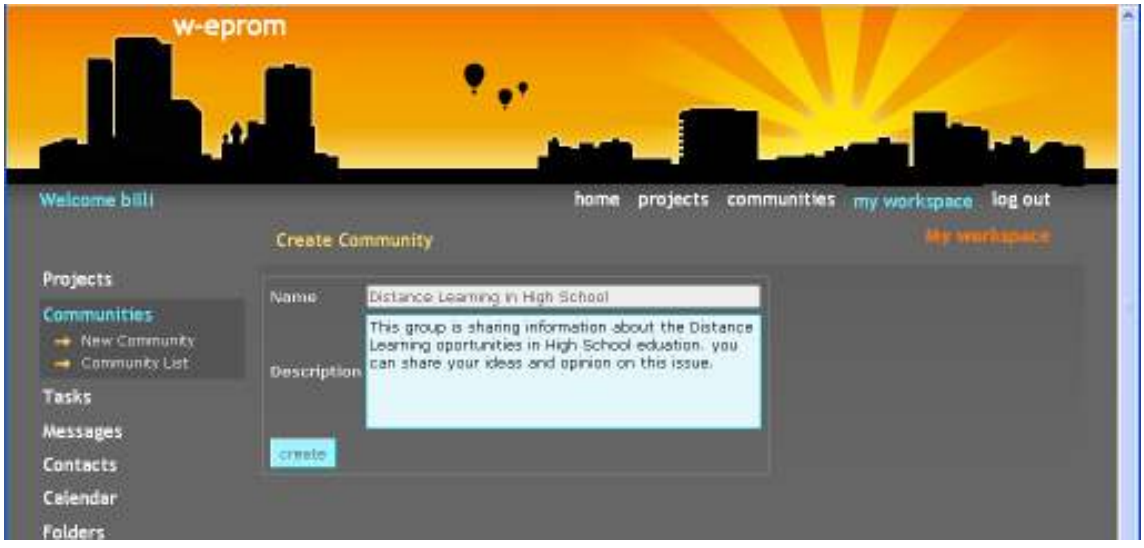
Създаване на група по интереси

Функцията се достига от лявото меню Communities/New Community, когато потребителят е „логнат”.

При създаването на нова група по интереси (Фиг. 9.15), две са само характеристиките, които могат да се зададат – име и описание.

Има възможност да сте дублирали името на групата с вече съществуващо. Тогава на екрана се появява съобщение за грешка.

След като въведете данните, натиснете бутона Create.

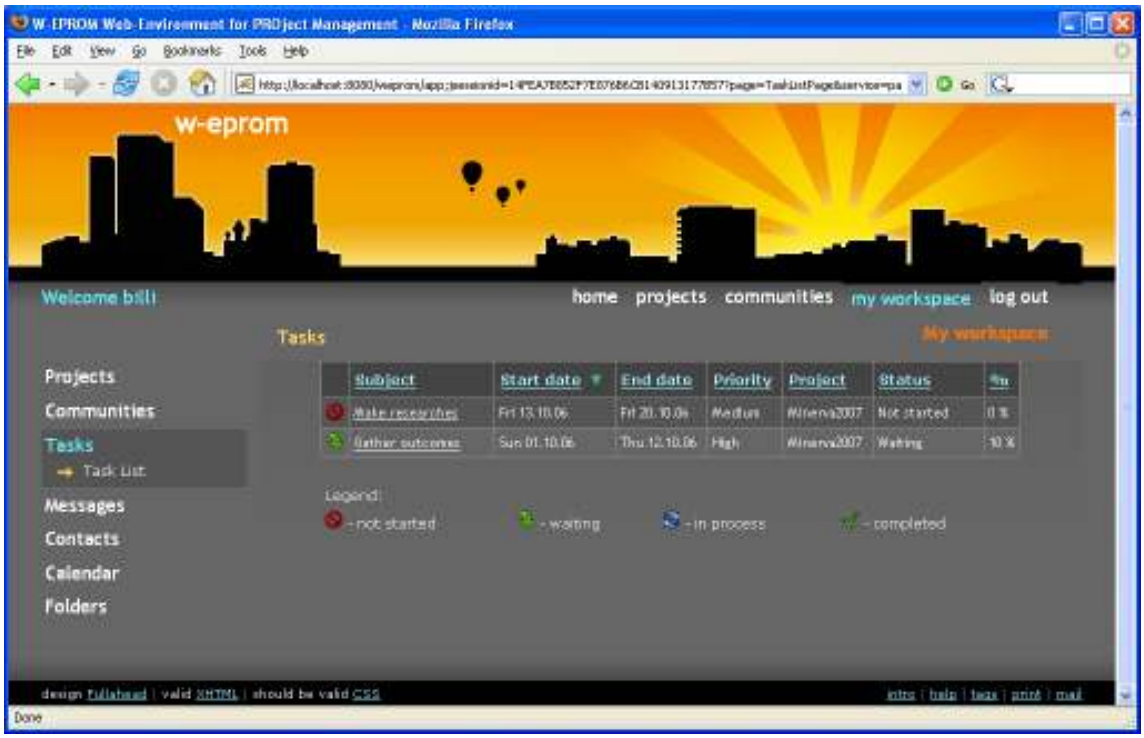


Фигура 9.15. Създаване на нова група по интереси

Ако транзакцията е успешна, на екрана се появява утвърдително съобщение, а ако се върнете на списъка с групи по интереси, новата е вече изброена, аналогично на тези случаи при проектите.

9.1.4.3. Задачи

Във Вашето лично пространство, вие имате достъп до всички задачи, на които сте собственик или изпълнител като изберете от лявото меню Tasks/Task List (Фиг. 9.16).



Фигура 9.16. Списък със задачите на потребителя

Статусът, в който се намират задачите е идентифициран със съответните икони:

- .нестартирана;
- в процес на изпълнение;
- процес на изчакване;
- завършена.

От екрана със списъка на задачите, можете да достъпите всяка една от тях, като я изберете по предмет (Subject) от таблицата. Така достигате част от WEPROM, която Ви позволява редактиране на свойствата на избраната задача или изтриването ѝ от проекта.

Вие може да редактирате задача, само ако сте неин собственик или изпълнител, а можете да изтриете задача, само ако сте неин собственик.

Задачите могат да се сортират по предмет, начална и крайна дата, приоритет, изпълнител, статус и степен на изпълнение за по-голямо удобство.

Вие не можете да създавате задача от Вашето лично пространство, защото всяка една такава е свързана с даден проект. Затова за повече информация относно операциите върху задачи, които можете да извършвате, вижте 9.5.3.

9.1.4.4. Съобщения

Вие имате възможност да ползвате услугите на WEPROM, касаещи съобщенията, единствено когато се намирате в своето работно пространство. Съобщенията се достигат от лявото меню Messages. Те биват няколко типа, идентифицирани със съответни иконки:

- ново съобщение ;
- прочетено съобщение ;
- такова, на което е отговорено ;
- такова, което е препратено ;
- такова, на което е отговорено и което е препратено .

Изпращане на съобщение

За да създадете и изпратите съобщение, изберете от лявото меню Messages/Compose Message (Фиг. 9.17).

Фигура 9.17. Създаване и изпращане на съобщение

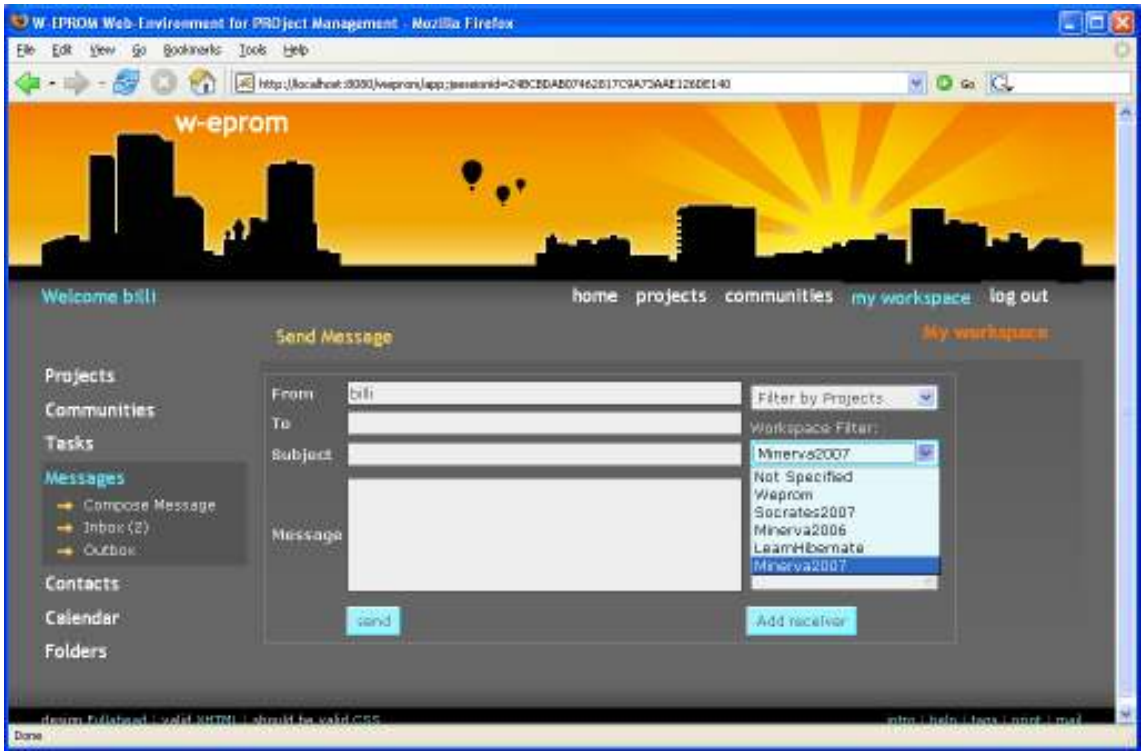
Изпращачът на съобщението е твърдо зададен – текущият потребител.

В дясната част на екрана има един под друг два филтъра и списък с потребители най-отдолу. Те са създадени, за да улеснят добавянето на потребител(и) към получателите на съобщението.

Първият филтър е по тип на работното пространство, съответно:

- филтър по *проекти* (Filter by Projects),
- филтър по *групи по интереси* (Filter by Communities).

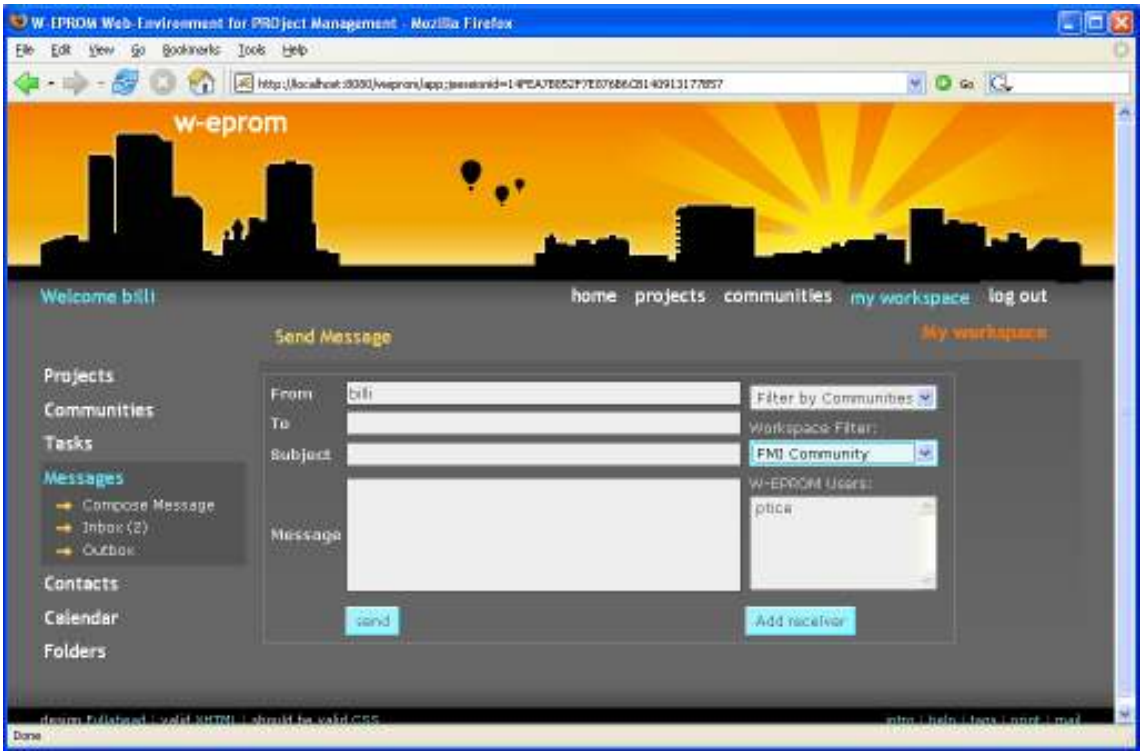
При избор на *филтър по проекти* (Фиг. 9.18), вторият филтър се пълни с имената на всички проекти, създадени в системата. Съответно при избор на *филтър по групи по интереси* (Фиг. 9.19), вторият филтър се пълни с имената на всички групи по интереси.



Фигура 9.18. Филтър по проекти

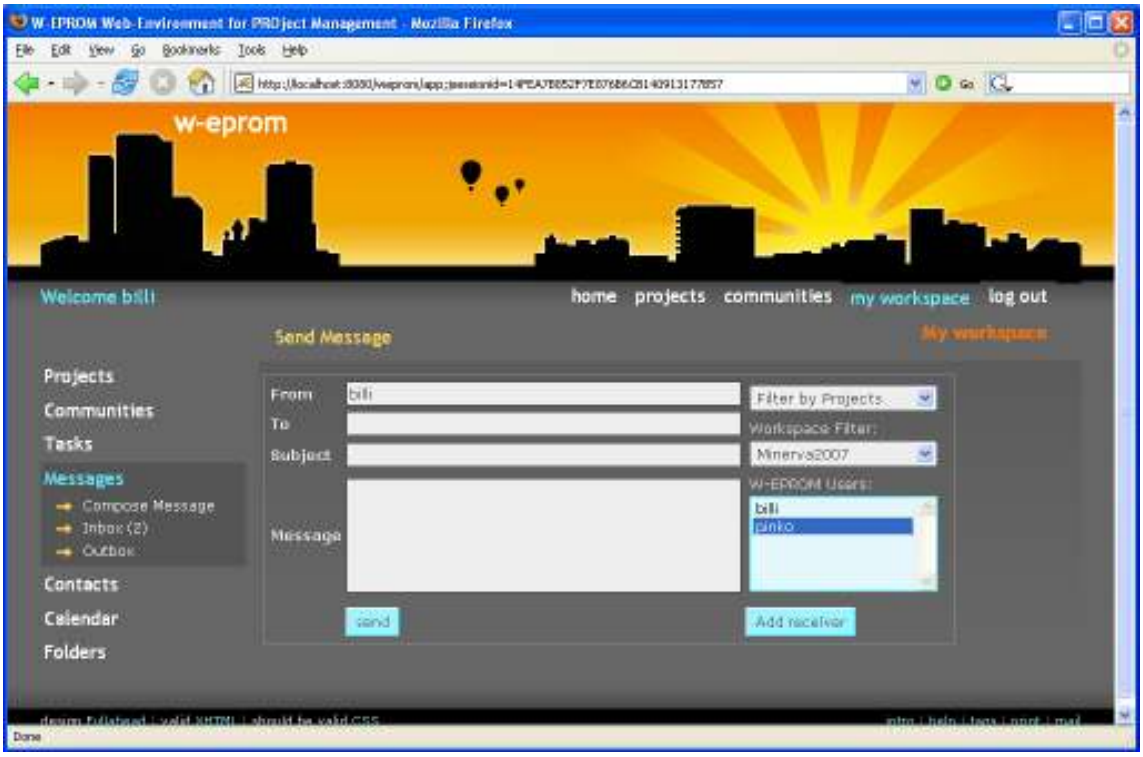
При избор на даден проект или група, в списъка с потребители се изброяват всички членове на съответните проект/ група.

Ако не са специфицирани стойности във филтрите, тогава списъкът съдържа всички потребители на WEPROM.



Фигура 9.19. Филтър по групи по интереси

След като изберете получател, натиснете бутона Add receiver (Фиг. 9.20).



Фигура 9.20. Избор на получател на съобщение

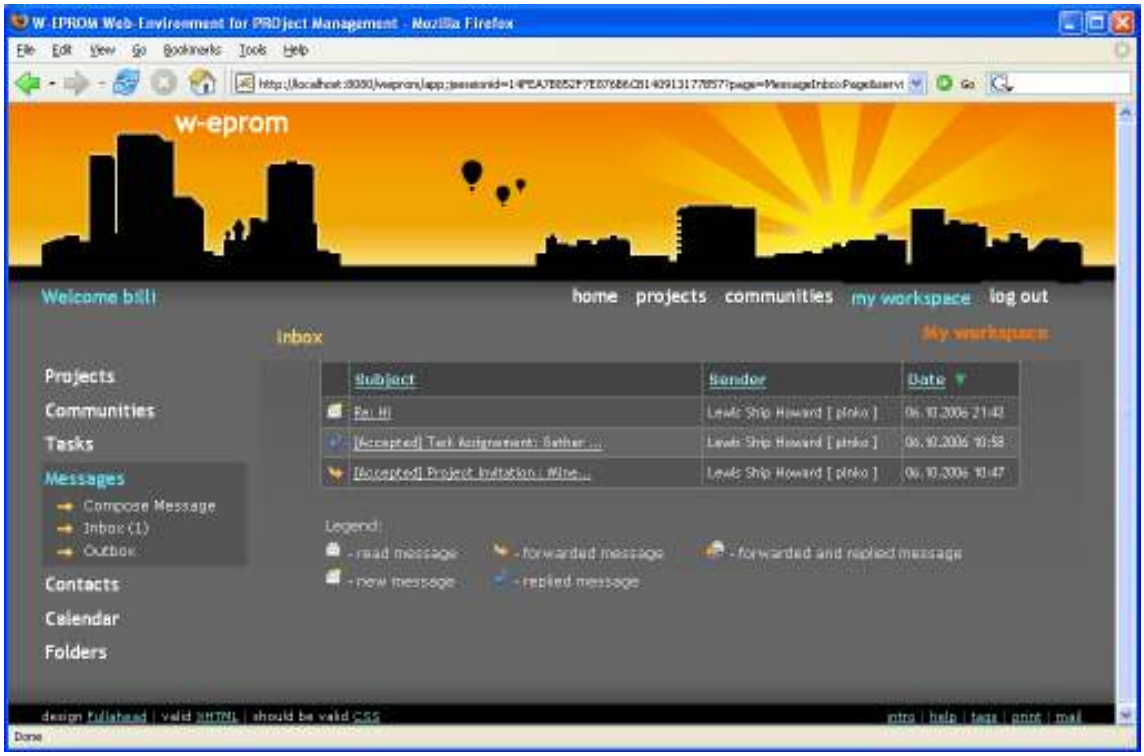
Разбира се, едно съобщение може да има повече от един получатели. Може да го изпратите до всички членове на даден проект или до избрани потребители. Всеки от тях ще получи по едно копие от Вашето съобщение във входящата си кутия.

Остава само да напишете самото съобщение и да натиснете бутона send.

Ако съобщението е изпратено успешно, на екрана се извежда „Your message has been sent!"

Входяща кутия

Като регистриран потребител на WEPROM със собствено работно пространсво, Вие разполагате с входяща кутия (Inbox), в която получавате всички нови текстови или системни съобщения (Фиг. 9.21). Броят на новите съобщения е изписан в лявото меню в скобки.



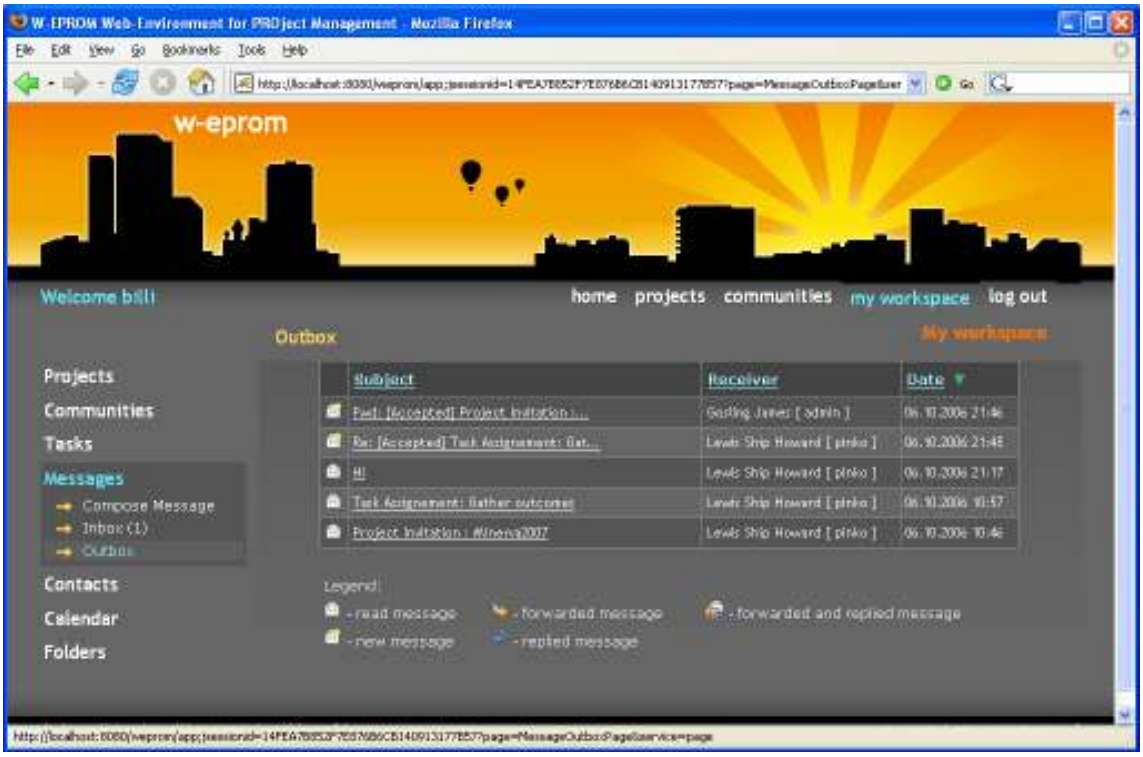
Фигура 9.21. Входяща кутия (Inbox) за съобщения на потребителя

Съобщенията тук могат да се сортират по предмет, изпращач и дата.

Изходяща кутия

Разбира се, всяко изпратено от вас съобщение автоматично се запазва в изходящата кутия (Outbox), с която разполагате (Фиг. 9.22).

Съобщенията в нея могат да се сортират по предмет, получател и дата за по голямо удобство.



Фигура 9.22. Изходяща кутия (Outbox) за съобщения на потребителя

9.1.4.5. Контакти

Вие можете да менажирате своите контакти, само от личното си работно пространство. Контактите се достигат от лявото меню Contacts.

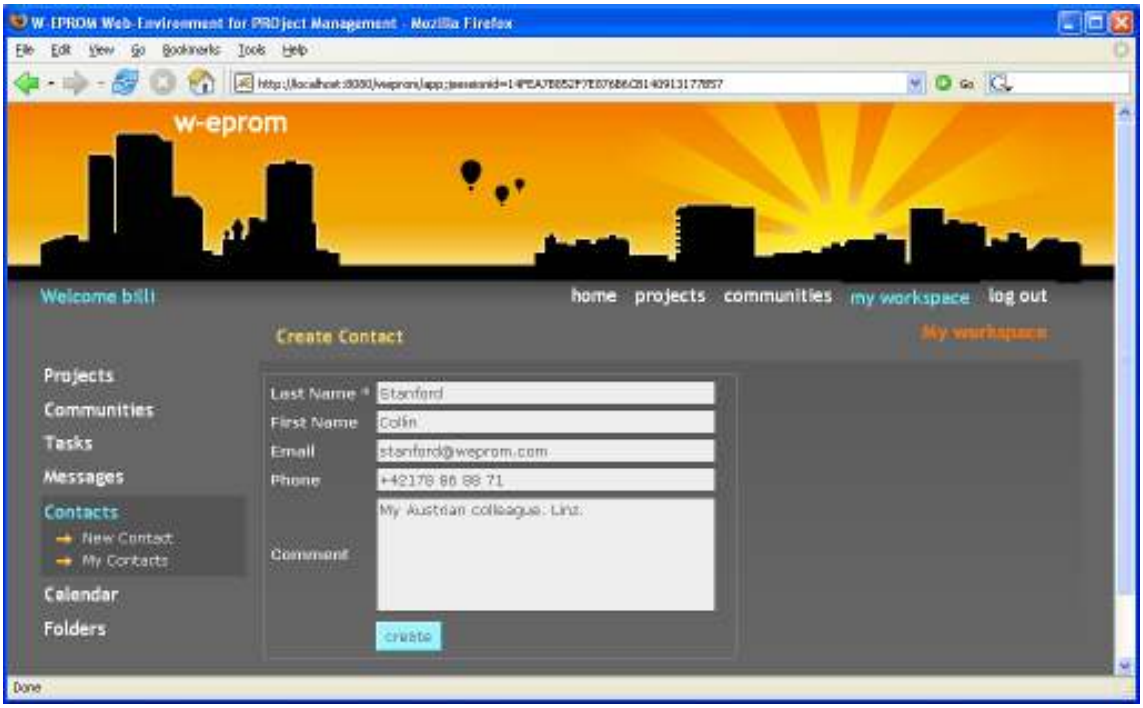
Създаване на нов контакт

Нов контакт може да бъде създаден от Contacts/New Contact.

