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

на тема:

Управление на неполучени електронни писма

Дипломант: Иван Иванов Васев, фак. № M21649

Специалност: Информатика

Специализация: Разпределени системи и мобилни технологии

Научен ръководител: доцент д-р Боян Бончев

София, 2007 г.

Съдържание

Съдържание 2
1. Увод 5
1.1. Въведение 5
1.2. Цел и задачи на дипломната работа 6
1.3. Структура на изложението 7
2. Анализ на проблемите, свързани с отхвърлените и неполучените електронни писма 9
2.1. Причини за неполучаване на електронни писма 9
2.2. Съобщения за грешки връщани от различните имейл сървъри 13
2.3. Проблеми свързани с отхвърлените и неполучени писма при липса на отговор от страна на имейл сървъра на адресата 16
3. Решения на проблема свързан с отхвърлените и неполучените електронни писма 17
3.1. Съществуващи приложения 17
3.1.1. eMail Bounce Handler 17
3.1.2. Bounce eMail Manager 18
3.1.3. Email Processor 19
3.1.4. b*Bounce Server 2005 19
3.1.5. BoogieTools 19
3.1.6. switchemail 20
3.2. Сравнение на съществуващите приложения и решението на проблема предложен с настоящата дипломна работа 21
4. Реализация на приложението 24
4.2. Дизайн на приложението 24
4.2.1. Инфраструктура 24
4.2.2. Клиент-сървър комуникация и изисквания към клиента на EBM 25
4.2.3. EBM API 27
4.2.4. EBM API за клиенти 30
4.2.5. Съответствие между изпратен и получен имейл 32
4.2.6. Предприемане на действия при откриване на bounce проблем 33

4.2.7. Откриване на причина за bounce имейл 34
4.3. Използвани средства 35
4.4. Реализация 35
4.4.1. Регистриране на EBM клиент 36
4.4.2. Отрегистриране на EBM клиент 38
4.4.3. Изпращане на имейл 40
4.4.4. Получаване на имейл 42
4.4.5. Структура на базата данни 46
5. Тестване на приложението 55
5.1. Необходими условия за тестване на приложението 55
5.2. Стартиране на приложението и неговият клиент 56
5.3. Регистриране на клиент 57
5.4. Изпращане на имейли 59
5.5. Настройки за действия за изпълнение при откриване на soft bounce 60
5.5.1. Добавяне на настройки 61
5.5.2. Разглеждане на настройки 63
5.5.3. Изтриване на настройки 64
5.6. Разглеждане на списъците с изпратени и получени имейли 65
5.6.1. Разглеждане на списъка с изпратените имейли 65
5.6.2. Разглеждане на списъка с получените имейли 66
5.7. Получаване на имейл 67
5.7.1. Обработка при валиден отговор 67
5.7.2. Обработка при неразпознат получен имейл 67
5.7.3. Обработка при soft bounce 67
5.7.4. Обработка при hard bounce 68
5.8. Тестване на пиложението при валиден получен имейл 68
5.9. Тестване на приложението при soft bounce 68
5.10. Тестване на приложението при изгубен имейл 69
5.11. Тестване на приложението при неразпознат имейл 69
6. Заключение и насоки за бъдещо развитие на приложението 70
7. Списък на използваните съкращения и термини 72
8. Използвана литература 75

9. Приложение с SQL заявки необходими за работата на приложението 76
9.1. SQL заявки необходими за създаването на таблиците в базата данни на EBM 76
9.2. SQL заявки необходими за въвеждане на регулярни изрази свързани с различните кодове за грешка 78
9.3. SQL заявки необходими за успешната работа на EBM с неговият клиент 80
10. Списък на разгледаните фигури 81

1. Увод

1.1. Въведение

Електронната поща с времето придобива все по-голяма популярност и намира нови приложения, като замества конвенционалните пощенски услуги и ползването на твърд носител в процеса на комуникация. От друга страна спецификата на използваните имейл сървъри (e-mail servers), налага и някои ограничения. Един от проблемите, които възникват, е невъзможността в някои ситуации да се следи статусът на даден изпратен имейл. Фактори, които водят до тази неопределеност са начин на настройка на отделните имейл сървъри, настройки на защитни стени (firewalls) и др. В резултат на това, след изпращането на даден имейл, изпращачът не може да бъде сигурен, че въпросният имейл е пристигнал успешно до получателя.