Когато създавате контакт (Фиг. 9.23), Вие имате възможност да зададете следните характеристики:

- фамилия;
- име;
- електронен адрес;
- телефон;
- коментар.

Не забравяйте да зададете фамилия на контакта.



Фигура 9.23. Създаване на нов контакт на потребителя

Попълните желаната информация, натиснете бутона create. При успешна транзакция на екрана се извежда утвърдително съобщение “Your contact has been created!”.

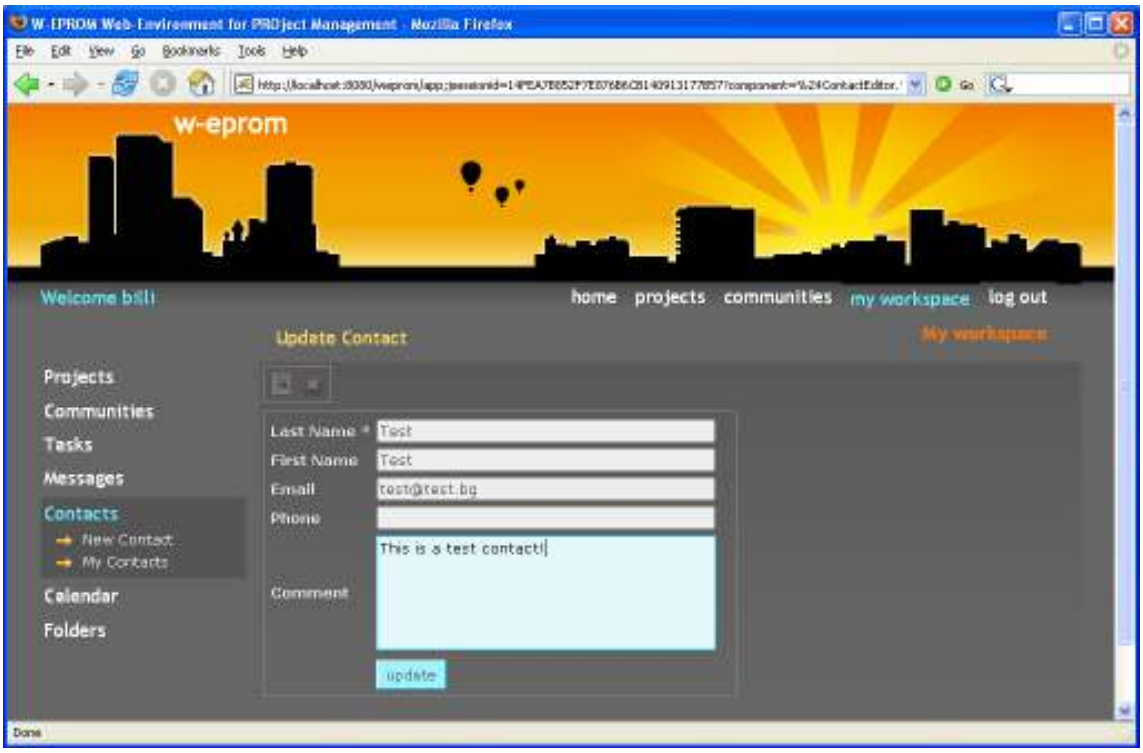
Редакция и изтриване на контакти

Ако изберете от лявото меню Contacts/My Contacts, Вие ще достъпите своя бележник с контакти (Фиг. 9.24). В него Вие можете да сортирате контактите по име, фамилия и електронен адрес.

Фигура 9.24. Бележник с контакти на потребителя

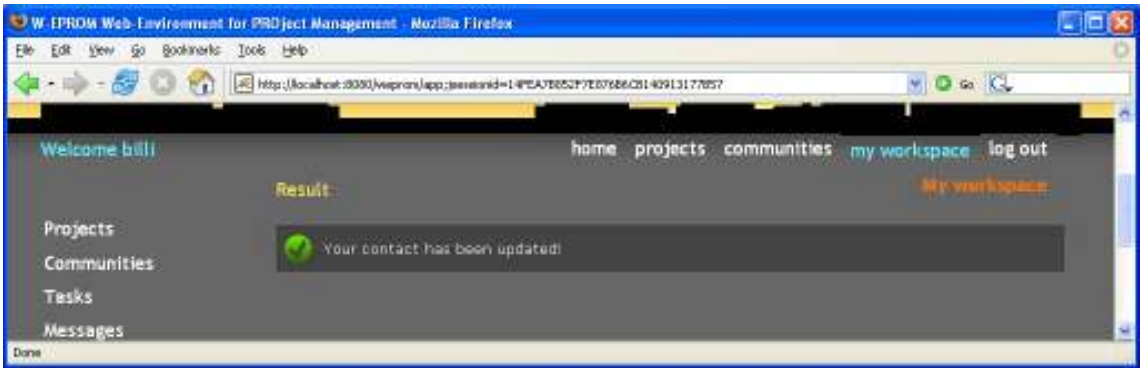
Контакт е достъпен за редактиране, ако го изберете по фамилия от таблицата. Така достигате част от WEPROM, която Ви позволява редактиране на свойствата на избрания контакт или изтриването му.

За режим на редактиране (Фиг. 9.25), натиснете бутона  .



Фигура 9.25. Редактиране на контакт

След като направите желаните промени, натиснете бутона Update. При успешна транзакция на екрана се извежда утвърдително съобщение (Фиг. 9.26).



Фигура 9.26. Резултат от редактиране на контакт

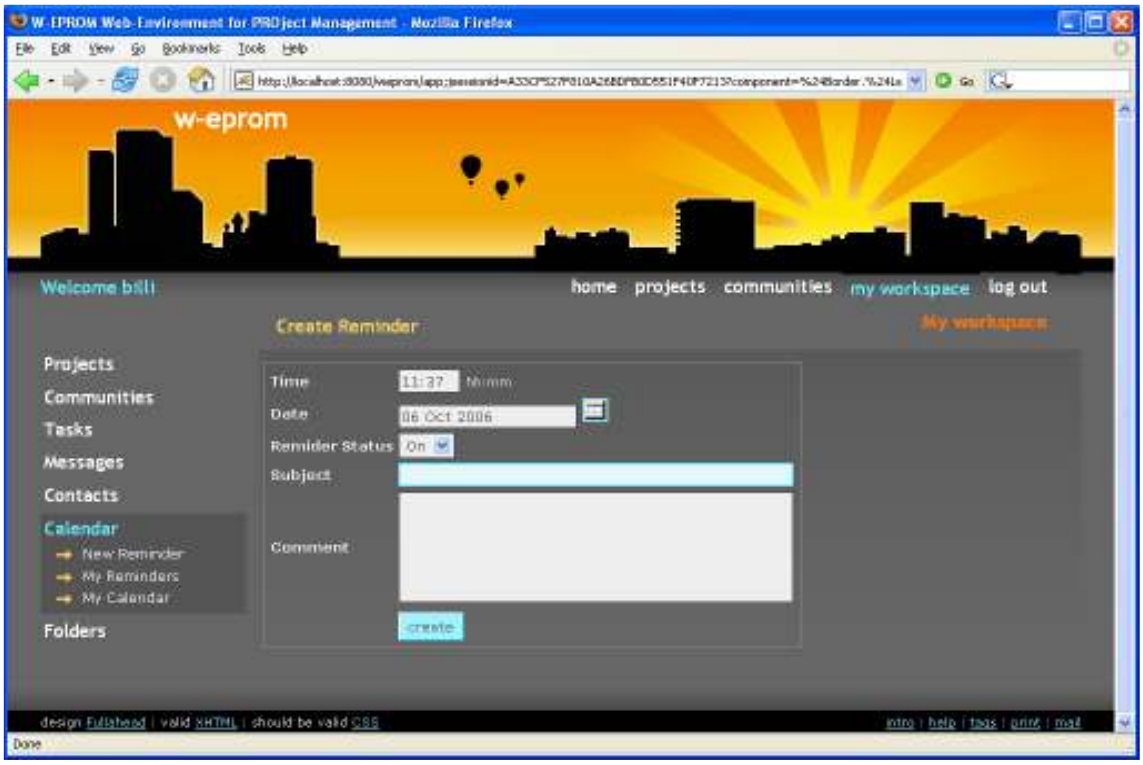
За режим на изтриване, натиснете бутона  . А след това и „delete”. При успешна транзакция на екрана се извежда подобно утвърдително съобщение.

9.1.4.6. Календар

Създаване на нова паметка

Нова паметка в личното работното пространство на потребител може да бъде създадена от:

- Calendar New Reminder – в този случай датата по подразбиране на паметката е текущата дата (Фиг. 9.27).



Фигура 9.27. Създаване на нова паметка от лявото меню

- от My Calendar, ако изберете произволен ден, за който няма все още създадени паметки, автоматично ще се отвори за създаване нова паметка за конкретния ден (Фиг. 9.28). В този случай датата по подразбиране на паметката е датата на избрания ден (Фиг. 9.29).

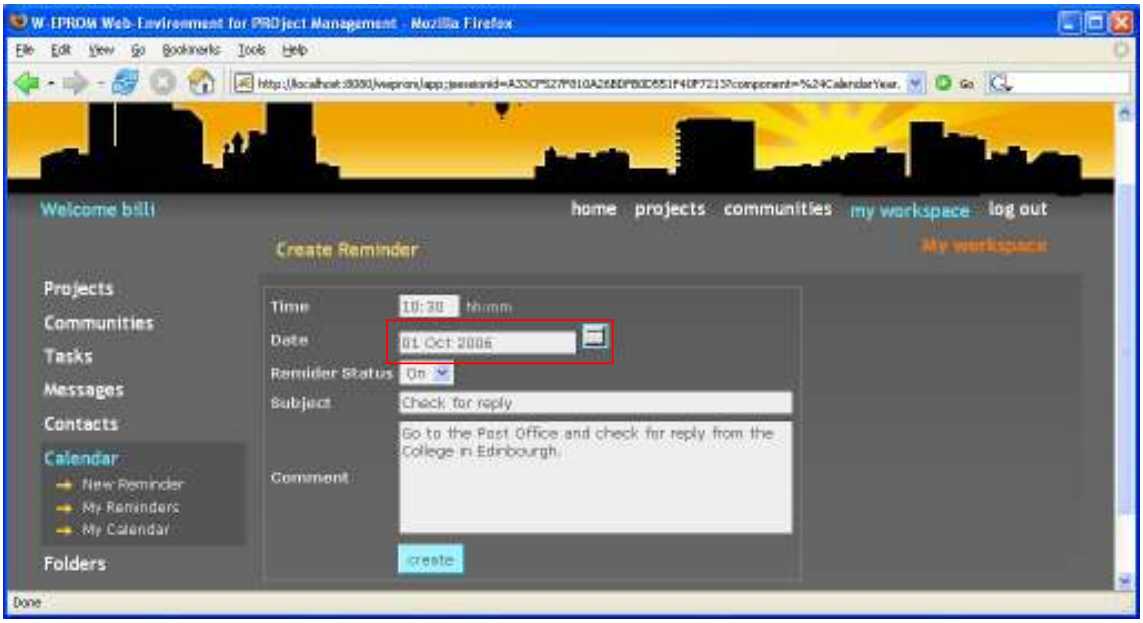
Когато създавате паметка, Вие имате възможност да дадете следните й характеристики:

- време (час:минути задължително във формат чч:мм),
- дата,
- статус (активна/ неактивна),
- преедмет,

- описание.

Не забравяйте да зададете предмет на паметката. След като въведете данните, натиснете бутона Create (Фиг. 9.29).

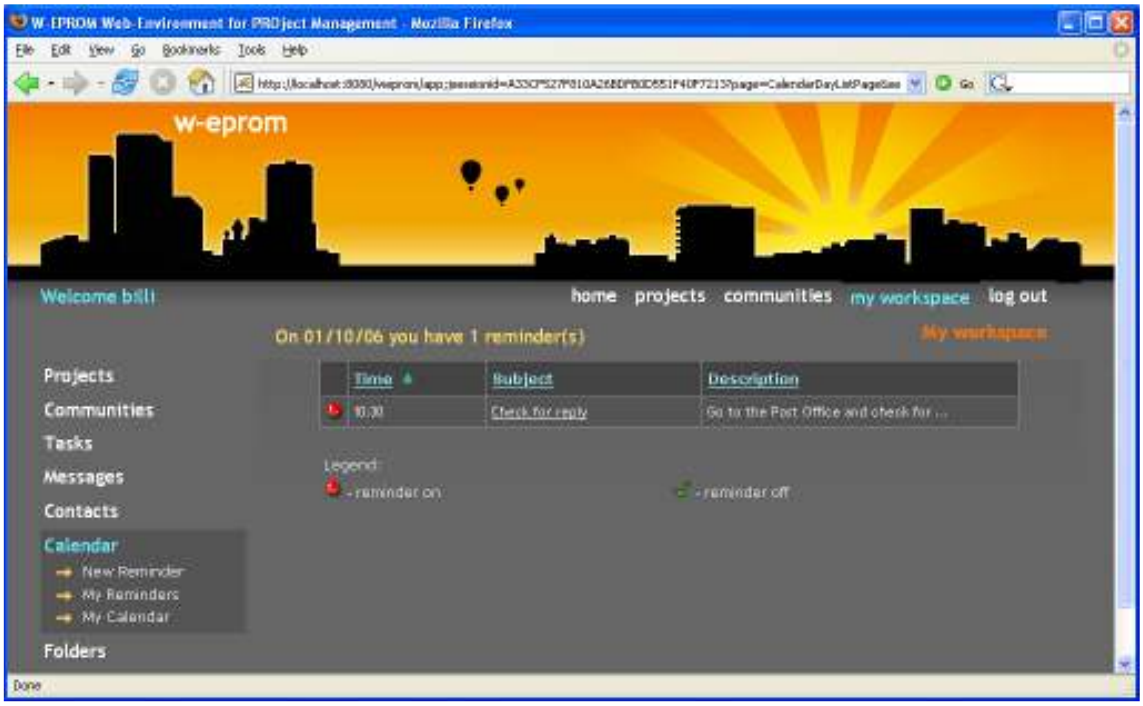
Фигура 9.28. Създаване на нова паметка от календара



Фигура 9.29. Създаване на нова паметка

Фигура 9.30. Резултат от създаване на нова паметка

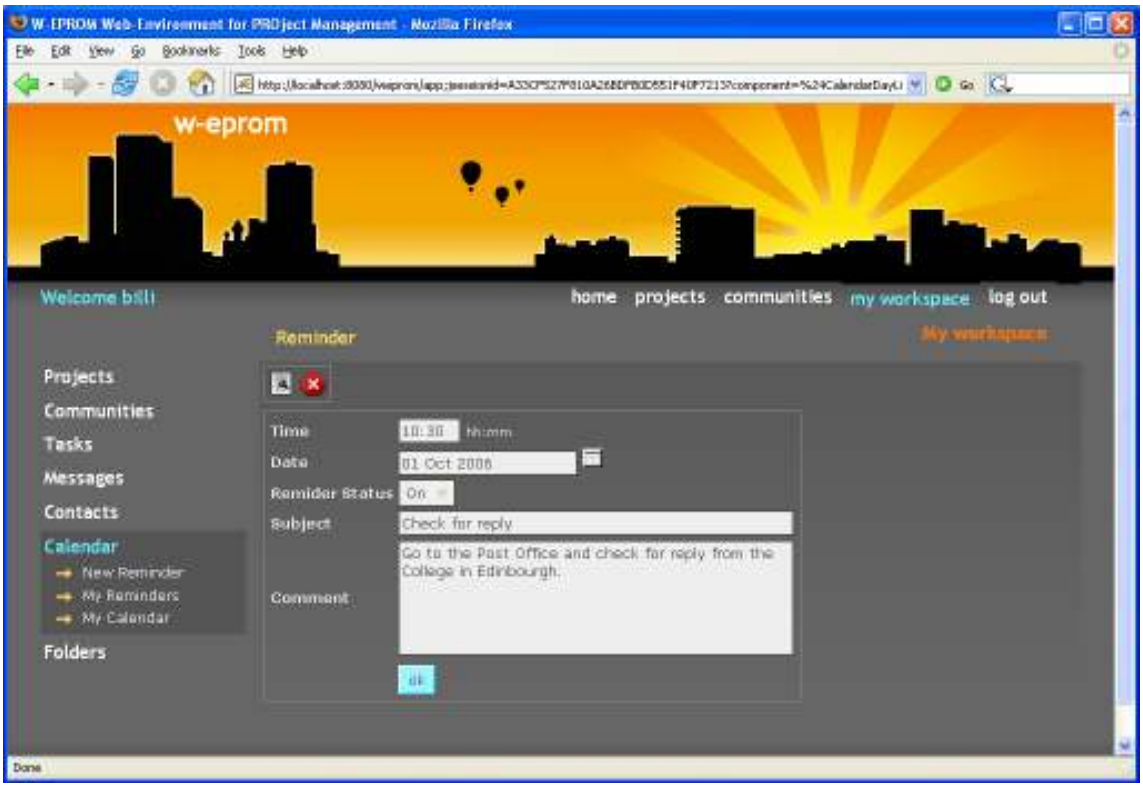
Връзката “Back to reminders” (Фиг. 9.30) Ви препраща към списъка с памет-ки за току що редактирания ден. Легендата под таблицата ще Ви помогне да се ориентирате по-бързо за статуса на паметката- дали тя е активна или не (Фиг. 9.31).



Фигура 9.31. Списък с паметки за даден ден

Редакция и изтриване на паметка.

От екрана със списъка на паметките за дадения ден, можете да достъпите всяка една от тях, като я изберете по предмет от таблицата. Така достигате част от WEPROM, която Ви позволява редактиране на свойствата на избраната паметка или изтриването (Фиг. 9.32).



Фигура 9.32. Преглеждане на паметка

За режим на редактиране, натиснете бутона  . След като направите желаните промени, натиснете бутона Update. При успешна транзакция на екрана се извежда утвърдително съобщение (Фиг. 9.33).

Фигура 9.33. Резултат от редакция на паметка

За режим на изтриване, натиснете бутона  . А след това и „delete”. При успешна транзакция на екрана се извежда утвърдително съобщение (Фиг. 9.34).

Фигура 9.34. Резултат от изтриване на паметка

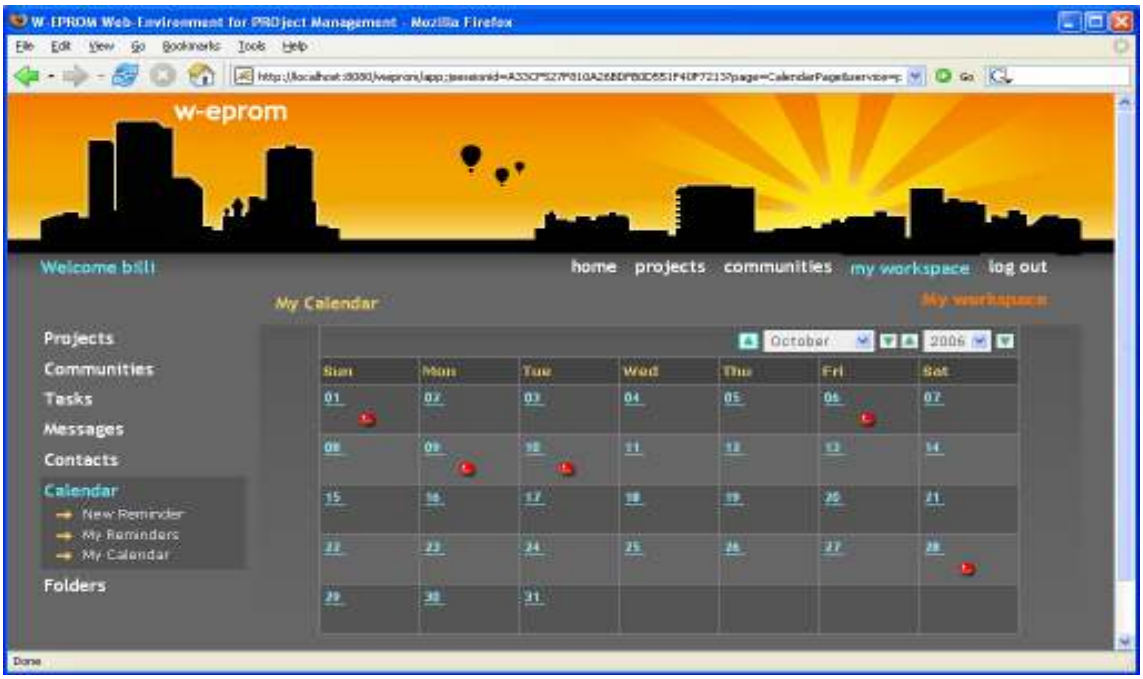
Разглеждане на всички паметки.

Ако желаете да разгледате всички свои паметки наведнъж, можете да го направите през Calendar My Reminders. Паметките могат да се сортират по време, предмет и описание за по голямо удобство (Фиг. 9.35).

Фигура 9.35. Списък от всички паметки

Разглеждане на календара.

Вашият календар е достъпен през Calendar My Calendar.



Фигура 9.36. Календар

В него са отбелязани нагледно дните, в които има записани паметки. Освен това, чрез падащите менюта за месец и година и бутоните до тях, можете да се „движите" по календара (Фиг. 9.36).

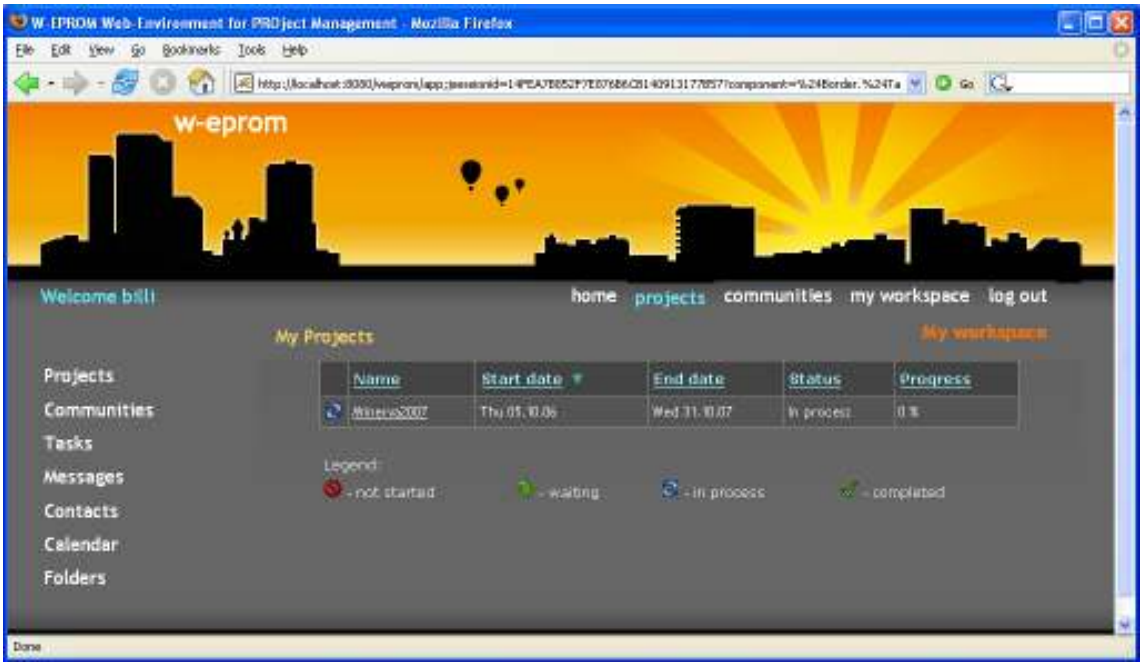
9.4.7 Папки и документи

Папките и документите са достъпни през Folders (Фиг. 9.37).

Фигура 9.37. Документи

9.1.5. Управление на проект.

От екрана със списъка с вашите проекти, можете да достъпите всеки един от тях, като го изберете по име от таблицата (Фиг. 9.38).



Фигура 9.38. Списък с проекти

Така автоматично влизате в работното пространство на този проект (Фиг. 9.39).

Даден проект може да бъде достъпен само от членовете му. Той съдържа информация за тях, както и за всички задачи и паметки. Проектът разполага също така с календар и главна папка.

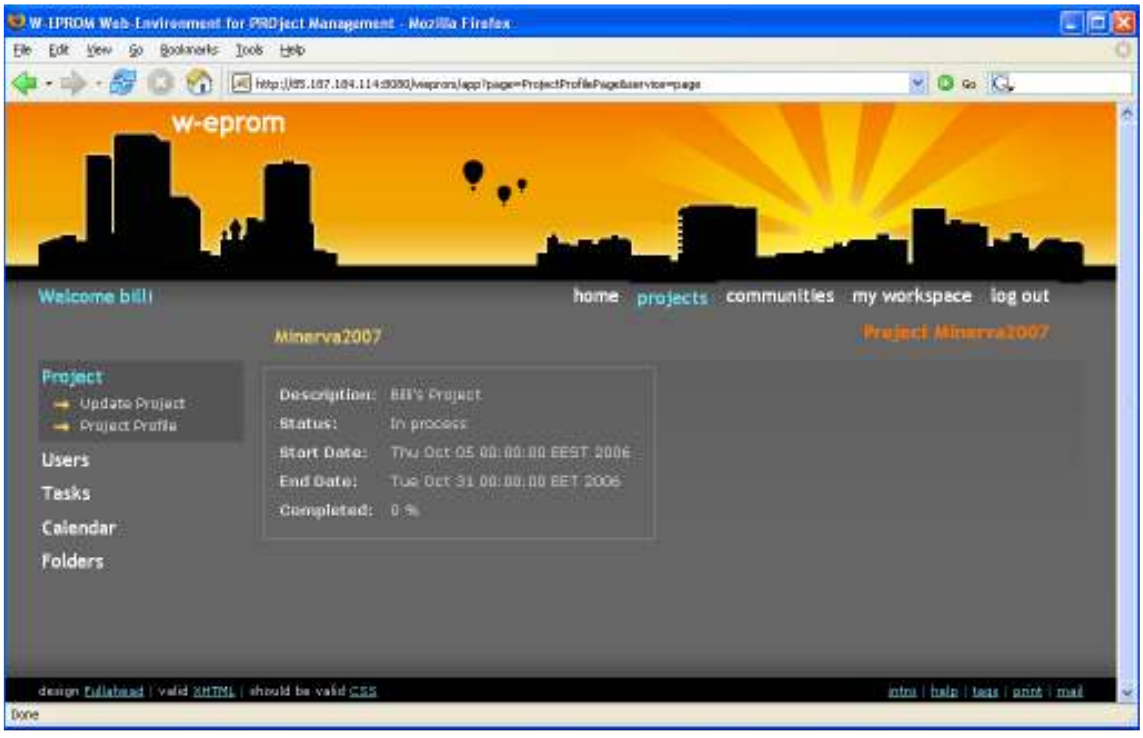
9.1.5.1. Проектен профил.

Проектният профил е достъпен за разглеждане от Project Project Profile (Фиг. 9.40), а за редакция от Project Update Project бутона  (Фиг. 9.41).

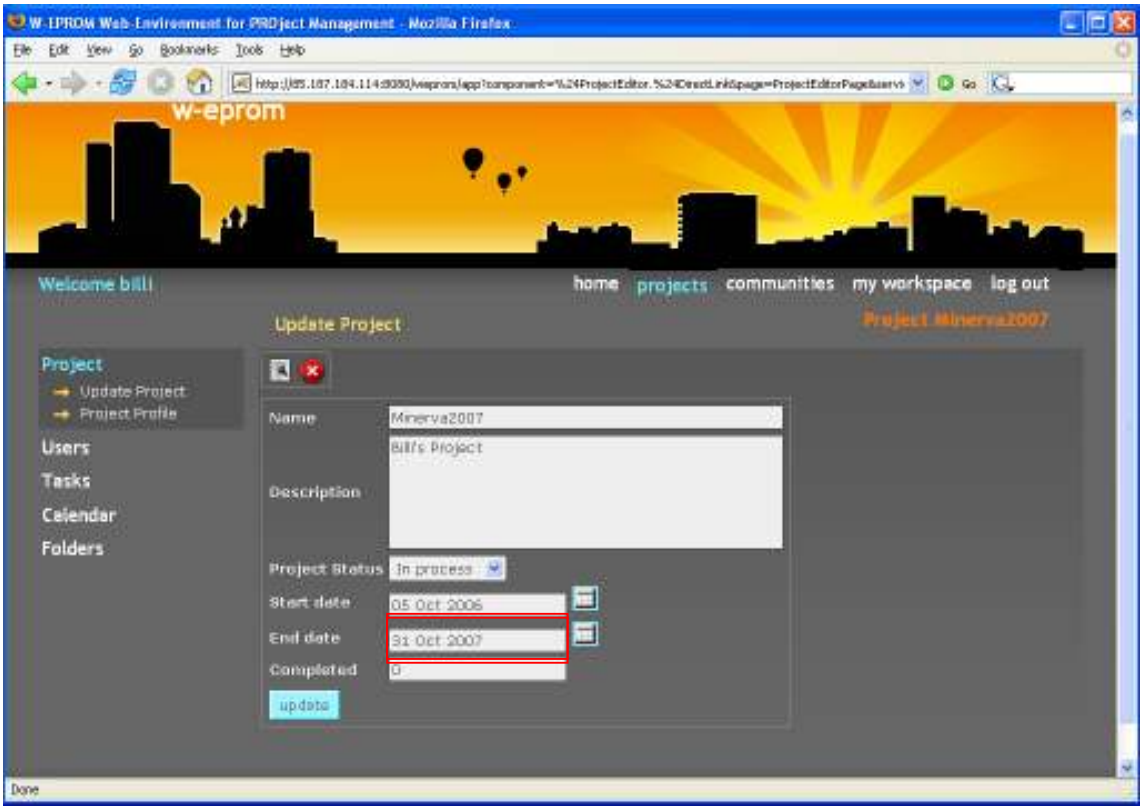
След редактиране на данните, натиснете бутона Update (Фиг. 9.41). Ако транзакцията е успешна, на екрана се изписва утвърдително съобщение (Фиг. 9.42).

Ако разгледате отново проектния профил, ще видите промените (Фиг. 9.43).

Фигура 9.39. Работно пространство на проект



Фигура 9.40. Профил на проект



Фигура 9.41. Профил на проект, отворен за редакция

Фигура 9.42. Резултат от редакция на проектен профил

Фигура 9.43. Променен проектен профил

9.1.5.2. Членове на проект.

Изпращане на покана до потребител за членство в проект.

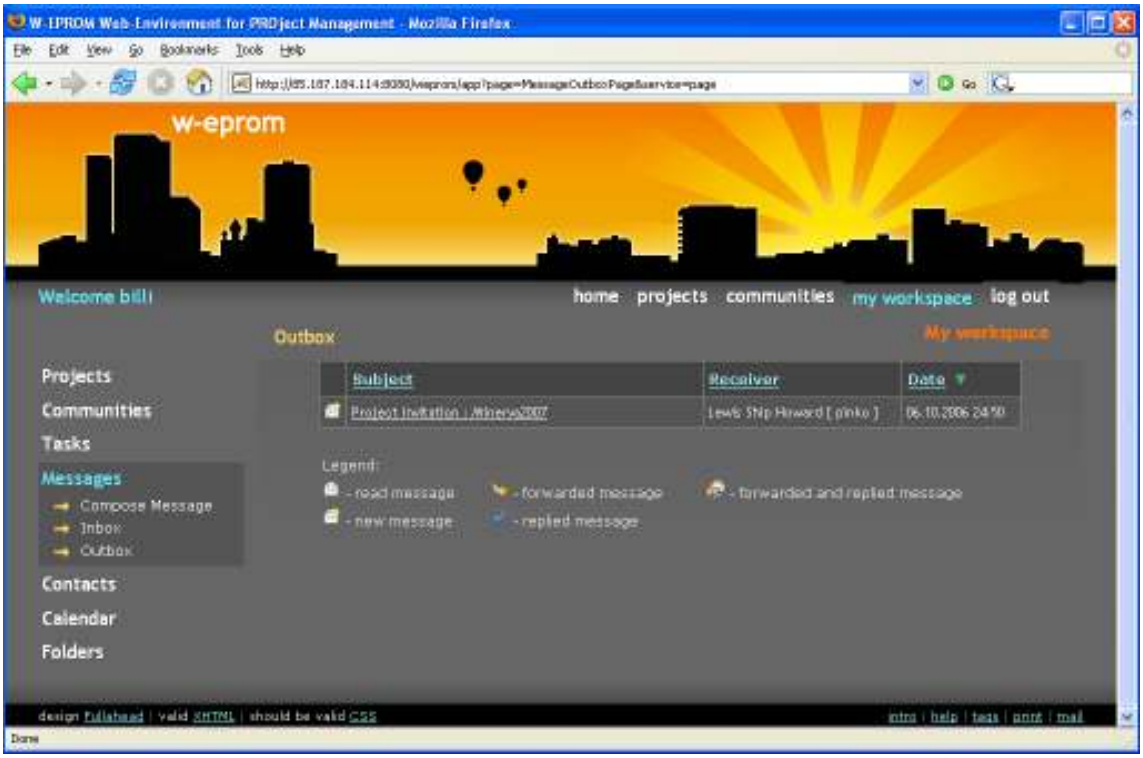
Ако сте координатор, за да изпратите покана до потребител за членство в проект, изберете Users Invite User. Извежда се списък с всички регистрирани потребители на системата, които не са членове на текущия проект (Фиг. 9.44).

Фигура 9.44. Изпращане на покана до потребител за членство в проект

При „кликване” на нечие потребителско име от таблицата (в случая на pinko), автоматично се изпраща системно съо**бщение- покана за присъединяване към проекта** (Фиг. 9.45).

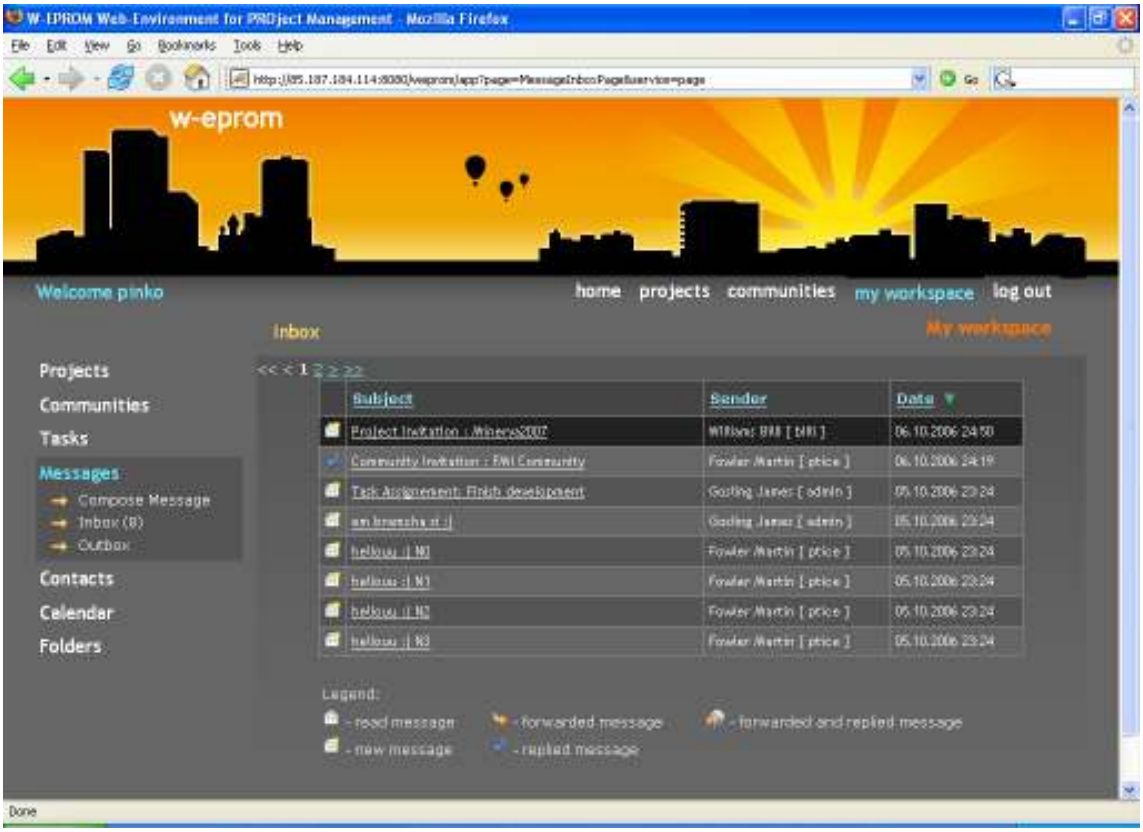
Фигура 9.45. Изпращане на покана до потребител за членство в проект

Това съобщение е изходящо за координатора- той е изпращач (Фиг. 9.46),



Фигура 9.46. Изпращане на изходящо съобщение покана

входящо за поканения потребител- той е получател (Фиг. 9.47) –

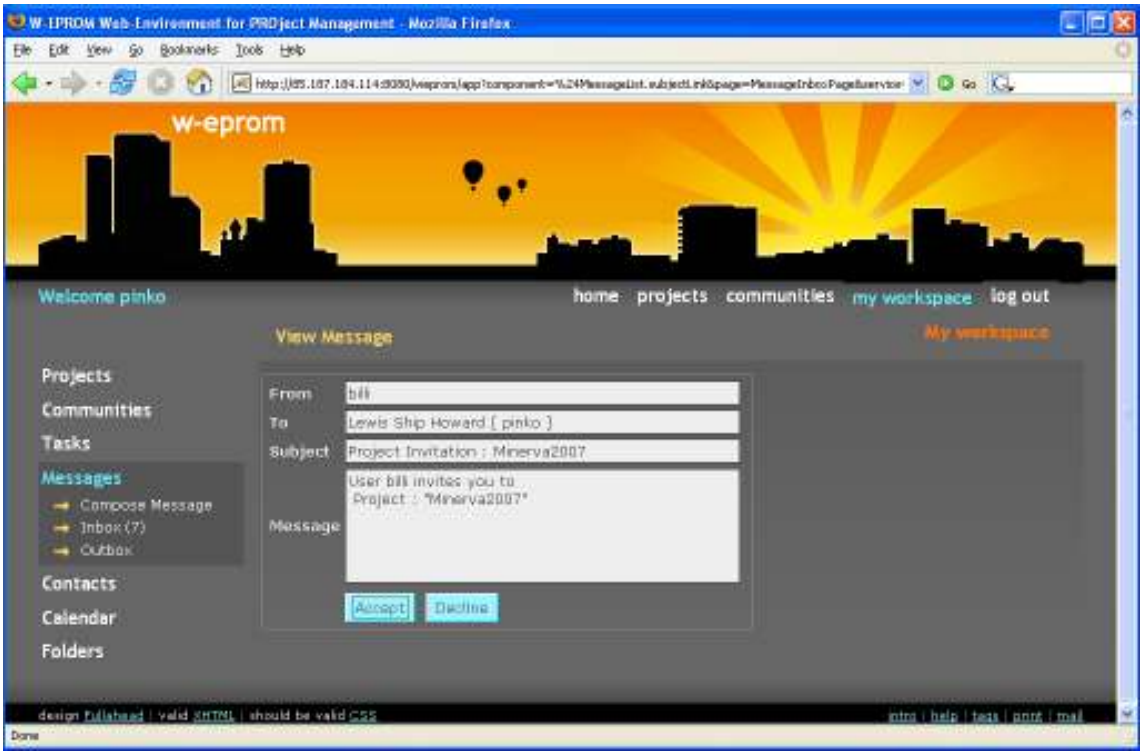


Фигура 9.47. Получаване на входящо съобщение покана

и има твърдо зададени предмет и текст (Фиг. 9.48).

Фигура 9.48. Съобщение покана

Когато поканеният потребител прочете **съобщението- покана**, той има две възможности- да приеме или да откаже поканата (Фиг. 9.49). Ако приеме с натискане на бутона Accept, той се добавя към членовете на проекта(Фиг. 9.50).



Фигура 9.49. Приемане на покана

Фигура 9.50. Резултат от приемане на покана

Ако откаже съответно чрез Decline, не се добавя към тях.

И в двата случая обаче, се изпраща ново системно **съобщение с дадения отговор** до координатора на проекта, който първоначално е изпратил поканата. Това съобщение е изходящо за поканения потребител- той е изпращач (Фиг. 9.51),

Фигура 9.51. Изпращане на системно съобщение за приемане поканата

входящо за координатора - той е получател (Фиг. 9.52)-

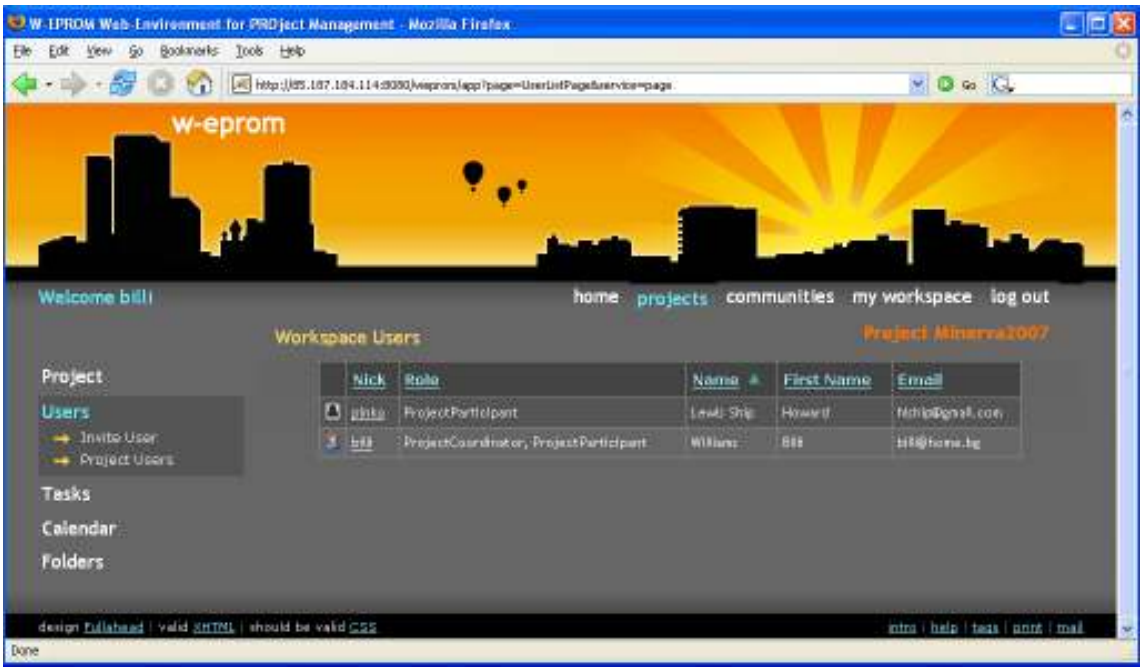
Фигура 9.52. Получаване на системно съобщение за приемане поканата

и има твърдо зададени предмет и текст (Фиг. 9.53).

Фигура 9.53. Системно съобщение за приемане на поканата

Редакция и изтриване на член на проект.

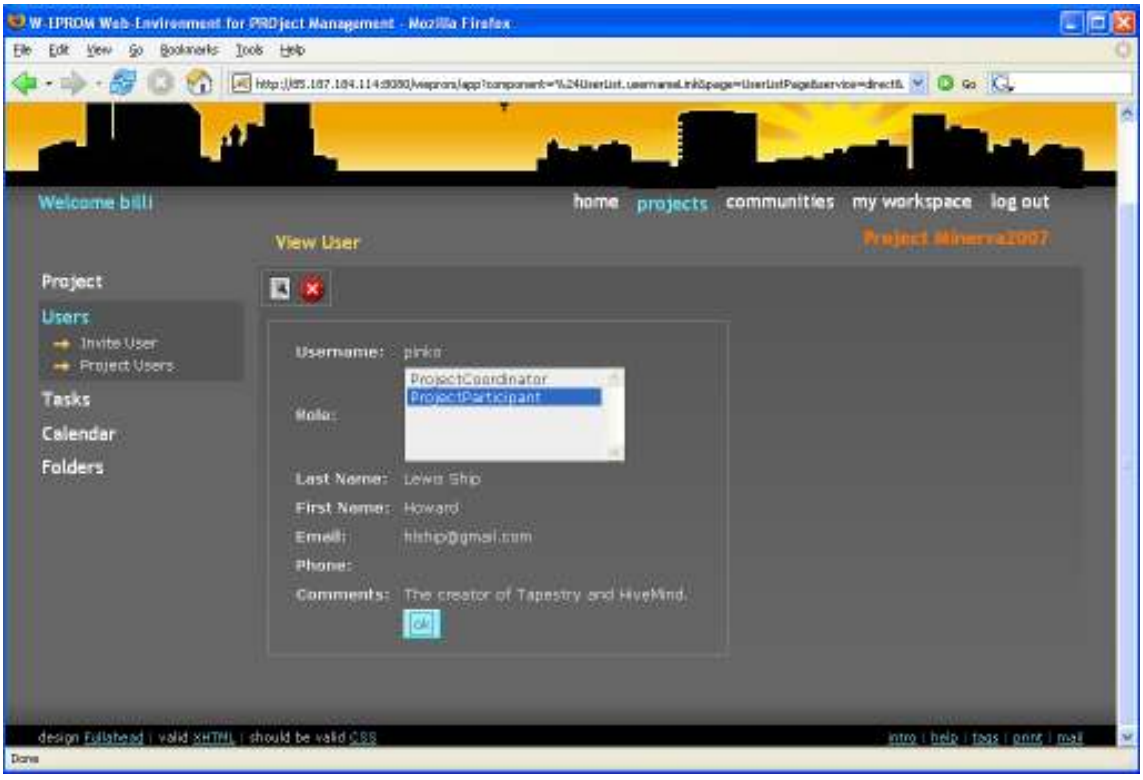
Членовете на проект са достъпни от Users Project Users(Фиг. 9.54).



Фигура 9.54. Членове на проект

Вие може да сортирате членовете на проекта по потребителско име, роля, фамилия, име и електронен адрес.

От екрана със списъка на членовете, можете да достъпите всеки един от тях, като го изберете по потребителско име от таблицата. Така достигате потребителския редактор (Фиг. 9.55), който Ви позволява редактиране на ролите на съответния член или премахването му от проекта. Двете операции са позволени само за координаторите на проекта.

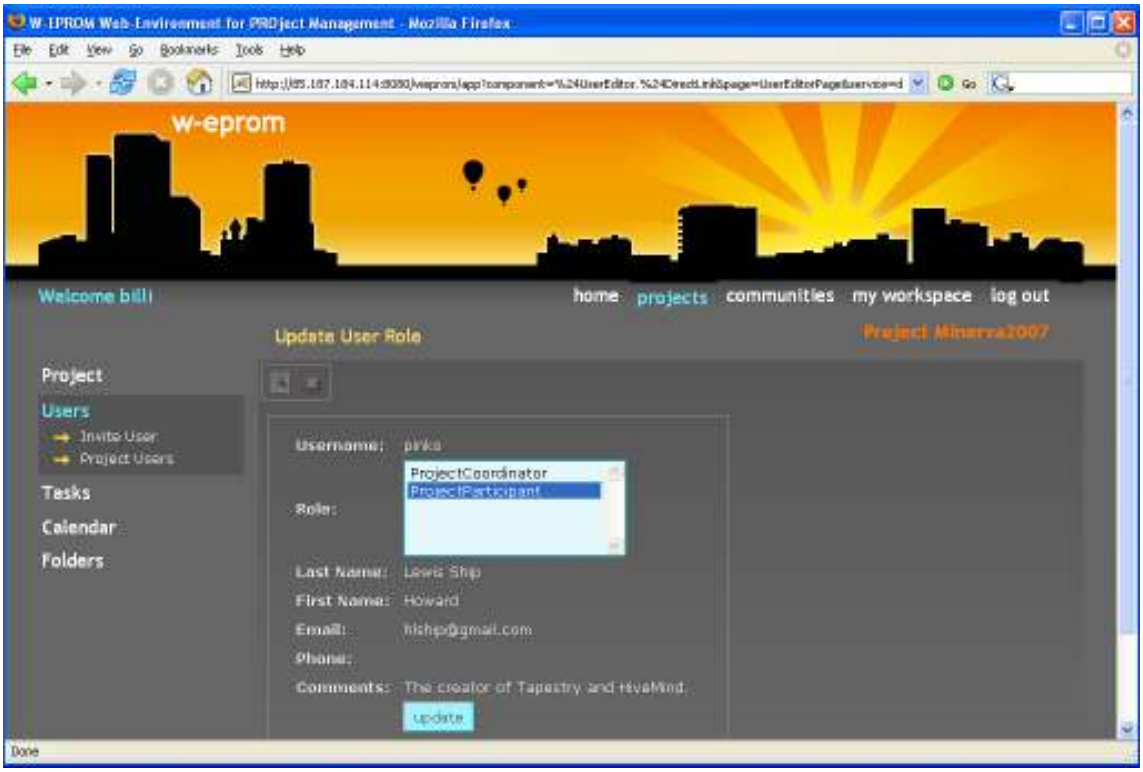


Фигура 9.55. Редактиране на член на проект

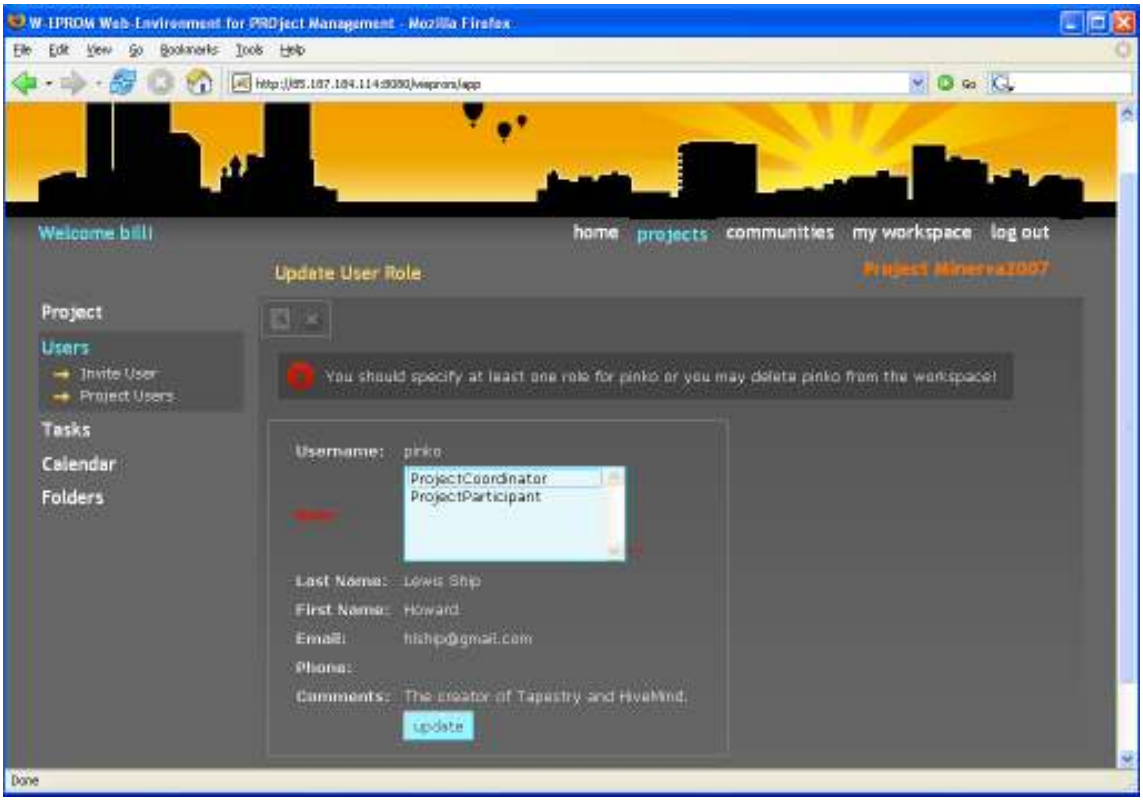
За режим на редактиране, натиснете бутона  (Фиг. 9.56).