Електронните писма (имейл), които не могат да бъдат получени, се разделят на две групи – hard bounce и soft bounce. *Hard bounce* са електронни писма, които не могат да бъдат получени изобщо, т.е. *отхвърлени (от сървъра) писма*. Това може да се получи, когато домейнът не съществува, има мрежов проблем от страна на получателя, адресът е невалиден или имейл сървърът на получателя е блокирал сървъра на изпращача. *Soft bounce* са електронни писма, които са достигнали до имейл сървъра на получателя (адресът е валиден), но не достигат до самия него, т.е. *неполучени (от адресата) електронни писма*. Това може да се получи, ако имейл сървърът е претоварен, кутията на получателя е препълнена, или имейлът е твърде голям. В повечето случаи, при възникване на hard или soft bounce, изпращачът получава известие за възникналия проблем. Възможна е и ситуация, в която такова известие да не се получи (например имейл сървърът на получателя може да приеме въпросното писмо като спам и да го филтрира, без да изпрати съобщение уведомяващо за това изпращача). Такъв имейл можем да наречем изгубен. Този проблем е особено голям в полуавтоматизирани системи, където изпращачът не е човек, а софтуерна програма и е необходимо да се знае статуса на всеки имейл по всяко време. Нека разгледаме процес, в който си кореспондират човек

и програма. Работата на програмата е да изпрати на човека електронно писмо, да изчака определено време за отговор и след като го получи, да го обработи по предефиниран начин. Възможно е, обаче, програмата да изпрати писмо, което да бъде филтрирано като СПАМ (SPAM) и да не пристигне до получателя си. В такъв случай процесът се прекъсва и не може да завърши.

1.2. Цел и задачи на дипломната работа

Целта на дипломната работа е да анализира проблемите свързани с hard и soft bounce електронни писма. Като резултат от това изследване, ще бъде предложен дизайн и реализация на софтуерно приложение, което да служи за решение на hard и soft bounce проблема.

Съществува голяма гама от софтуерни продукти, опитващи се да се справят с проблема, като прочитат всички получени от даден изпращач и-мейл писма, отделят bounce имейлите и извършват предефинирани операции с тях. По този начин, обаче, остава нерешен проблемът с изгубените имейли. За да се справи с този проблем, предложеното тук решение, ще следи не само получените писма, но и изпратените. По този начин ще се следи статусът на всеки изпратен и получен имейл. То ще предлага API (Application Program Interface), чрез което клиентите да могат да четат и пращат имейли и да се регистрират/отпишат от работа с него. Приложението ще разполага и с потребителски интерфейс, чрез който от една страна да се задават настройките му свързани с начина на обработка на различните bounce имейли, а от друга, да се следи за статуса на всяко едно изпратено или получено писмо.

Задачите, произтичащи от тази цел са:

- Анализиране на soft и hard bounce проблема
- Проучване на съществуващи решения за този проблем
- Предлагане на разширено решение на проблема, включващо и обработка на изгубените имейли
- Разработка, дизайн и реализация на софтуерно приложение въз основа на предложеното решение

- Тестване на разработеното приложение
- Анализиране на резултатите и предимствата на разработеното решение

1.3. Структура на изложението

Настоящата дипломна работа се състои от десет глави: "Увод", "Анализ на проблемите свързани с отхвърлените и неполучените електронни писма", "Решения на проблема свързан с отхвърлените и неполучените електронни писма", "Реализация на приложението", "Тестване на приложението", "Заключение и насоки за бъдещо развитие на приложението", "Списък на използваните съкращения и термини", "Използвана литература", "Приложение с SQL заявки необходими за работата на приложението" и "Списък на разгледаните фигури".