Редактирайте ролите и натиснете бутона Update. Ако нито една роля не е избрана, на екрана се появява съобщение за грешка (Фиг. 9.57).

*Тъй като ролята „**координатор на проект**” включва в себе си „**участник в проект**”, не е нормално да е избрана само „**координатор на проект**”. Затова „**участник в проект**” не се изтрива, дори и да не е селектирана.*

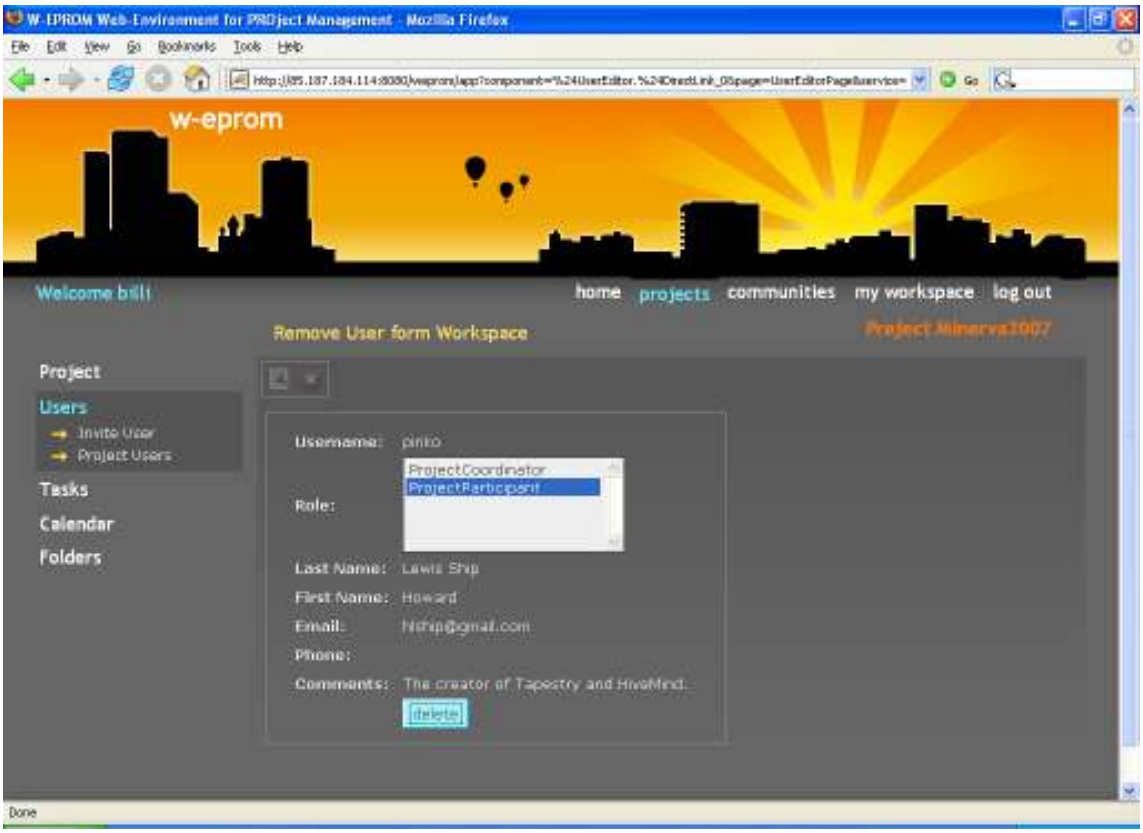


Фигура 9.56. Редактиране на член на проект



Фигура 9.57. Грешка при редактиране на член на проект

За режим на изтриване, натиснете  (Фиг. 9.58).



Фигура 9.58. Изтриване на член на проект

А след това и бутона delete(Фиг. 9.59).

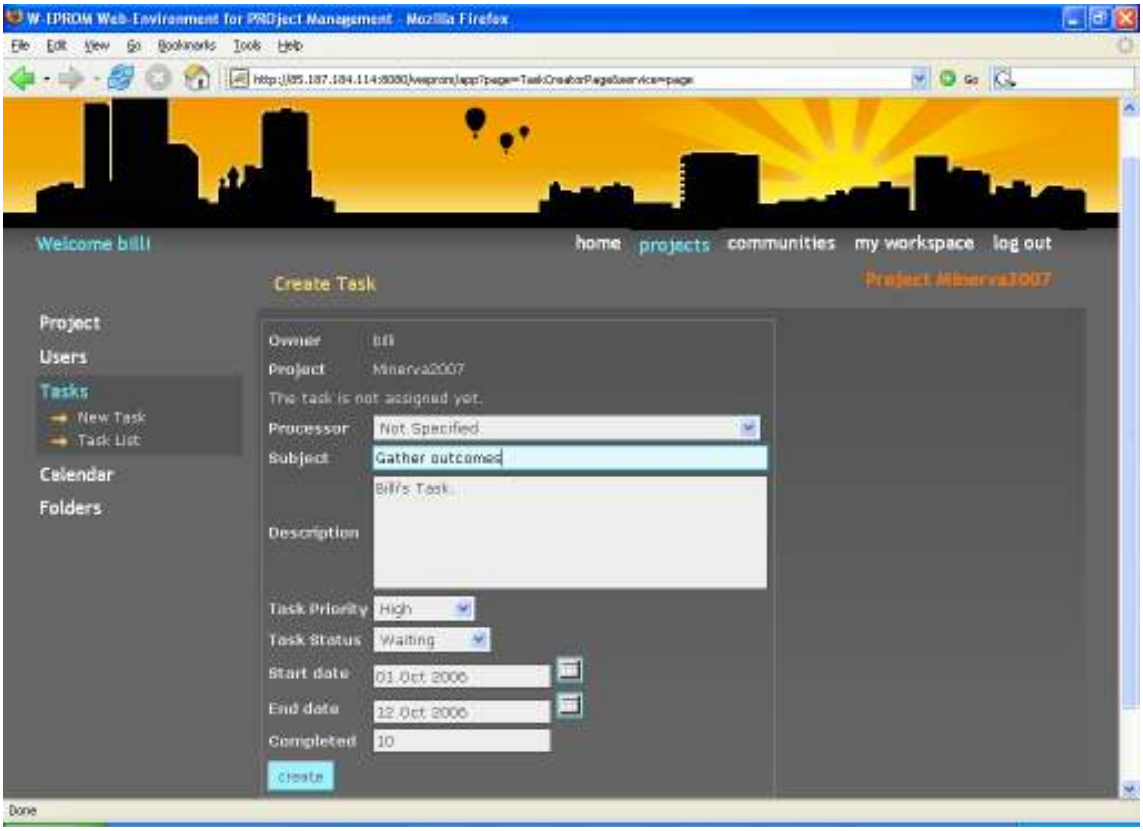
Фигура 9.59. Резултат от изтриване на член на проект

9.1.5.3. Задачи на проекта.

Създаване на нова задача в проект.

Нова задача може да бъде създадена само в работното пространство на проект от негов член-координатор и функцията е достъпна от Tasks New Task. В зависимост от това кой потребител и в кой проект я създава, веднага се определят две от основните константни свойства на задачата- притежател/собственик и проект, към който тя принадлежи. Няколко други характеристики също могат да се зададат (Фиг. 9.60):

- изпълнител,
- предмет,
- описание,
- приоритет (нисък, среден, висок, много висок),
- статус (нестартирана, в процес на изпълнение, изчакваща, завършена),
- начална дата,
- крайна дата,
- степента на завършеност на задачата в проценти.



Фигура 9.60. Създаване на задача в проект

Не забравяйте да зададете име и начална дата на задачата. След като въведете данните, натиснете бутона Create.

Съобщението „*The task is not assigned yet.*" се появява, когато задачата все още няма назначен изпълнител. При успешна транзакция на екрана се извежда утвърдително съобщение (Фиг. 9.61).

Фигура 9.61. Резултат от създаване на задача в проект

Редакция и изтриване на задача на проект.

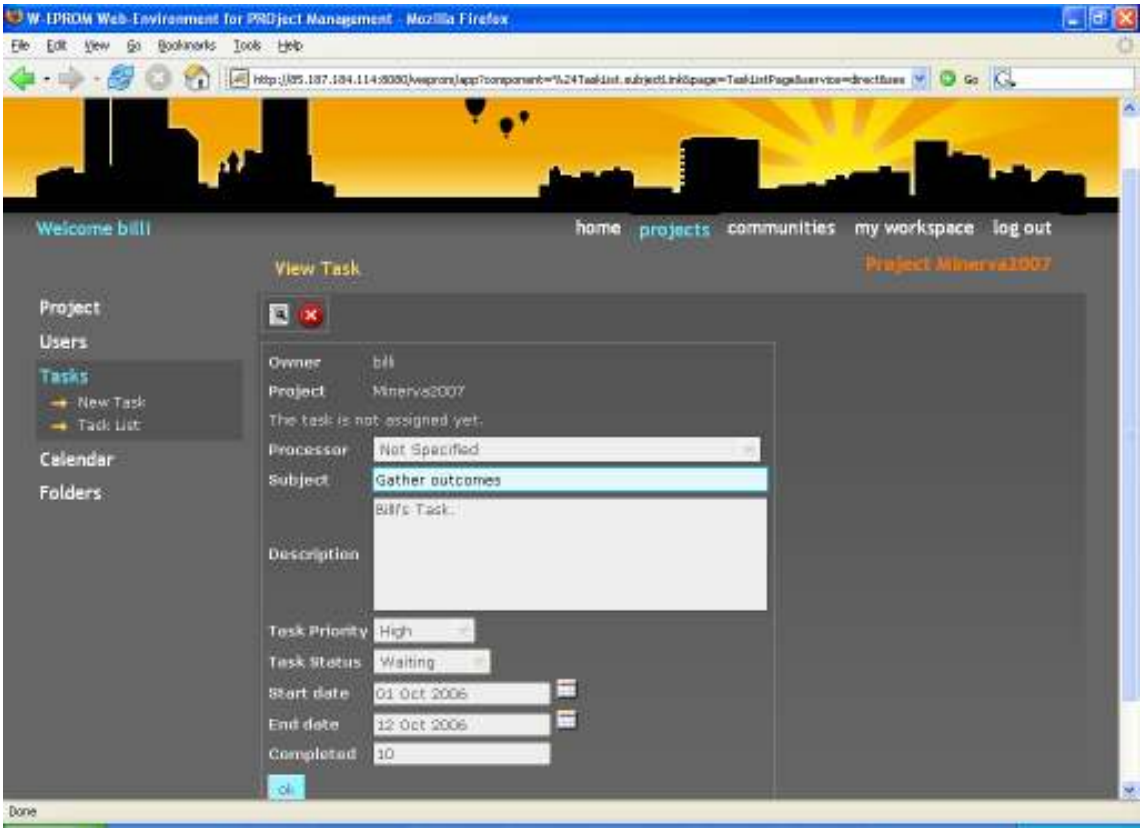
Задачите, дефинирани в даден проект, са достъпни от Tasks Task List.

Фигура 9.62. Резултат от създаване на задача в проект

От екрана със списъка на задачите (Фиг. 9.62), можете да достъпите всяка една от тях, като я изберете по предмет от таблицата. Така достигате част от WEPROM, която Ви позволява редактиране на свойствата на избраната задача или изтриването ѝ от проекта.

Вие може да редактирате задача, само ако сте неин собственик или изпълнител, а можете да изтриете задача, само ако сте неин собственик.

Задачите могат да се сортират по предмет, начална и крайна дата, приоритет, изпълнител, статус и степен на изпълнение за по голямо удобство.



Фигура 9.63. Редактиране на задача в проект

За режим на редактиране, натиснете бутона  (Фиг. 9.63),. След като редактирате която характеристика пожелаете, освен предмета на задачата, натиснете бутона Update.

При успешна транзакция на екрана се извежда утвърдително съобщение (Фиг. 9.64)

Фигура 9.64. Резултат от редактиране на задача в проект

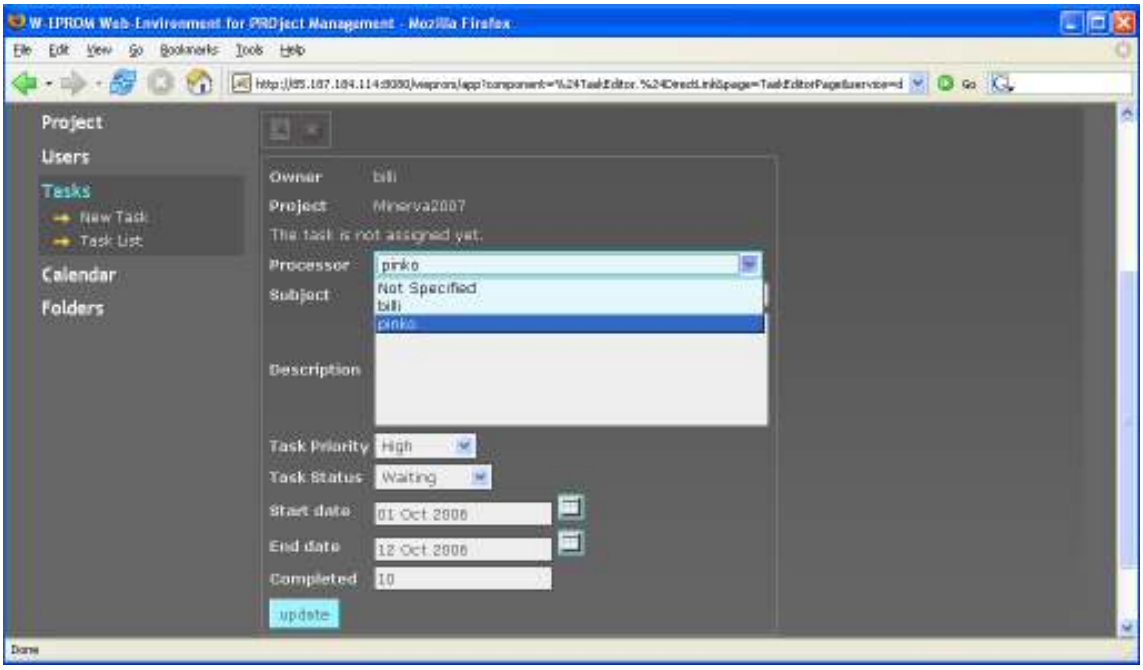
За режим на изтриване, натиснете бутона  . А след това и „delete”. При успешна транзакция на екрана се извежда утвърдително съобщение (Фиг. 9.65).

Фигура 9.65. Резултат от изтриване на задача в проект

Възлагане на задача.

Да обърнем по-специално внимание на редактирането на изпълнителя на една задача или в терминологията на WEPROM- процеса на възлагане на задача.

Може да изберете изпълнител на задачата от падащия списък, който съдържа всички членове на текущия проект. След това натиснете бутона Update (Фиг. 9.66).



Фигура 9.66. Възлагане на задача в проект

На екрана се извежда утвърдително съобщение при успешна транзакция (Фиг. 9.67).

Фигура 9.67. Резултат от възлагане на задача в проект

В списъка със задачи, вече се виждат последните промени- за изпълнител е избран потребителят pinko (Фиг. 9.68).

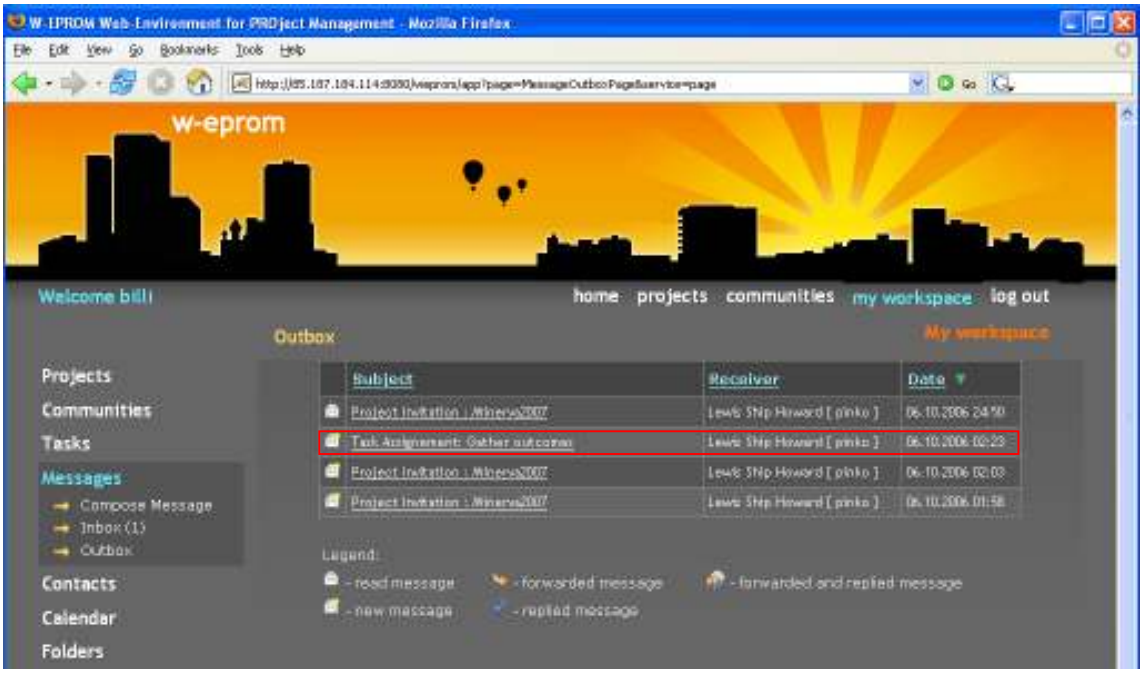
Фигура 9.68. Резултат от възлагане на задача в проект

Ако отново изберете задачата за редакция, виждате, че съобщението, касаещо възлагането й се е променило на „*The task assignment is pending. Answer to task assignment message is expected from: pinko*” (Фиг. 9.69).

Фигура 9.69. Резултат от възлагане на задача в проект

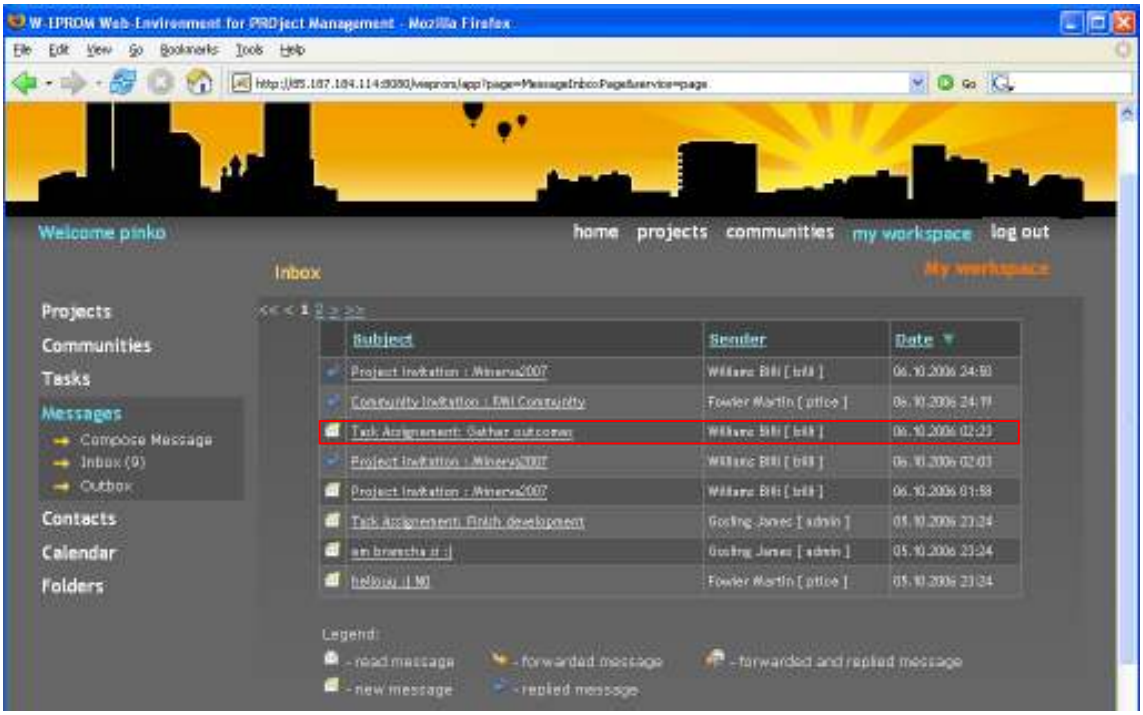
Това е така, защото скрито от потребителя, системата е изпратила **съобщение- заявка за възлагане на задача.**

Това съобщение е изходящо за координатора- той е изпращач (Фиг. 9.70)-



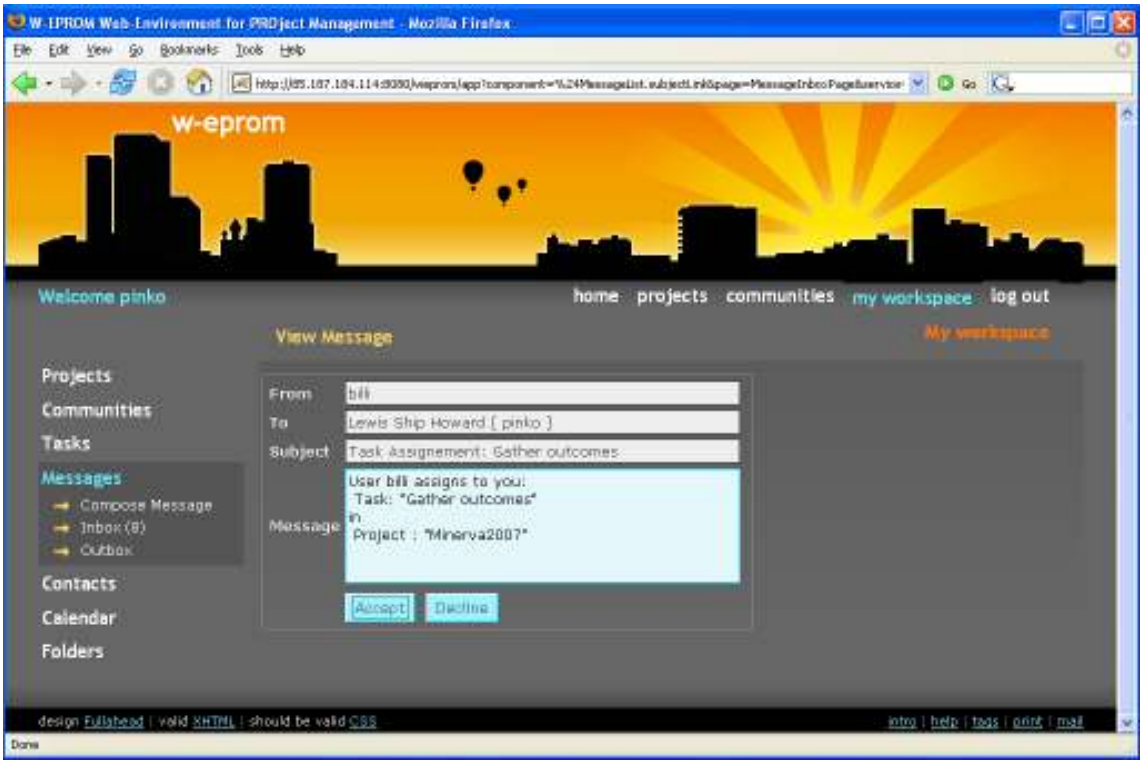
Фигура 9.70. Изпратено системно съобщение за възлагане на задача

входящо за евентуалния изпълнител- той е получател (Фиг. 9.71)-



Фигура 9.71. Получено системно съобщение за възлагане на задача

и има твърдо зададени предмет и текст (Фиг. 9.72).



Фигура 9.72. Съдържание на съобщение за възлагане на задача

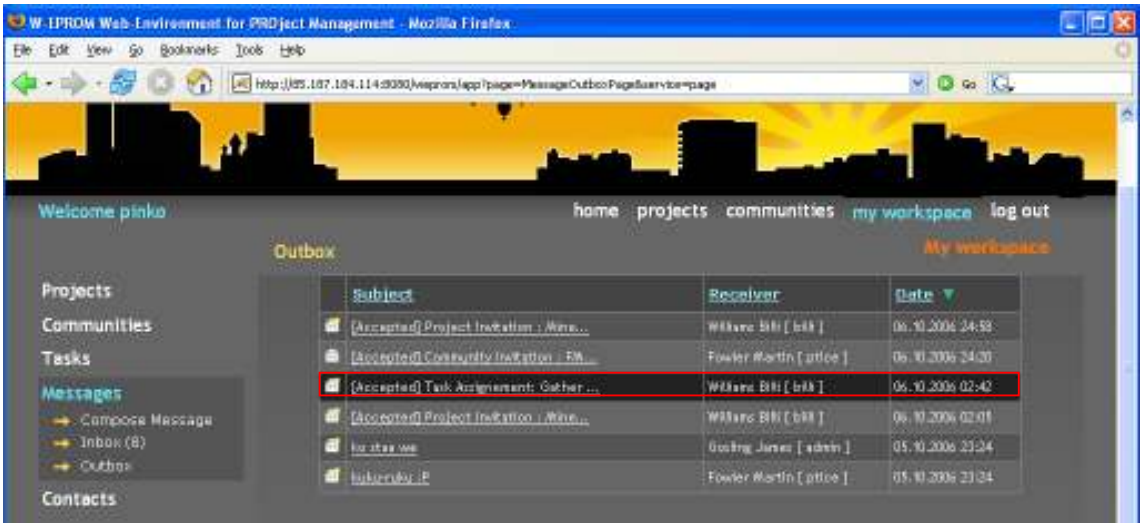
Когато евентуалният изпълнител прочете **съобщението- заявка**, той има две възможности- да приеме или да откаже поканата (Фиг. 9.72). Ако приеме с натискане на бутона Accept, задачата се възлага на него, добвя се към неговия списък със задачи и преминава в състояние „възложена" (Фиг. 9.73).

Фигура 9.73. Приемане на възлагането на задача

Фигура 9.74. Възложена задача

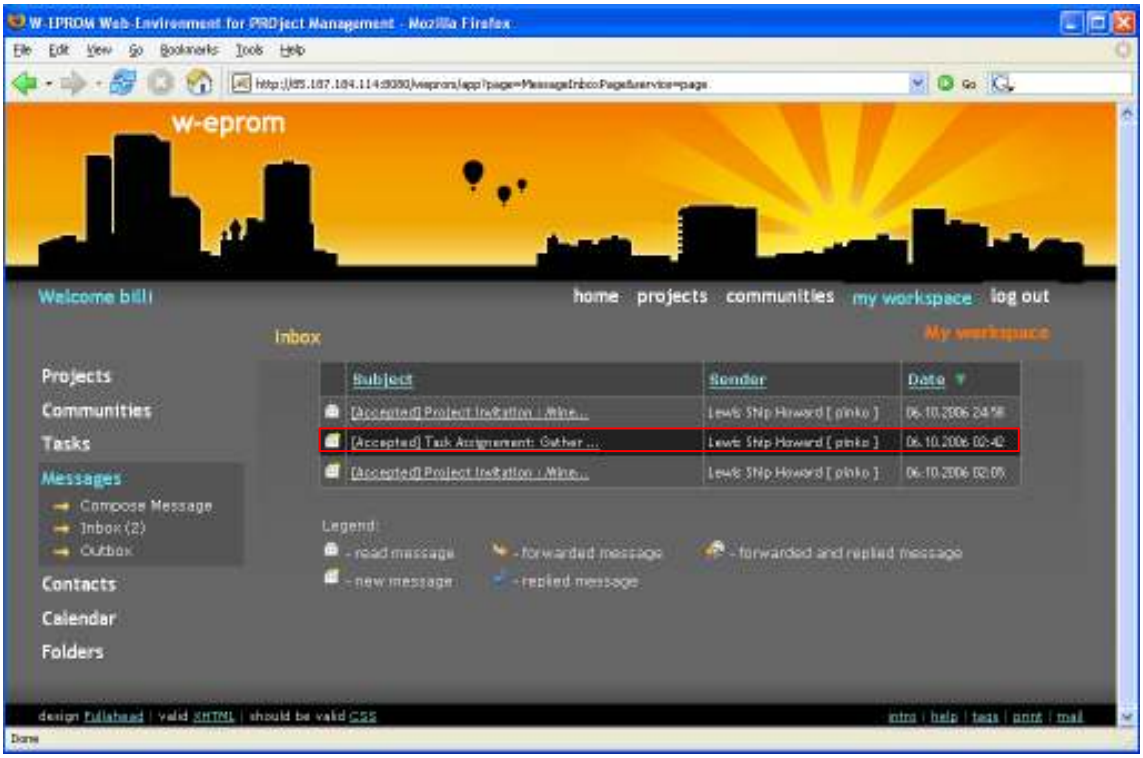
Съобщението, касаещо възлагането на задачата се е променило на „*The task is assigned to: pinko*" (Фиг. 9.74). Ако потребителят откаже съответно чрез Decline, задачата остава без изпълнител и в състояние „невъзложена".

И в двата случая обаче, се изпраща ново системно **съобщение с дадения отговор** до координатора на проекта, който първоначално е изпратил поканата. Това съобщение е изходящо за поканения потребител - той е изпращач (Фиг. 9.75)-



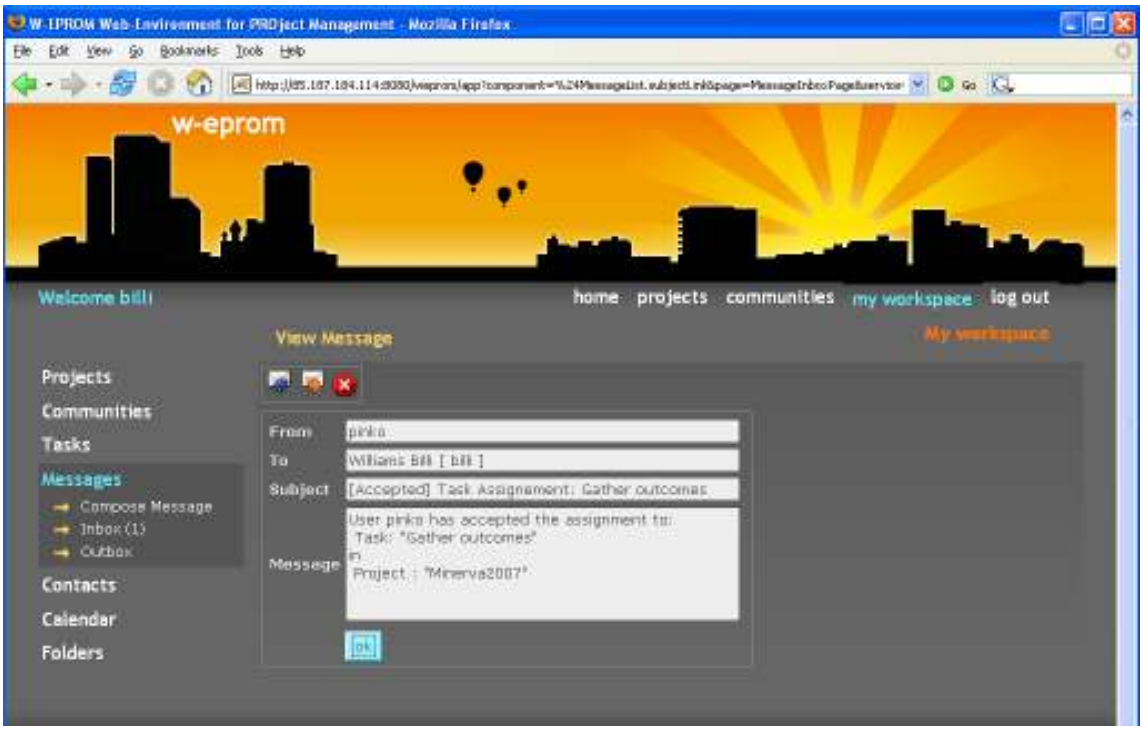
Фигура 9.75. Изпратено съобщение за съгласие с възложена задача

входящо за възложителя - той е получател (Фиг. 9.76)-



Фигура 9.76. Изпратено съобщение за съгласие с възложена задача

и има твърдо зададени предмет и текст (Фиг. 9.77).



Фигура 9.77. Изпратено съобщение за съгласие с възложена задача

9.1.5.4. Календар на проекта.

Създаване на нова паметка в проект.

Нова паметка в работното пространство на проект може да бъде създадена само от негов член-координатор. Членовете- участници не могат да създават паметки в проекта.

Процедурата на създаване е аналогична на тази за паметките в собственото работно пространство. За повече информация- тук.

Редакция и изтриване на паметка в проект.

Само член-координатор на проект може да редактира или изтрива паметки в работното пространство на проекта. Членовете- участници могат само да разглеждат паметките в редактора.

Процедурата на редакция и изтриване е аналогична на тази за паметките в собственото работно пространство. За повече информация- тук.

Разглеждане на всички паметки на проект.

Ако желаете да разгледате всички паметки в проекта наведнъж, можете да го направите през Calendar Project Reminders(Фиг. 9.78).

Фигура 9.78. Списък с паметките на проект

Разглеждане на календара.

Календарът на проект е достъпен през Calendar Project Calendar.

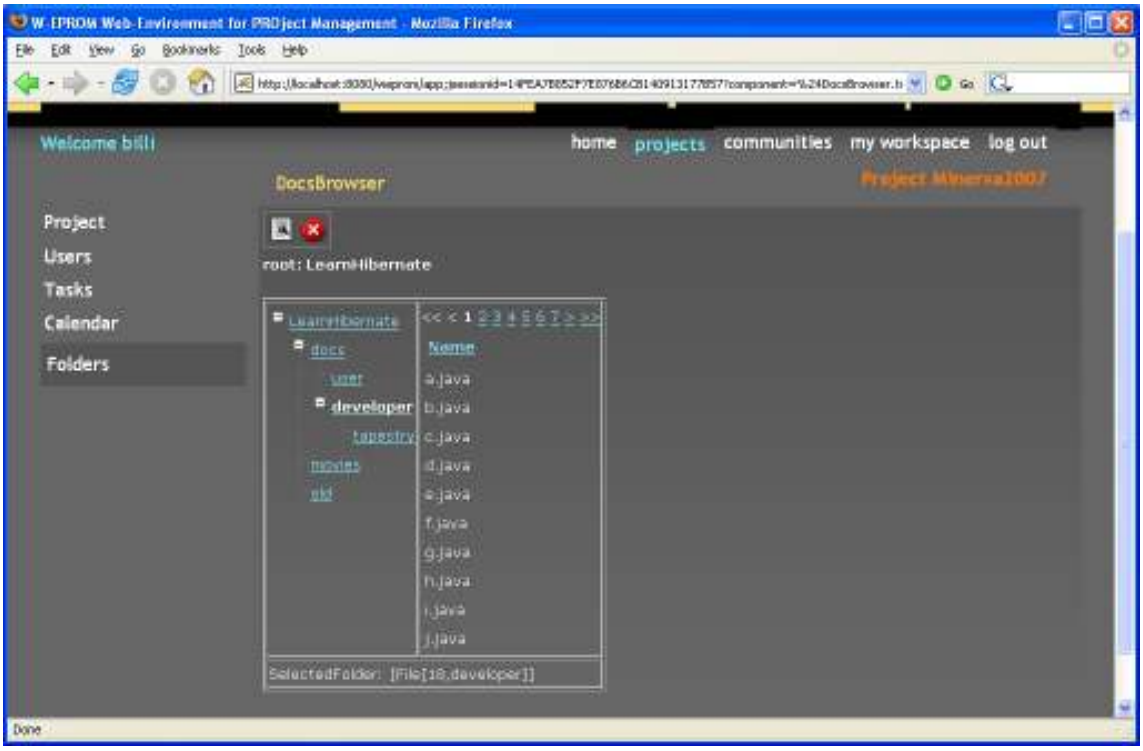
В него са отбелязани нагледно дните, в които има записани паметки. Освен това, чрез падащите менюта за месец и година и бутоните до тях, можете да се „движите” по календара (Фиг. 9.79).

Фигура 9.79. Календар на проект

Всеки член на проекта може да се консултира с календара, но само координаторите могат да създават, редактират или изтриват информация от него.

9.1.5.5. Папки и документи

Папките и документите са достъпни през Folders. Те се менажират чрез екран, наподобяващ на „експлорър”. В лявата му част са изброени папките на проекта в дървовидна структура, а дясната съдържа списък със съдържанието на текущо избраната папка (Фиг. 9.80).



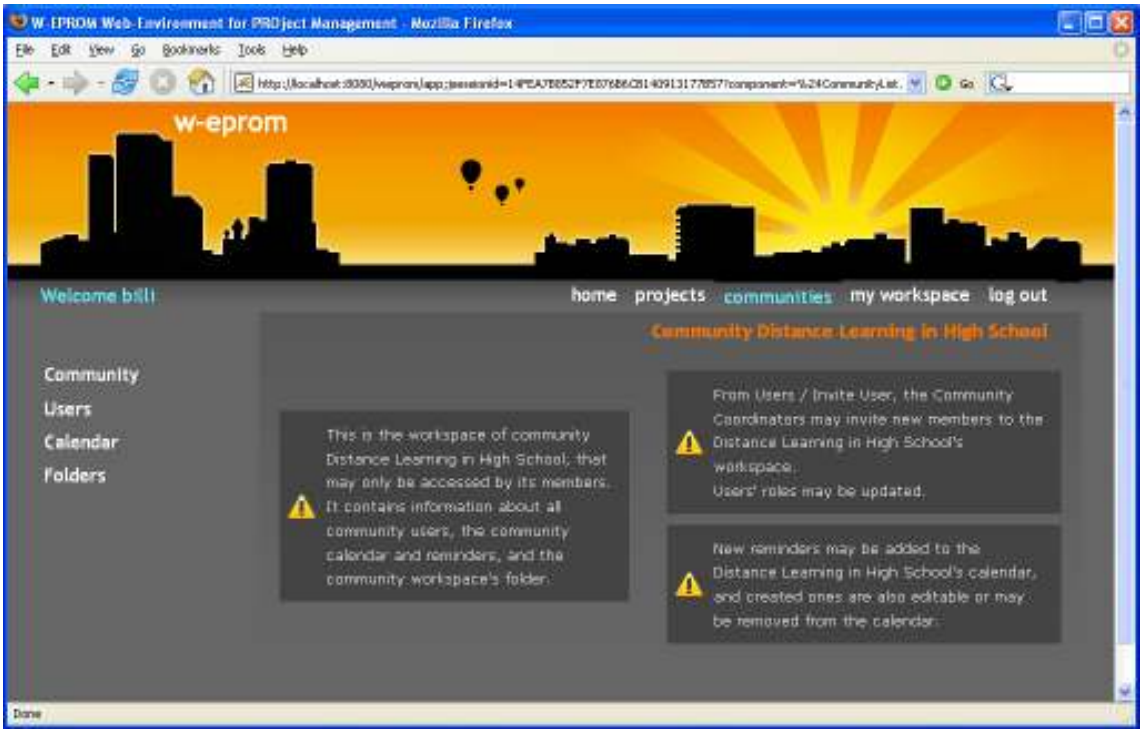
Фигура 9.80. Папки и документи на проект

9.1.6. . Управление на група по интереси.

От екрана със списъка с вашите групи по интереси, можете да достъпите всяка една от тях, като я изберете по име от таблицата (Фиг. 9.81).

Фигура 9.81. Списък с групи по интереси

Така влизате в работното пространство на тази група (Фиг. 9.82).



Фигура 9.82. Работно пространство на група по интереси

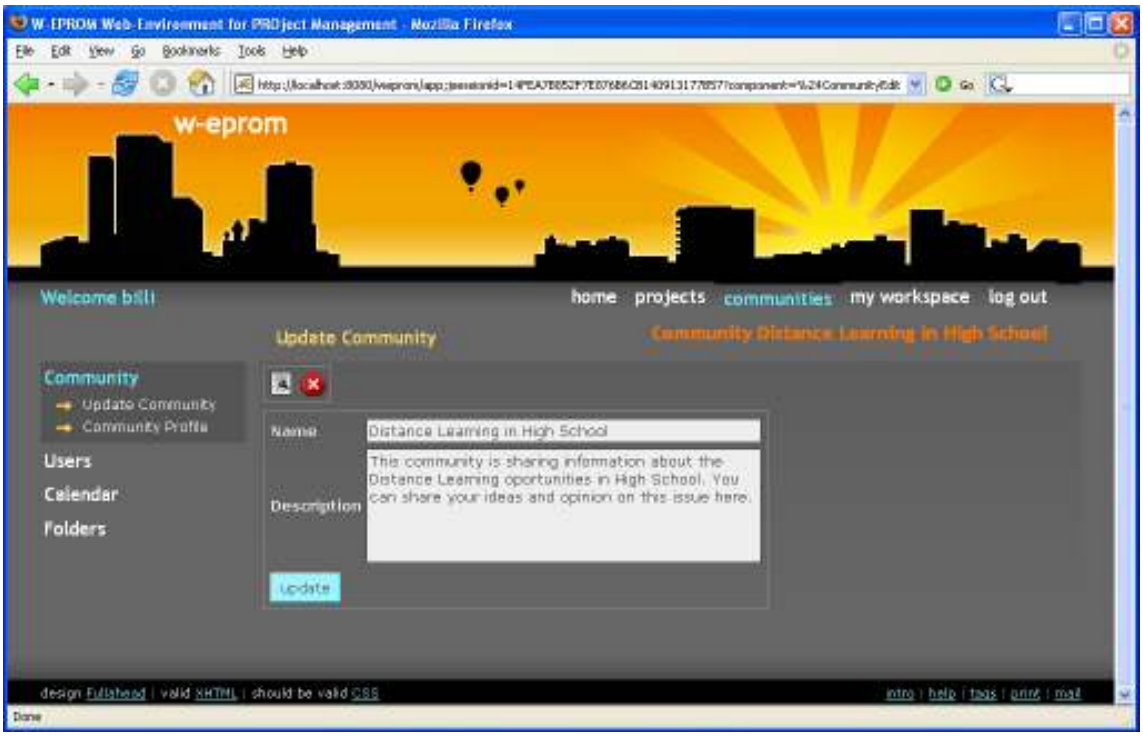
Дадена група може да бъде достъпена само от членовете й. Тя съдържа информация за всички тях и разполага също така с календар и главна папка.

9.1.6.1. Профил на групата по интереси.

Профилът на групата по интереси е достъпен за разглеждане от Community Community Profile (Фиг. 9.83)

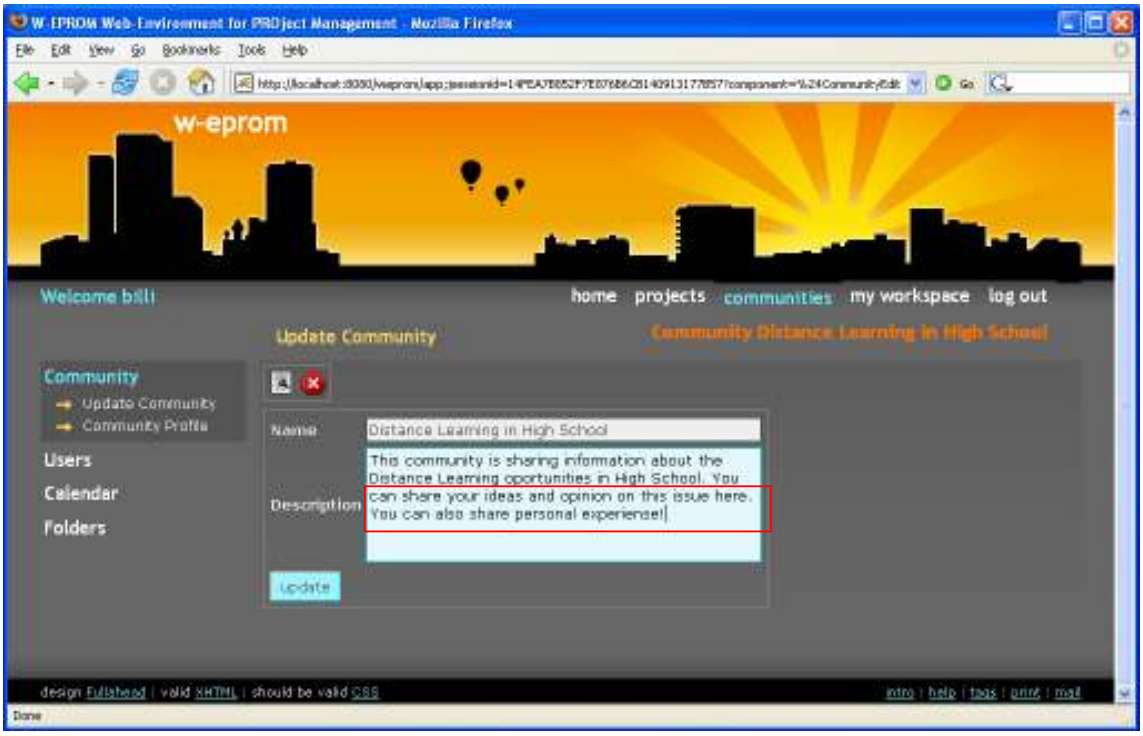
Фигура 9.83. Профил на група по интереси

а за редакция от Community Update Community бутона  (Фиг. 9.84).



Фигура 9.84. Редакция на профил на група по интереси

След редактиране на данните, натиснете бутона Update (Фиг. 9.85).



Фигура 9.85. Редакция на профил на група по интереси

Ако транзакцията е успешна, на екрана се изписва утвърдително съобщение (Фиг. 9.86).

Фигура 9.86. Резултата от редакция на профил на група по интереси

Ако разгледате отново профила на групата, ще видите промените (Фиг. 9.87).

Фигура 9.87. Редактиран профил на група по интереси

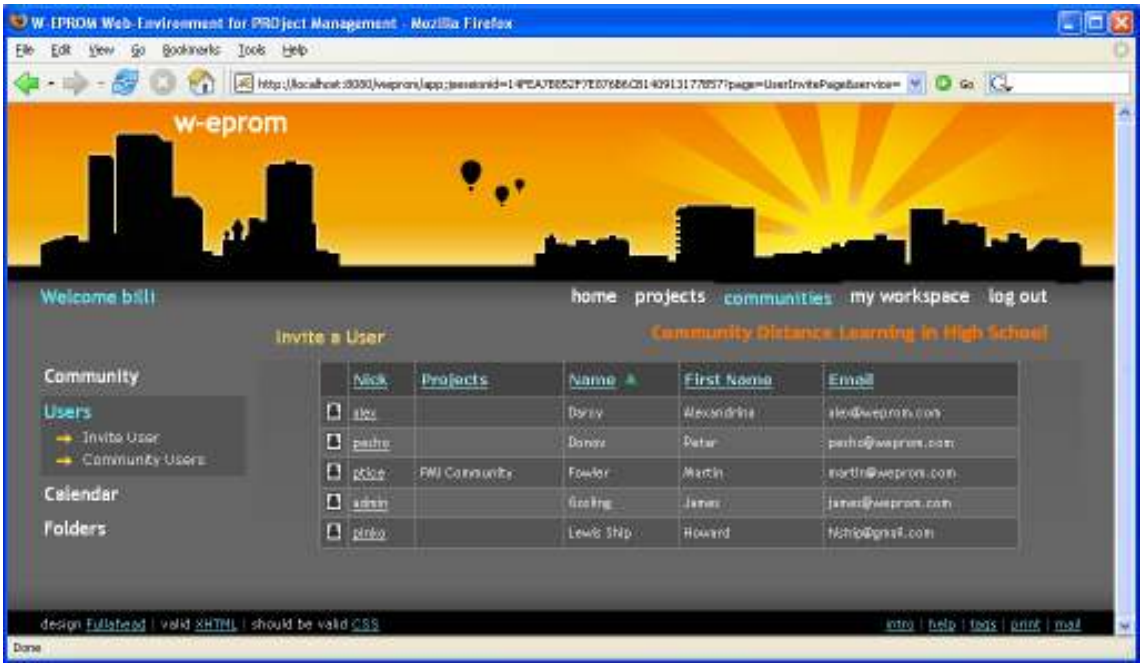
9.1.6.2. Членове на група по интереси.

Изпращане на покана до потребител за членство в група.

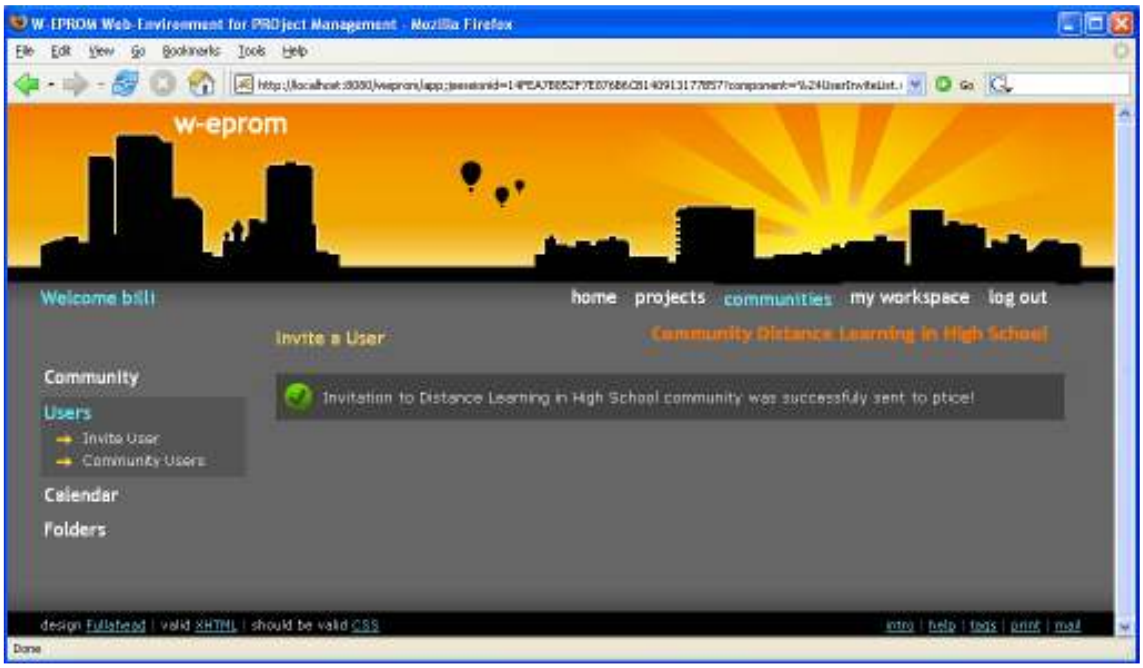
Ако сте координатор, за да изпратите покана до потребител за членство в група по интереси, изберете Users Invite User. Извежда се списък с всички регистрирани потребители на системата, които не са членове на текущата група (Фиг. 9.88).

При „кликване" на нечие потребителско име от таблицата (в случая на ptice), автоматично се изпраща системно **съобщение- покана за присъединяване към групата по интереси**(Фиг. 9.89).

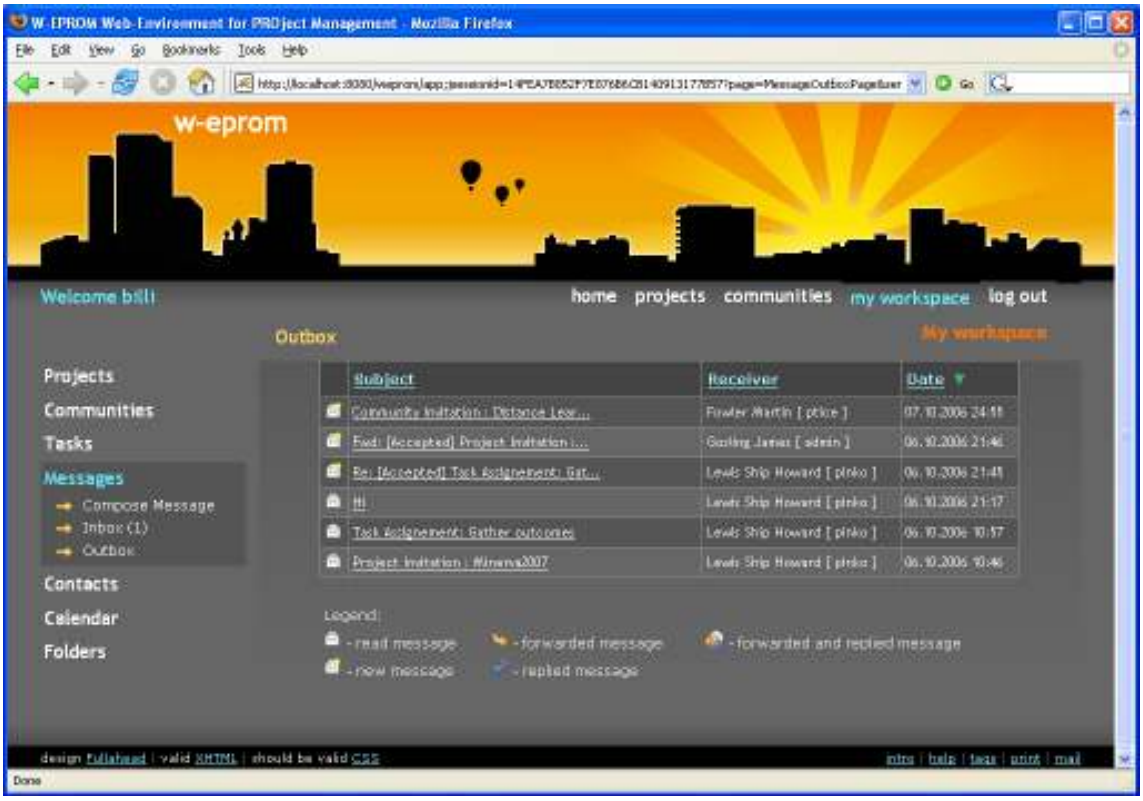
Това съобщение е изходящо за координатора, който е изпращач (Фиг. 9.90), входящо за поканения потребител, който е получател (Фиг. 9.91) и има твърдо зададени предмет и текст (Фиг. 9.92).



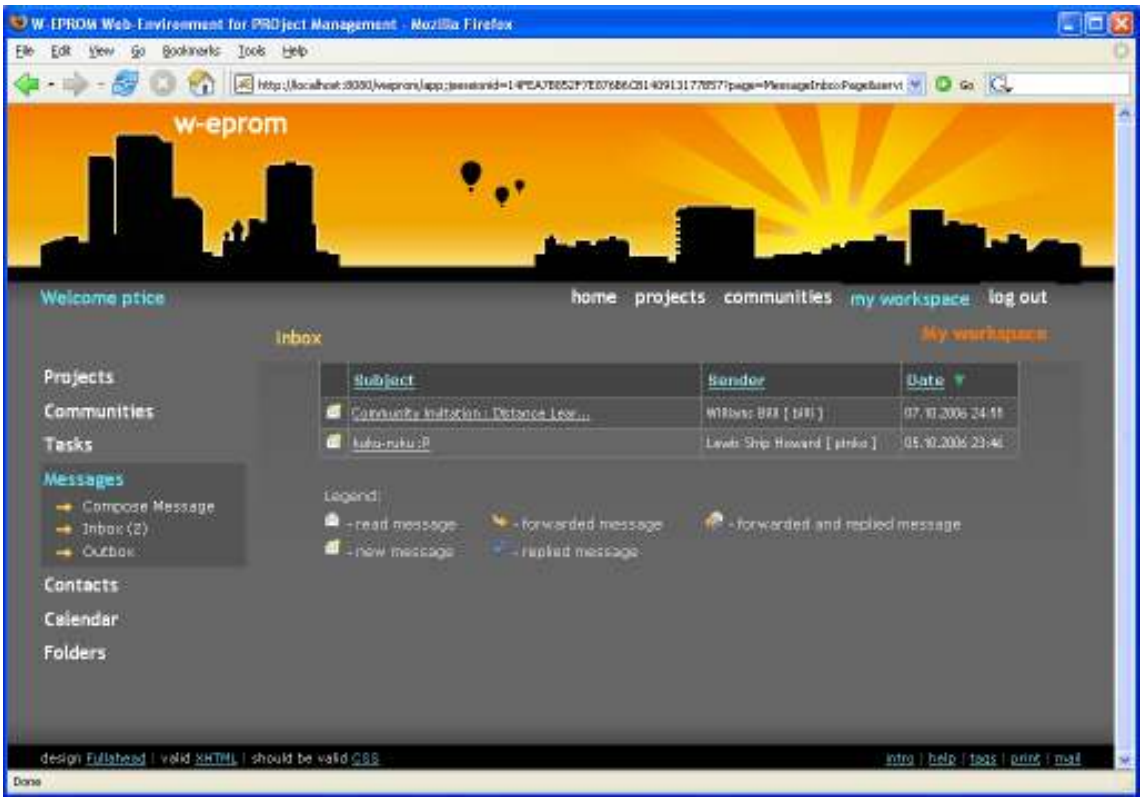
Фигура 9.88. Списък с потребители, които не членуват в група***та***



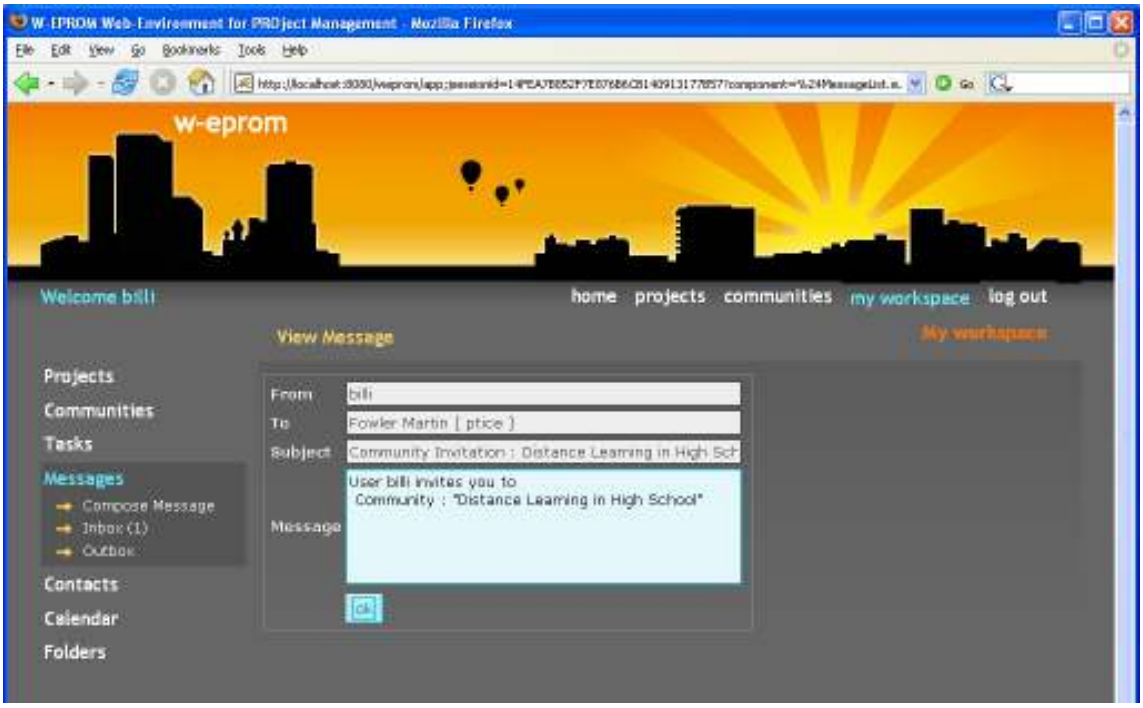
Фигура 9.89. Изпращане на покана за членство в група



Фигура 9.90. Изпратена покана за членство в група

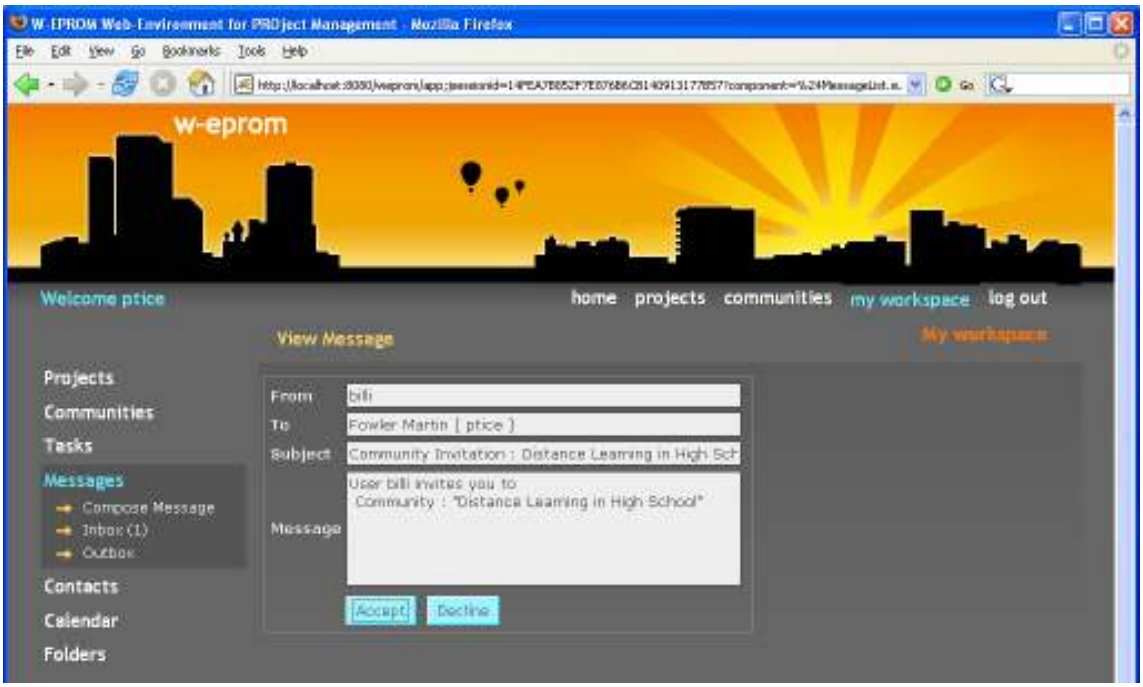


Фигура 9.91. Получена покана за членство в група



Фигура 9.92. Съобщение- покана за членство в група

Когато поканеният потребител прочете **съобщението- покана**, той има две възможности- да приеме или да откаже поканата (Фиг. 9.93). Ако приеме с натискане на бутона Accept, той се добавя към членовете на групата(Фиг. 9.94).



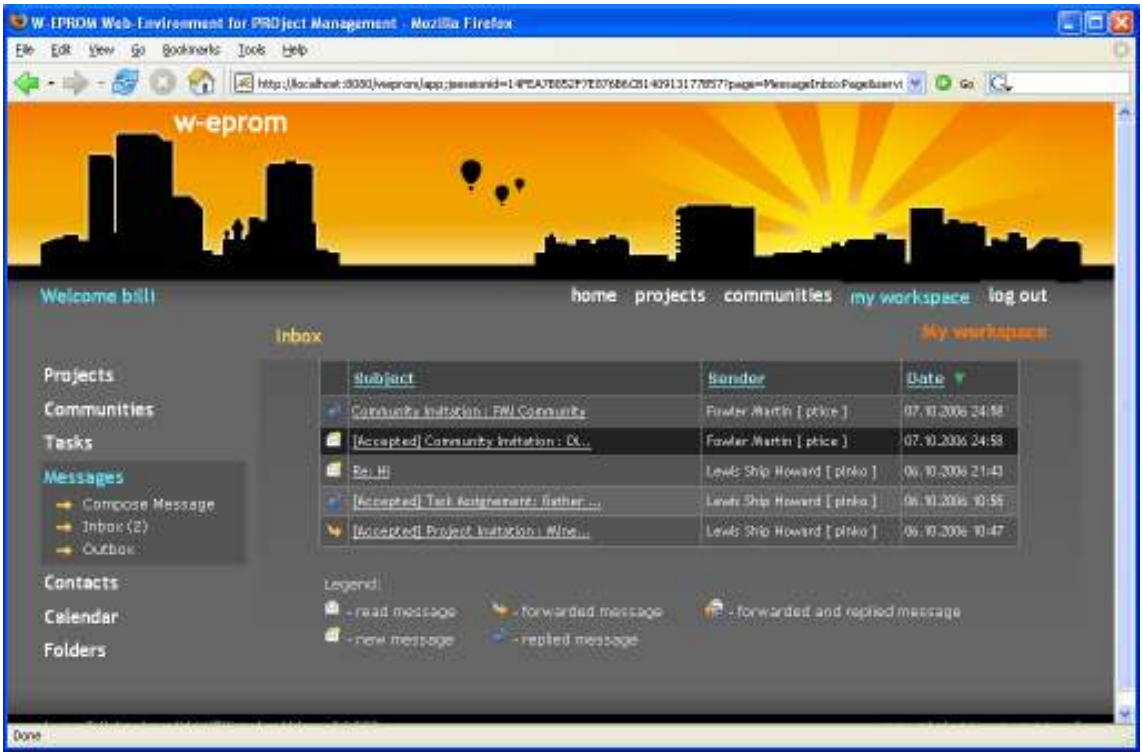
Фигура 9.93. Приемане на покана за членство в група

Фигура 9.94. Добавяне на член в група

Ако откаже съответно чрез Decline, не се добавя към тях. И в двата случая обаче, се изпраща ново системно **съобщение с дадения отговор** до координатора на групата по интереси, който първоначално е изпратил поканата. Това съобщение е изходящо за поканения потребител - той е изпращач (Фиг. 9.95)-

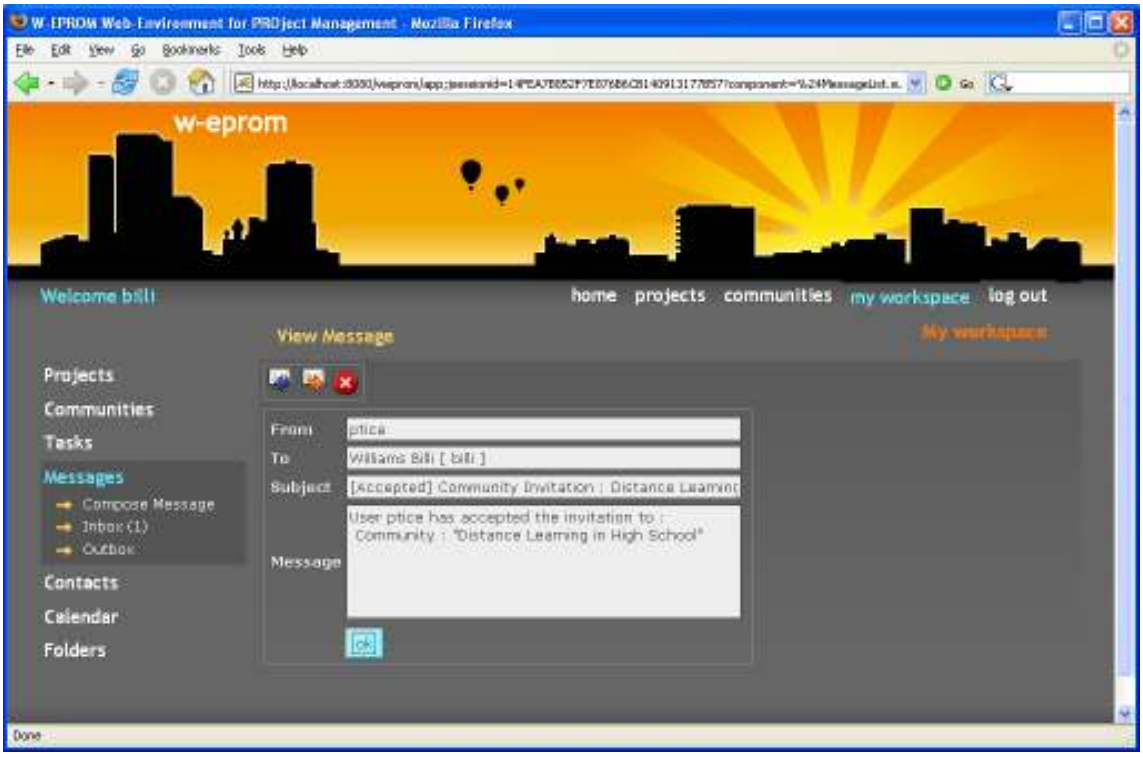
Фигура 9.95. Изпратен отговор на покана за членство в група

входящо за координатора - той е получател (Фиг. 9.96)-



Фигура 9.96. Получен отговор на покана за членство в група

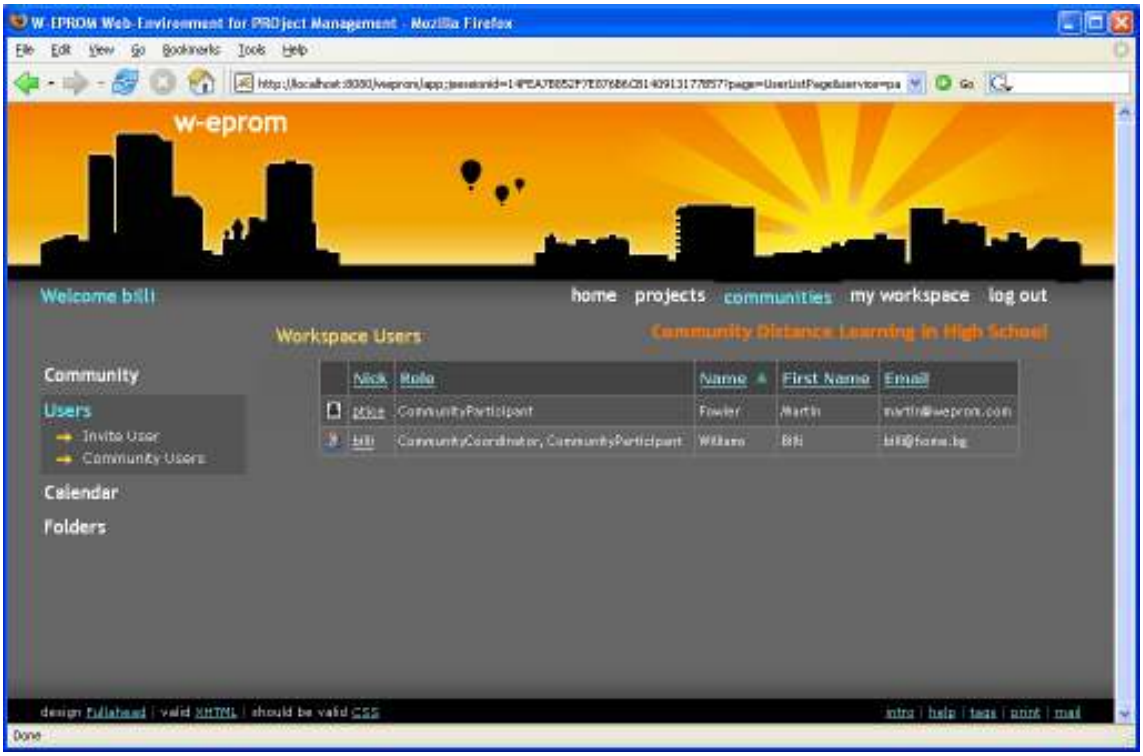
и има твърдо зададени предмет и текст (Фиг. 9.97).



Фигура 9.97. Отговор на покана за членство в група

Редакция и изтриване на член на група.

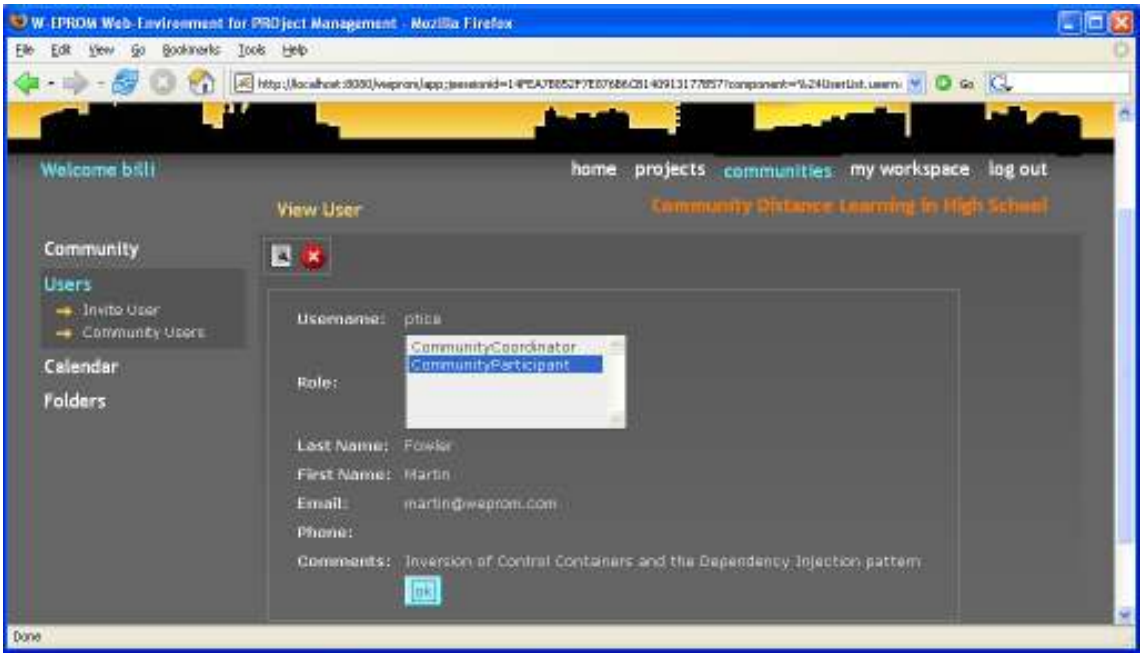
Членовете на група по интереси са достъпни от Users Community Users (Фиг. 9.98).



Фигура 9.98. Списък с членове на група

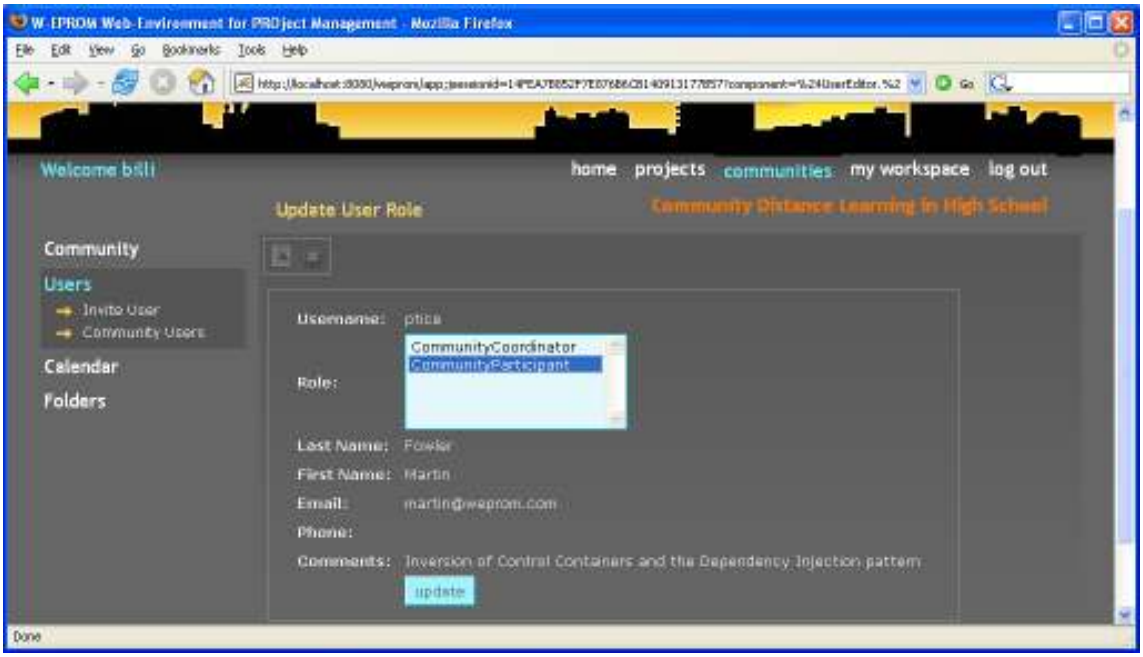
Вие може да сортирате членовете на групата по потребителско име, роля, фамилия, име и електронен адрес.

От екрана със списъка на членовете, можете да достъпите всеки един от тях, като го изберете по потребителско име от таблицата. Така достигате потребителския редактор, който Ви позволява редактиране на ролите на съответния член или премахването му от групата (Фиг. 9.99). Двете операции са позволени само за координаторите на групата.



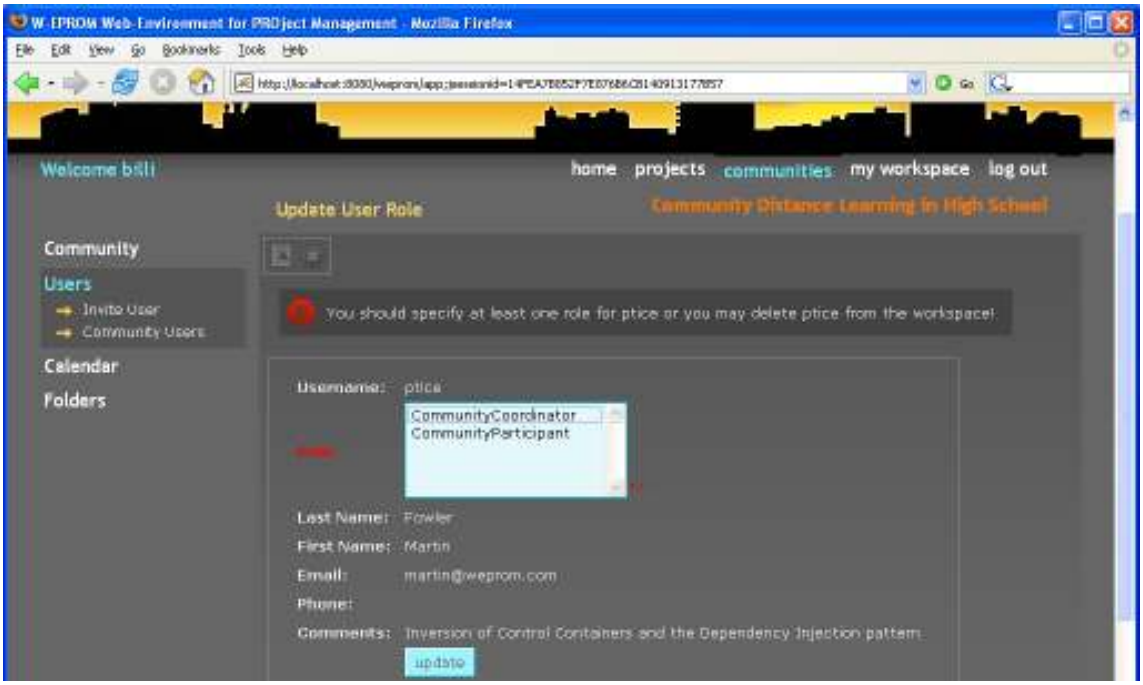
Фигура 9.99. Редакция на член на група

За режим на редактиране, натиснете бутона  (Фиг. 9.100).



Фигура 9.100.Редакция на член на група

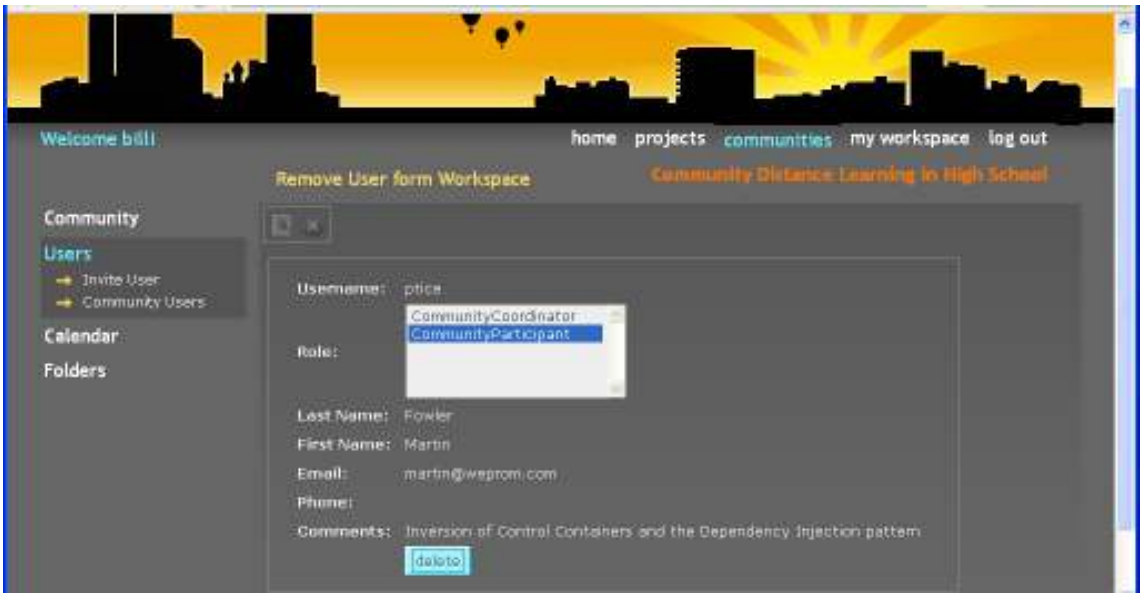
Редактирайте ролите и натиснете бутона Update. Ако нито една роля не е избрана, на екрана се появява съобщение за грешка (Фиг. 9.101).



Фигура 9.101.Грешка при редакция на член на група

*Тъй като ролята „**координатор на група по интереси**” включва в себе си „**участник в група по интереси**”, не е нормално да е избрана само „**координатор на група по интереси**”. Затова „**участник в група по интереси**” не се изтрива, дори и да не е селектирана.*

За режим на изтриване, натиснете  (Фиг. 9.102), а след това и бутона delete.



Фигура 9.102.Изтриване на член на група

9.1.6.3. Календар на групата по интереси.

Създаване на нова паметка в група.

Нова паметка в работното пространство на група по интереси може да бъде създадена само от неин член-координатор. Членовете- участници не могат да създават паметки в групата.

Процедурата на създаване е аналогична на тази за паметките в собственото работно пространство. За повече информация- виж 4.3.1.

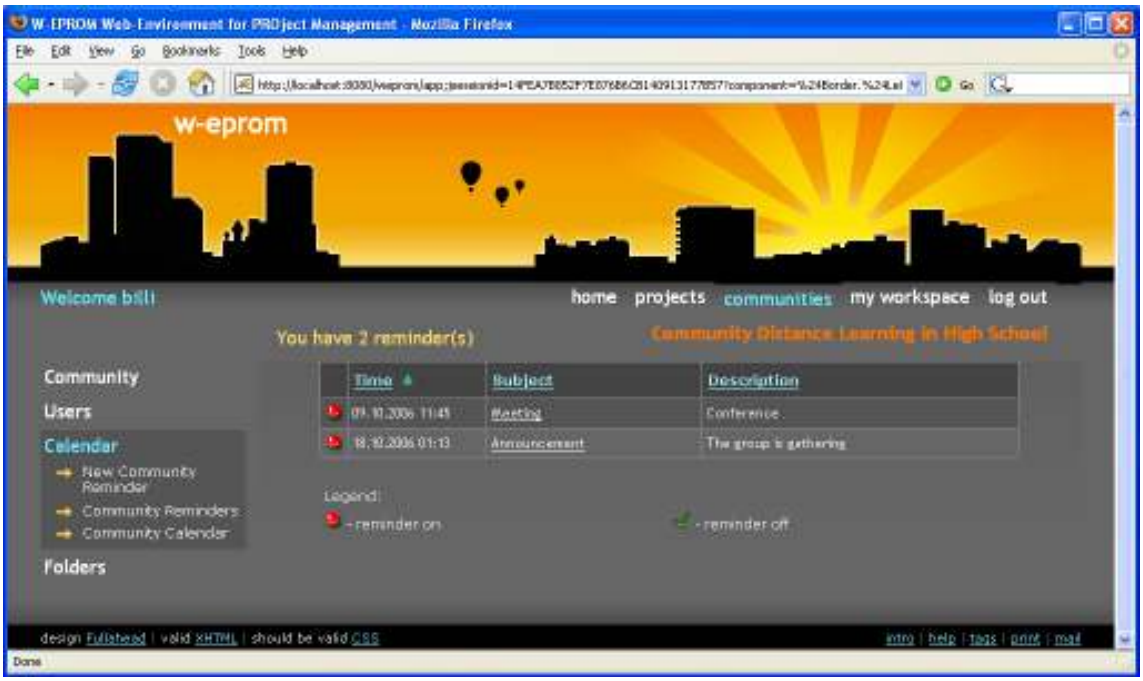
Редакция и изтриване на паметка в група.

Само член-координатор на група може да редактира или изтрива паметки в работното й пространство. Членовете- участници могат само да разглеждат паметките в редактора.

Процедурата на редакция и изтриване е аналогична на тази за паметките в собственото работно пространство. За повече информация- виж 4.3.2.

Разглеждане на всички паметки на група.

Ако желаете да разгледате всички паметки в групата наведнъж, можете да го направите през Calendar Community Reminders (Фиг. 9.103).

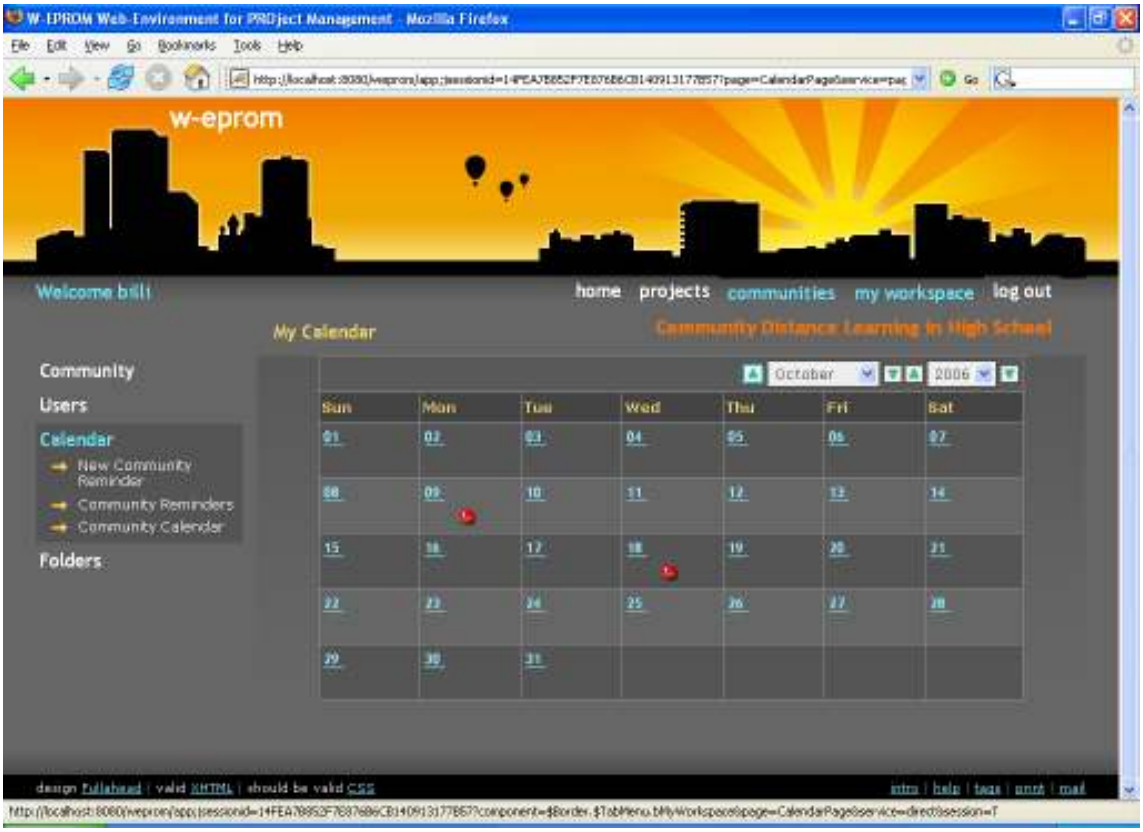


Фигура 9.103. Паметки на групата по интереси

Разглеждане на календара.

Календарът на група по интереси е достъпен през Calendar Community Calendar.

В него са отбелязани нагледно дните, в които има записани паметки. Освен това, чрез падащите менюта за месец и година и бутоните до тях, можете да се „движите” по календара (Фиг. 9.104).

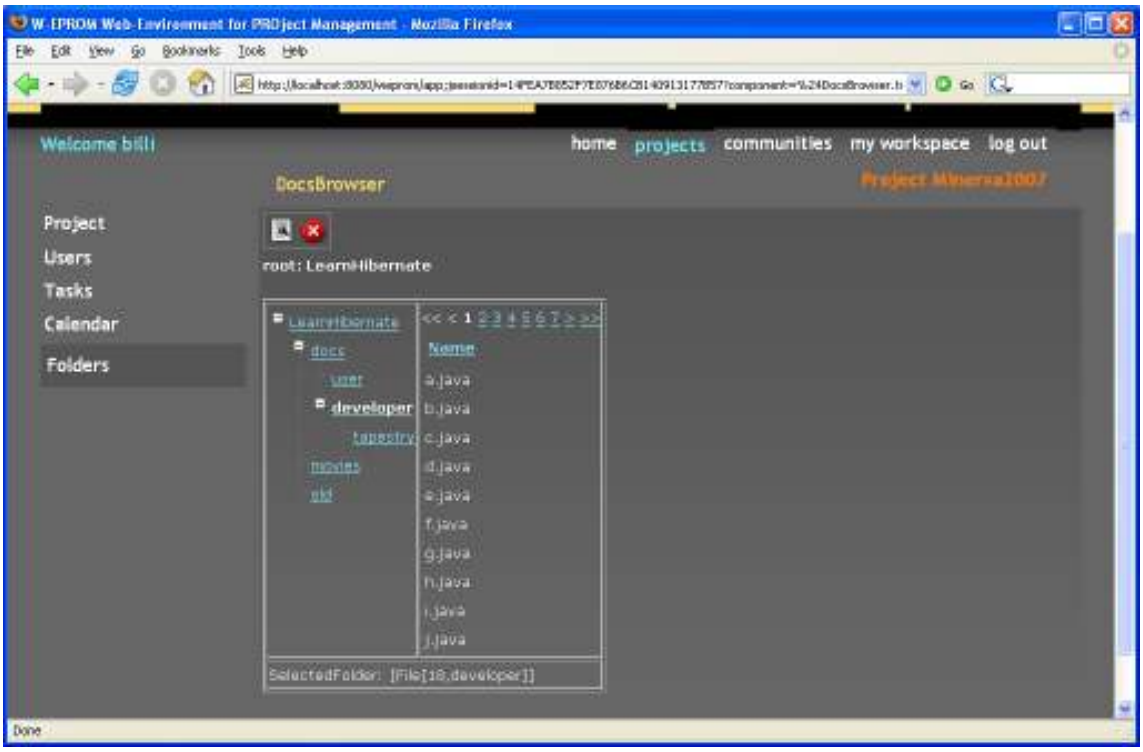


Фигура 9.104. Календар на групата по интереси

Всеки член на групата може да се консултира с календара, но само координаторите могат да създават, редактират или изтриват информация от него.

9.1.6.4. Папки и документи

Папките и документите са достъпни през Folders. Те се менажират чрез екран, наподобяващ на „експлорър”. В лявата му част са изброени папките на проекта в дървовидна структура, а дясната съдържа списък със съдържанието на текущо избраната папка (Фиг. 9.105).



Фигура 9.105. Папки и документи на групата по интереси

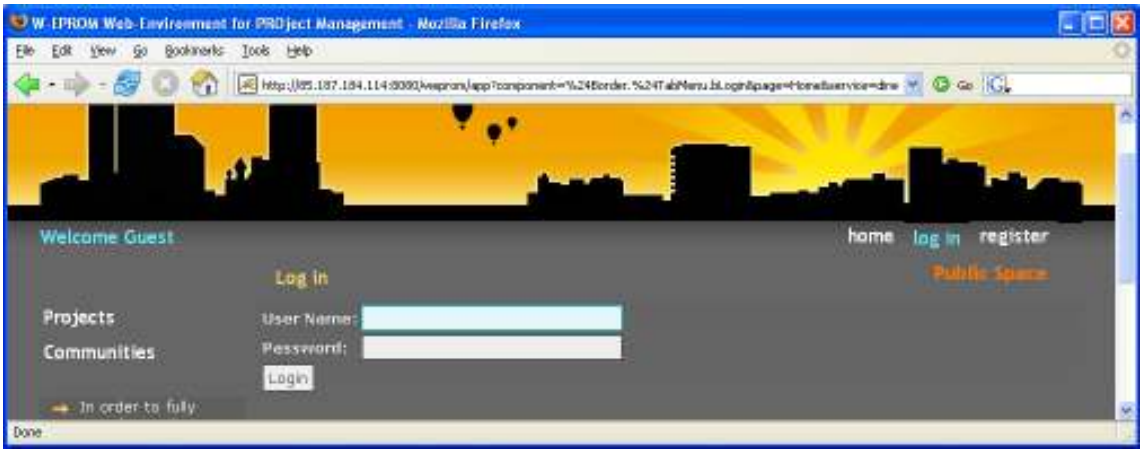
9.1.7. Хоризонтално меню за бърз достъп и навигация

Хоризонталното меню в горната дясна част на екрана има два режима. Ако потребителят не е влезнал в WEPROM, то се състои от следните бутони:

- - *начало* (home) – за показване на началната страница на приложението (Фиг. 9.106),

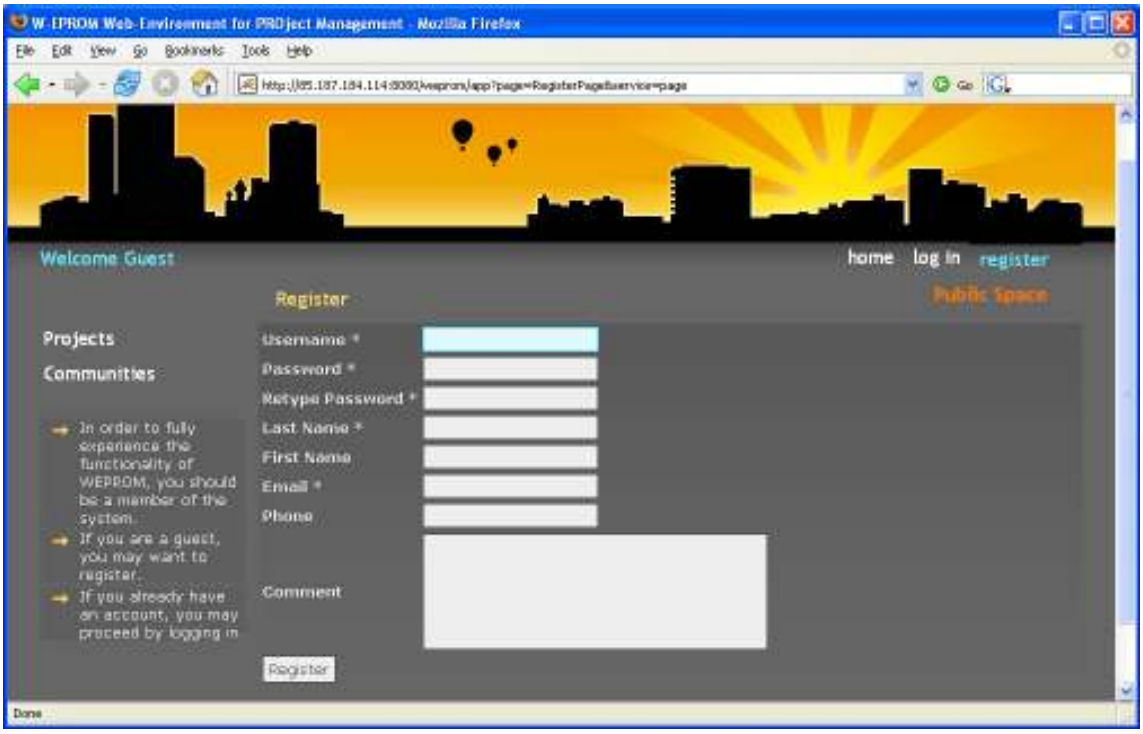
Фигура 9.106. Начална страница на приложението

- - *вход* (login) - за влизане в системата (Фиг. 9.107),



Фигура 9.107. Логин

- *регистрация* (register) – за регистрация (Фиг. 9.108)



Фигура 9.108. Регистрация

Ако потребителят е „логнат" в системата, то се състои от следните бутони:

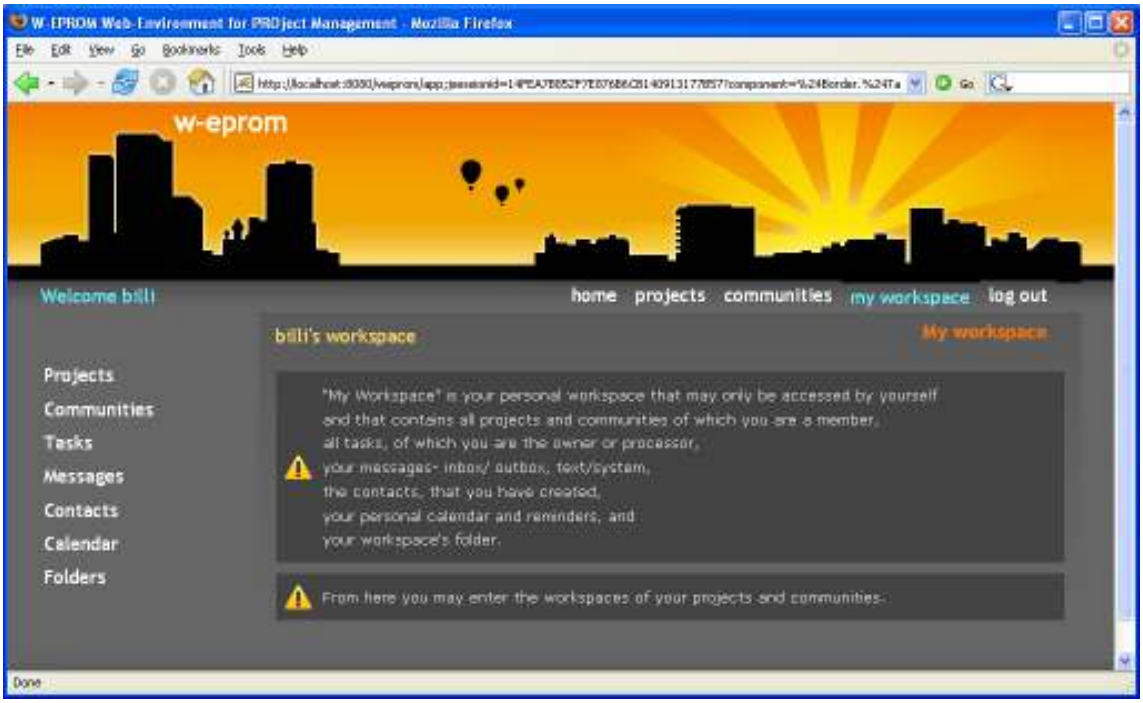
- - *начало* (home) – за показване на началната страница на приложението,
- - *проекти* (projects) – за достъп до списъка с проектите, на които потребителят е член (Фиг. 9.109),

Фигура 9.109. Проекти

- - *групи по интереси* (communities) – за достъп до списъка с групи по интереси, на които потребителят е член (Фиг. 9.110),

Фигура 9.110.Групи по интереси

- - *моето работно пространство* (my workspace) – за достъп до личното работно пространство (Фиг. 9.111),



Фигура 9.111.Лично работно пространство

- - *изход* (logout) - за излизане от системата,

9.2. Листинги на части от програмния код на слоя бизнес логика

Поради значителния размер на слоя бизнес логика (над 8000 реда Java код), тук са изложени само класовете и интерфейсите на обектите от проблемния домейн (пакет *com.fmi.weprom.api*) и тези на абстракцията на цикъла заявка-отговор (пакет *com.fmi.weprom.util*). Т.е. всички класове и интерфейси от които има нужда презентационният слой.

9.2.1. Пакет com.fmi.weprom.api

9.2.1.1. AccessRights.java

```
package com.fmi.weprom.api;

import java.util.List;

public interface AccessRights {

  public void grantUserRole(User user, Role r, Workspace w) throws WEPROMException;

  public void grantUserRoles(User user, List roles, Workspace w) throws
WEPROMException;

  public void deleteUserRole(User user, Role r, Workspace w);

  public void deleteUserRoles(User user, List roles, Workspace w);

  public void deleteUserRoles(User user, Workspace w);

  public boolean checkPermission(User user, Workspace w, Permission p);

  public List getUserRoles(User user, Workspace w);

  public Permission getPermission(String name);

  public Role getRole(String name);

  public List getUsersWithoutRoles(Workspace w);

  public boolean checkPermission(User user, int workspaceID, Permission p);
}
```

9.2.1.2. Calendar.java

```
package com.fmi.weprom.api;

import java.util.Date;
```

```
import java.util.List;

public interface Calendar {

  public Reminder createReminder();

  public List getReminders();

  public List getReminders(Date startDate, Date endDate);

  public Reminder getReminder(int id);
}
```

9.2.1.3. Community.java

```
package com.fmi.weprom.api;

import java.util.Date;

public interface Community extends Workspace {

  public Calendar getCalendar();

  public void save() throws WEPROMException;

  public void update() throws WEPROMException;

  public void delete() throws WEPROMException;

  public String getName();

  public void setName(String name) ;

  public Date getStartDate();

  public Date getEndDate();

  public String getDescription();

  public int getCompleted();

  public int getStatus();

  public void setStartDate(Date startDate);

  public void setEndDate(Date endDate);

  public void setDescription(String  description);

  public void setCompleted(int completed);
```

```
  public void setStatus(int status);

  public void inviteUser(User user) throws WEPROMException;
}
```

9.2.1.4. Contact.java

```
package com.fmi.weprom.api;

public interface Contact {

  public void save() throws WEPROMException;

  public void update() throws WEPROMException;

  public void delete() throws WEPROMException;

  public String getComment();

  public void setComment(String comment);

  public String getEmail();

  public void setEmail(String email);

  public String getFirstName();

  public void setFirstName(String firstName);

  public String getName();

  public void setName(String name);

  public String getPhone();

  public void setPhone(String phone);

  public int getID();
}
```

9.2.1.5. File.java

```
package com.fmi.weprom.api;

import java.util.Date;
import java.util.List;

import com.fmi.weprom.impl.mysql.WorkspaceImpl;

public interface File {

  public static final byte TYPE_FOLDER = 4;
```

```
  public static final byte TYPE_FILE = 5;

  public String getName();

  public void setName(String name);

  public Date getDate();

  public void setDate(Date date);

  public long getSize();

  public void setSize(long size);

  public String getPath();

  public int getID();

  public boolean isFolder();

  public List getFiles();

  public List getFolders();

  public int getParentID();

  public File createChild(byte type);

  public void save() throws WEPROMException;

  public int getFoldersCount() throws WEPROMException;

  public int getFilesCount() throws WEPROMException;

  public List getFiles(String sortColumn, boolean sortOrder, int offset, int pageSize);

  public File getRoot();

  public Workspace getRootWorkspace();
}
```

9.2.1.6. Message.java

```
package com.fmi.weprom.api;

import java.util.Date;
import java.util.List;

public interface Message {

  public static final int MESSAGE_TYPE_TEXT = 0;
```

```
public static final int MESSAGE_TYPE_PROJECT_INVITATION = 1;
public static final int MESSAGE_TYPE_COMMUNITY_INVITATION = 2;
public static final int MESSAGE_TYPE_TASK_ASSIGNMENT = 3;

public int getID();

public void accept() throws WEPROMException;

public void decline() throws WEPROMException;

public void send() throws WEPROMException;

public void delete() throws WEPROMException;

public void update() throws WEPROMException;

public String getSubject();

public void setSubject(String subject);

public String getBody();

public void setBody(String text);

public int getType();

public void setType(int type);

public Date getDate();

public void setReceiver(User receiver);

public void setSender(User sender);

public User getSender();

public User getReceiver();

public boolean isNew();

public void setNew(boolean b);

public boolean isForwarded();

public void setForwarded(boolean b);

public boolean isReplied();

public void setReplied(boolean b);
```

}

9.2.1.7. Permission.java

```
package com.fmi.weprom.api;


public abstract class Permission {

  ////////////////////////////////////////////////////////////////////////////////////////
  // Permissions for USER Role
  ////////////////////////////////////////////////////////////////////////////////////////

  // UserWorkspace releated
  public static final String VIEW_CALENDAR = "calendar.view";
  public static final String VIEW_REMINDER = "reminder.view";
  public static final String CREATE_REMINDER = "reminder.create";
  public static final String DELETE_REMINDER = "reminder.delete";
  public static final String UPDATE_REMINDER = "reminder.update";
  public static final String CREATE_FOLDER= "folder.create";
  public static final String DELETE_FOLDER= "folder.delete";
  public static final String VIEW_FOLDER = "folder.view";
  public static final String CREATE_DOC = "document.create";
  public static final String DELETE_DOC = "document.delete";
  public static final String VIEW_DOC = "document.view";
  public static final String CREATE_CONTACT = "contact.create";
  public static final String UPDATE_CONTACT = "contact.update";
  public static final String VIEW_CONTACT = "contact.view";
  public static final String VIEW_CONTACTS = "contact.list.view";
  public static final String CREATE_MESSAGE = "message.create";
  public static final String VIEW_MY_WORKSPACE = "user.workspace.view";

  // Project related
  // TODO check if it is better to move them to another role, for example
PROJECT_PARTICIPANT
  public static final String VIEW_PROJECT = "project.view";
  public static final String CREATE_PROJECT = "project.create";
  public static final String VIEW_PROJECT_FOLDER = "project.folder.view";
  public static final String CREATE_PROJECT_FOLDER= "project.folder.create";
  public static final String DELETE_PROJECT_FOLDER= "project.folder.delete";
  public static final String CREATE_PROJECT_DOC = "project.document.create";
  public static final String DELETE_PROJECT_DOC = "project.document.delete";
  public static final String VIEW_PROJECT_DOC = "project.document.view";
  public static final String UPDATE_PROJECT_TASK = "project.task.update";
  public static final String VIEW_PROJECT_TASK = "project.task.view";
  public static final String VIEW_PROJECT_TASKS = "project.tasks.view";
  public static final String VIEW_PROJECT_REMINDER = "project.reminder.view";
  public static final String VIEW_PROJECT_CALENDAR = "project.calendar.view";

  // Community related ones
  public static final String VIEW_COMMUNITY = "community.read";
```

```
  public static final String CREATE_COMMUNITY = "community.create";
  public static final String CREATE_COMMUNITY_FOLDER = "community.folder.create";
  public static final String DELETE_COMMUNITY_FOLDER = "community.folder.delete";
  public static final String VIEW_COMMUNITY_FOLDER = "community.folder.read";
  public static final String CREATE_COMMUNITY_DOC = "community.doc.create";
  public static final String DELETE_COMMUNITY_DOC = "community.doc.delete";
  public static final String VIEW_COMMUNITY_DOC = "community.doc.read";
  public static final String VIEW_COMMUNITY_CALENDAR = "communi-
ty.calendar.read";
  public static final String VIEW_COMMUNITY_REMINDER = "communi-
ty.reminder.read";

  ////////////////////////////////////////////////////////////////////////////////////
  // Permissions for PROJECT_COORDINATOR Role
  ////////////////////////////////////////////////////////////////////////////////////
  public static final String DELETE_PROJECT = "project.delete";
  public static final String UPDATE_PROJECT = "project.update";
  public static final String PROJECT_USER_EDIT_ROLES = "project.user.roles.edit";
  public static final String CREATE_PROJECT_REMINDER = "project.reminder.create";
  public static final String DELETE_PROJECT_REMINDER = "project.reminder.delete";
  public static final String UPDATE_PROJECT_REMINDER = "project.reminder.update";
  public static final String CREATE_PROJECT_TASK = "project.task.create";
  public static final String DELETE_PROJECT_TASK = "project.task.delete";
  public static final String ASSIGN_TASKS = "project.task.assign";

  ////////////////////////////////////////////////////////////////////////////////////
  // Permissions for COMMUNITY_COORDINATOR Role
  ////////////////////////////////////////////////////////////////////////////////////
  public static final String UPDATE_COMMUNITY = "community.update";
  public static final String DELETE_COMMUNITY = "community.delete";
  public static final String COMMUNITY_USER_EDIT_ROLES = "communi-
ty.user.roles.edit";
  public static final String CREATE_COMMUNITY_REMINDER = "communi-
ty.reminder.create";
  public static final String UPDATE_COMMUNITY_REMINDER = "communi-
ty.reminder.update";
  public static final String DELETE_COMMUNITY_REMINDER = "communi-
ty.reminder.delete";

  protected String name;
  protected int id;

  public Permission(int id, String name) {
    this.id = id;
    this.name = name;
  }

  public String getName() {
    return name;
  }
```

```
  public int getID() {
    return id;
  }

  public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Permission[");
    sb.append(id).append(',').append(name).append(']');
    return sb.toString();
  }

  public abstract boolean implies(Permission permission);

}
```

9.2.1.8. Project.java

```
package com.fmi.weprom.api;

import java.util.Date;
import java.util.List;

public interface Project extends Workspace {

  // project statuses
  public static final int PROJECT_STATUS_NOTSTARTED = 0;
  public static final int PROJECT_STATUS_INPROCESS = 1;
  public static final int PROJECT_STATUS_WAITING = 2;
  public static final int PROJECT_STATUS_COMPLETED = 3;

  public Calendar getCalendar();

  public Task createTask(User owner);

  public List getTasks();

  public Task getTask(int id);

  public void save() throws WEPROMException;

  public void update() throws WEPROMException;

  public void delete() throws WEPROMException;

  public String getName();

  public void setName(String name) ;

  public Date getStartDate();
```

```
  public Date getEndDate();

  public String getDescription();

  public int getCompleted();

  public int getStatus();

  public void setStartDate(Date startDate);

  public void setEndDate(Date endDate);

  public void setDescription(String  description);

  public void setCompleted(int completed);

  public void setStatus(int status);

  public void inviteUser(User user) throws WEPROMException;
}
```

9.2.1.9. Reminder.java

```
package com.fmi.weprom.api;

import java.util.Date;

public interface Reminder {

  public static final int STATUS_ON = 1;
  public static final int STATUS_OFF = 2;

  public int getID();

  public Date getDate();

  public void setDate(Date date);

  public String getSubject();

  public void setSubject(String subject);

  public String getDescription();

  public void setDescription(String description);

  public int getStatus();

  public void setStatus(int status);

  public void save() throws WEPROMException;
```

```
  public void delete() throws WEPROMException;

  public void update() throws WEPROMException;
}
```

9.2.1.10. Role.java

```
package com.fmi.weprom.api;

import java.util.List;


public abstract class Role {

  public static final String ADMINISTRATOR = "Administrator";
  public static final String PROJECT_COORDINATOR = "ProjectCoordinator";
  public static final String COMMUNITY_COORDINATOR = "CommunityCoordinator";
  public static final String USER = "User";
  public static final String GUEST = "Guest";
  public static final String PROJECT_PARTICIPANT = "ProjectParticipant";
  public static final String COMMUNITY_PARTICIPANT = "CommunityParticipant";

  protected String name;
  protected int id;


  public Role(int id, String name) {
    this.id = id;
    this.name = name;
  }

  public String getName() {
    return name;
  }

  public int getID() {
    return id;
  }

  public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("Role[");
    sb.append(id).append(',').append(name).append(']');
    return sb.toString();
  }

  public abstract List getPermissions();

  public int hashCode() {
    return name.hashCode();
```

```
  }

  public boolean equals(Object anObject) {
   if (this == anObject)
       return true;

   if (anObject instanceof Role) {
     Role r = (Role)anObject;
      return r.name.equals(name);
    }
    return false;
  }
}
```

9.2.1.11. Task.java

```
package com.fmi.weprom.api;

import java.util.Date;
import java.util.List;

public interface Task {
  // task statuses
  public static final int TASK_STATUS_NOTSTARTED = 0;
  public static final int TASK_STATUS_INPROCESS = 1;
  public static final int TASK_STATUS_WAITING = 2;
  public static final int TASK_STATUS_COMPLETED = 3;

  // task priorities
  public static final int TASK_PRIORITY_LOW = 0;
  public static final int TASK_PRIORITY_MEDIUM = 1;
  public static final int TASK_PRIORITY_HIGH = 2;
  public static final int TASK_PRIORITY_VERYHIGH = 3;

  public int getID();

  public void setProcessor(User processor);

  public User getProcessor();

  public void save() throws WEPROMException;

  public void delete() throws WEPROMException;

  public void update() throws WEPROMException;

  public int getCompleted();

  public void setCompleted(int completed);

  public String getDescription();
```

```
  public void setDescription(String description);

  public Date getEndDate();

  public void setEndDate(Date end_date);

  public int getPriority();

  public void setPriority(int priority);

  public Date getStartDate();

  public void setStartDate(Date start_date);

  public int getStatus();

  public void setStatus(int status);

  public String getSubject();

  public void setSubject(String subject);

  public User getOwner();

  public Project getProject();

  public void setAssigned(boolean assigned);

  public boolean isAssigned();

}
```

9.2.1.12. User.java

```
package com.fmi.weprom.api;

import java.io.Serializable;
import java.util.List;

public abstract class User implements Serializable {

  protected int id;
  protected String nick;
  protected String password;
  protected String name;
  protected String lastName;
  protected String email;
  protected String info;
  protected String phone;
```

```
public abstract void save() throws WEPROMException;

public abstract void update();

public abstract void delete();

public abstract UserWorkspace getUserWorkspace();

public abstract List getCommunities();

public abstract List getProjects();

//public abstract AccessRights getAccessRights(Workspace w);

public String getNick() {
  return nick;
}

public String getEmail() {
  return email;
}

public void setEmail(String email) {
  this.email = email;
}

public String getLastName() {
  return lastName;
}

public void setLastName(String lastName) {
  this.lastName = lastName;
}

public String getName() {
  return name;
}

public void setName(String name) {
  this.name = name;
}

public String getPassword() {
  return password;
}

public void setPassword(String password) {
  this.password = password;
}
```

```
  public int getID() {
    return id;
  }

  public boolean checkPassword(String password) {
    return this.password.equals(password);
  }

  public String toString() {
    StringBuilder sb = new StringBuilder();
    sb.append("User[").append(id);
    sb.append(',').append(nick).append(']');
    return sb.toString();
  }

  public String getInformation() {
    return info;
  }

  public void setInformation(String information) {
    this.info = information;
  }

  public String getPhone() {
    return phone;
  }

  public void setPhone(String phone) {
    this.phone = phone;
  }
  public int hashCode() {
    return nick.hashCode();
  }

  public boolean equals(Object anObject) {
   if (this == anObject)
       return true;

    if (anObject instanceof User) {
      User u = (User)anObject;
      return u.nick.equals(nick);
    }
    return false;
  }
}
```

9.2.1.13. UserWorkspace.java

```
package com.fmi.weprom.api;

import java.util.List;
```

```
public interface UserWorkspace extends Workspace {

  public List getInboxMessages() throws WEPROMException;

  public List getOutboxMessages() throws WEPROMException;

  public int getUnreadMessageCount() throws WEPROMException;

  public Message createMessage();

  public Message getMessage(int id);

  public List getContacts();

  public Contact createContact();

  public Contact getContact(int id);

  public List getTasks();

  public Calendar getCalendar();

}
```

9.2.1.14. WEPROMException.java

```
package com.fmi.weprom.api;

public class WEPROMException extends Exception {

  public WEPROMException(String message) {
    super(message);
  }

  public WEPROMException(String message, Throwable cause) {
    super(message, cause);
}
}
```

9.2.1.15. WEPROMFactory.java

```
package com.fmi.weprom.api;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.fmi.weprom.impl.mysql.UserImpl;

public class WEPROMFactory {

  private static final Log LOG = LogFactory.getLog(UserImpl.class);
```

```
  public static WEPROMSystem getWEPROMSystem() {
    // load the default implementation
    try {
      Class clazz = Class.forName("com.fmi.weprom.impl.mysql.WEPROMSystemImpl");
      return (WEPROMSystem) clazz.newInstance();
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    } catch (InstantiationException e) {
      e.printStackTrace();
    } catch (IllegalAccessException e) {
      e.printStackTrace();
    }
    return null;
  }
}
```

9.2.1.16. WEPROMSystem.java

```
package com.fmi.weprom.api;

import java.util.List;

public interface WEPROMSystem {

  public User getUser(String nick);

  public User getUser(int id);

  public List getUsers();

  public User createUser(String nick);

  public Community createCommunity(User owner);

  public Project createProject(User user);

  public List getProjects();

  public Project getProject(int id);

  public Community getCommunity(int id);

  public List getCommunities();

  public AccessRights getAccessRights();

  public File getFile(int id);
  }
```

9.2.1.17. Workspace.java

```
package com.fmi.weprom.api;

import java.util.List;

public interface Workspace {

  public int getID();

  public User getOwner();

  public List getUsers();

  public File getRootFolder();

}
```

9.2.2. Пакет com.fmi.weprom.util

9.2.2.1. WepromRequestCycle.java

```
package com.fmi.weprom.util;

public interface WepromRequestCycle {

  public void open();

  public void close();

}
```

9.2.2.2. WepromRequestCycleFactory.java

```
package com.fmi.weprom.util;

public interface WepromRequestCycleFactory {

  public WepromRequestCycle createWepromRequestCycle();

}
```

9.2.2.3. WepromRequestCycleUtil.java

```
package com.fmi.weprom.util;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

import com.fmi.weprom.api.User;
import com.fmi.weprom.api.WEPROMSystem;
public class WepromRequestCycleUtil {

  private static final Log LOG = LogFactory.getLog(WepromRequestCycleUtil.class);
```

```
  private static String SESSION_FACTORY =
"com.fmi.weprom.impl.mysql.SessionFactory";
  private static WepromRequestCycleFactory wrcFactory = null;


  ////////////////////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////////////

  private static final ThreadLocal threadLocalUser = new ThreadLocal();

  public static void setUser(User user) {
    LOG.debug("Caller set to : "+user);
    if (threadLocalUser.get()!=null) {
      throw new IllegalStateException("Caller already set!");
    }
    threadLocalUser.set(user);
  }

  public static User currentUser() {
    User user = (User) threadLocalUser.get();
    //LOG.debug("ThreadLocal current user: " + user);
    return user;
  }

  private static void clearUser() {
    threadLocalUser.set(null);
  }

  ////////////////////////////////////////////////////////////////////////////////////
  ////////////////////////////////////////////////////////////////////////////////////

  private static final ThreadLocal threadLocalWepromRequestCycle = new ThreadLocal();

  public static WepromRequestCycle currentWepromRequestCycle() {
    //LOG.debug("currentSession()");
    WepromRequestCycle session = (WepromRequestCycle) threadLocalWepromRequest-
Cycle.get();

    // Open a new Session, if this Thread has none yet
    if (session == null) {
      session = createWepromRequestCycle(); //this should be thread-safe method
      threadLocalWepromRequestCycle.set(session);
    }
    return session;
  }

  private static void closeWepromRequestCycle() {
    //LOG.debug("closeSession()");
    WepromRequestCycle wrc = (WepromRequestCycle) threadLocalWepromRequest-
Cycle.get();
```

```
  threadLocalWepromRequestCycle.set(null);
  if (wrc != null) {
    wrc.close();
  }
 }

 ////////////////////////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////////////////


 private static final ThreadLocal threadLocalSharedSessionData = new ThreadLocal();

 public static void setSharedSessionData(Object ssData) {
  LOG.debug("SharedSessionData set to : "+ssData);
  threadLocalSharedSessionData.set(ssData);
 }

 public static Object currentSharedSessionData() {
  Object ssData = threadLocalSharedSessionData.get();
  //LOG.debug("Current SharedSessionData: " + ssData);
  return ssData;
 }

 private static void clearSharedSessionData() {
  threadLocalSharedSessionData.set(null);
 }

 ////////////////////////////////////////////////////////////////////////////////////////
 ////////////////////////////////////////////////////////////////////////////////////////

 private static WepromRequestCycleFactory getWepromRequestCycleFactory() {
  if (wrcFactory==null) {
    try {
      LOG.info("getWepromRequestCycleFactory() "+"instantiate factory: " +
SESSION_FACTORY);
      Class factoryClass = Class.forName(SESSION_FACTORY);
      wrcFactory = (WepromRequestCycleFactory) factoryClass.newInstance();
    } catch (Exception e) {
      LOG.error("getWepromRequestCycleFactory() "+"FAILED to instantiate factory: " +
SESSION_FACTORY, e);
    }
  }
  return wrcFactory;
 }

 private static WepromRequestCycle createWepromRequestCycle() {
  LOG.debug("createWepromRequestCycle()");
  WepromRequestCycleFactory mlsf = getWepromRequestCycleFactory();
  if (mlsf==null) {
    throw new Error("no WepromRequestCycleFactory available");
```

```
    }
    return mlsf.createWepromRequestCycle();
    //return new MysqlSession(); // later may need to create a SessionFactory ...
  }

  public static void close() {
    LOG.debug("close()");
    closeWepromRequestCycle();
    clear();
  }

  public static void open() {
    LOG.debug("open()");
    // clear to prevent dirty obejcts usage
    clear();

    // currentWepromRequestCycle().open();
  }

  public static void beginRender() {
    LOG.debug("beginRender()");
    WepromRequestCycle wrc = (WepromRequestCycle) threadLocalWepromRequest-
Cycle.get();
    if (wrc!=null) {
      wrc.beginRender();
    }
  }

  private static void clear() {
    clearUser();
    clearSharedSessionData();
  }

}
```

9.2.2.4. WepromRequestFilter.java

```
package com.fmi.weprom.util;

import java.io.IOException;
import java.text.DateFormat;
import java.util.Date;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.tapestry.engine.state.ApplicationStateManager;
import org.apache.tapestry.services.WebRequestServicer;
import org.apache.tapestry.services.WebRequestServicerFilter;
import org.apache.tapestry.web.WebRequest;
import org.apache.tapestry.web.WebResponse;
import org.apache.tapestry.web.WebSession;
```

```
import com.fmi.weprom.impl.mysql.RequestFilter;

import workspace.Visit;

public class WepromRequestFilter implements WebRequestServicerFilter {

  private static final Log LOG = LogFactory.getLog(WepromRequestFilter.class);
  static final String VISIT_KEY = "visit";
  private ApplicationStateManager appStateManager;


  public void service(WebRequest request, WebResponse response, WebRequestServicer ser-
vicer) throws IOException {
    String serviceName = request.getParameterValue("service");
    String path = "";
    if ("asset".equals(serviceName)) {
      path = " path: "+request.getParameterValue("path");
    }
    LOG.debug("BEGIN service: " + serviceName+path);

    WepromRequestCycleUtil.open();

    // Tapestry Request Cycle Begin
    servicer.service(request, response);
    // Tapestry Request Cycle End

    storeSharedSessionData();
    WepromRequestCycleUtil.close();

    LOG.debug("END");
  }

  private void storeSharedSessionData() {
    // Visit do not need to be aware of SharedSessionData, it could be stored using Applica-
tionStateManager.store(...)
    if (appStateManager.exists(VISIT_KEY)) {
      Visit visit = (Visit) appStateManager.get(VISIT_KEY);
      LOG.debug("SharedSessionData --> Visit");
      Object ssData = WepromRequestCycleUtil.currentSharedSessionData();
      visit.setSharedSessionData(ssData);
    }
  }

  public void setAppStateManager(ApplicationStateManager asm) {
    appStateManager = asm;
  }
}
```

9.2.3. Други по-интересни класове

9.2.3.1. WEPROMBasePage.java

```
package workspace.web.pages;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.apache.tapestry.PageRedirectException;
import org.apache.tapestry.annotations.InjectStateFlag;
import org.apache.tapestry.event.PageBeginRenderListener;
import org.apache.tapestry.event.PageDetachListener;
import org.apache.tapestry.event.PageEvent;
import org.apache.tapestry.event.PageRenderListener;
import org.apache.tapestry.event.PageValidateListener;
import org.apache.tapestry.html.BasePage;

import workspace.Visit;

import com.fmi.weprom.api.User;
import com.fmi.weprom.util.WepromRequestCycleUtil;

public abstract class WEPROMBasePage extends BasePage implements
   PageValidateListener, PageBeginRenderListener {//, PageDetachListener {

 @InjectStateFlag("visit")
 public abstract boolean getVisitExists();

 private static final Log LOG = LogFactory.getLog(WEPROMBasePage.class);

 public void pageValidate(PageEvent event) {
  User user = WepromRequestCycleUtil.currentUser();
  if (user == null) {
   Visit visit = (Visit) getPage().getVisit();
   if (visit.isUserLoggedIn()) {
    user = visit.getUser();

    // attach the current user to this thread
    WepromRequestCycleUtil.setUser(user);
    // attach the current shared session data to this thread
    WepromRequestCycleUtil.setSharedSessionData(visit.getSharedSessionData());
    return;
   } else {
    // no logged in user, so redirect him/her to the login page :)
    LoginPage login = (LoginPage) getRequestCycle().getPage("LoginPage");
    // login.setCallback(new PageCallback(this)); // should be uncomented if
    // need to remember the current page
    throw new PageRedirectException(login);
   }
  }
 }
```

```
  // we should not have sql transaction during the rendering!
  public void pageBeginRender(PageEvent event) {
    //LOG.debug("pageBeginRender()");
    if (!event.getRequestCycle().isRewinding()) {
      WepromRequestCycleUtil.beginRender();
    }
  }
}
```

9.2.3.2. PageValidateListener.java

```
package org.apache.tapestry.event;

import java.util.EventListener;

/**
 *  An interface for objects that want to take part in the validation of the page.
 *
 *  @author Mindbridge
 *  @since 3.0
 **/

public interface PageValidateListener extends EventListener
{
    /**
     *  Invoked by the page from its
     *  {@link org.apache.tapestry.IPage#validate(org.apache.tapestry.IRequestCycle)}
method.
     *
     *  <p>May throw a {@link org.apache.tapestry.PageRedirectException}, to redirect the
user
     *  to an appropriate part of the system (such as, a login page).
     **/

    public void pageValidate(PageEvent event);
}
```