- **Глава 1. "Увод"** съдържа кратко въведение и описание на поставените цели и задачи в дипломната работа. В тази глава е поместено и настоящата структура на изложението.
- **Глава 2. "Анализ на проблемите свързани с отхвърлените и неполучените електронни писма"** разглежда различните причини водещи до неполучаване на електронни писма. В нея се изброяват и някои основни съобщения за грешки връщани от различни имейл сървъри. В тази глава е обърнато внимание и на проблемите свързани с отхвърлените и неполучени писма, при липса на отговор от страна на имейл сървъра на адресата.
- **Глава 3. "Решения на проблема свързан с отхвърлените и неполучените електронни писма"** съдържа кратко описание на основни съществуващи на пазара приложения свързани с обработката на bounce имейли и ги съпоставя с решението предложено с настоящата дипломна работа.
- **Глава 4. "Реализация на приложението"** съдържа дизайна на предложеното приложение (EBM), разглежда използваните средства и основни моменти от реализацията на приложението.

- **Глава 5. "Тестване на приложението"** разглежда начина по който е тестван ЕВМ, както и необходимите условия за успешно тестване.
- **Глава 6. "Заключение и насоки за бъдещо развитие на приложението"** обобщава плюсовете и минусите на предложеното настоящата дипломна работа софтуерно решение и предлага насоки за бъдещото му развитие.
- **Глава 7. "Списък на използваните съкращения и термини"** разглежда използваните в дипломната работа термини и съкращения и тяхното значение.
- **Глава 8. "Използвана литература"** съдържа списък с използваната за написването на настоящата дипломна работа литература.
- **Глава 9. "Приложение с SQL заявки необходими за работата на приложението"** съдържа списък от SQL заявки, които трябва да се изпълнят при инсталиране на приложението.
- **Глава 10. "Списък на разгледаните фигури"** предлага списък с всички използвани в настоящата дипломна работа фигури.

2. Анализ на проблемите, свързани с отхвърлените и неполучените електронни писма

2.1. Причини за неполучаване на електронни писма

Когато даден потребител се опита да изпрати електронно писмо, e-mail системата, която той използва е необходимо да намери домейна[1] на получателя (например yahoo.com) и имейл сървъра отговарящ за този домейн. След като това стане, имейл сървърът на получателя сканира писмото, за да реши дали да го предаде на получателя му или не. Ако сървърът на получателя е настроен да не приема електронни писма от адреса на изпращача (например ако този адрес е блокиран като СПАМ[2]), то той ще отхвърли въпросното писмо. В някои случаи, писмото се връща при изпращача (в зависимост от настройките на сървъра на получателя), като в повечето случаи е придружено със съобщение за грешка (описващо проблема довел до неполучаването на писмото). Писмото може да се върне при изпращача и ако имейл сървърът на получателя е пренатоварен и не може да го обработи в този момент, или ако не съществува такъв адрес при този сървър.

Има ситуации, в които имейл сървърът на получателя първоначално приема писмото, а по-късно го отхвърля. Някои сървъри са програмирани да приемат всички електронни писма и да ги записват за по-късен анализ, без първоначално да проверят дали адресата съществува и дали е възможно те да бъдат получени [1].

Причините за неполучаване на електронни писма могат да се разделят на няколко основни вида:

- Причини свързани с имейл сървъра на получателя:

[1] domain
[2] SPAM

- **Неразпознат потребител**[3] - получава се, когато имейлът на получателя вече не съществува, бил е временно изключен от употреба, или домейн частта на адреса е невалидна. Това се случва най-често, когато някой напусне компанията, в която е работил, или когато някой смени своя интернет доставчик[4], или когато имейла на получателя е написан грешно.
- **Препълнена пощенска кутия**[5] - също така познат като "Recipient Over-quota", "<recipient>'s mailbox is full" или "Mailbox disk quota exceeded". В този случай, въпреки че имейл адресът на получателя е валиден, имейлът не пристига при адресата си. Това се получава, защото повечето имейл сървъри ограничават пощенските кутии на своите потребители, да събират определено количество имейли и когато се запълни това количество, повече писма, не могат да бъдат получени. Този проблем се отстранява лесно, като потребителят направи място в пощенската си кутия (най-често като изтрие стари и ненужни имейли от сървъра), а изпращачът изпрати отново съобщението си. В някои случаи, обаче – особено когато става дума за безплатни имейл сървъри, това съобщение на практика може да озачава, че потребителят вече не си проверява пощата.
- **Съобщението надхвърля позволения размер**[6] **-** в този случай размера на писмото (включително всички хедъри[7], текст и прикачени файлове), е по-голям от разрешения за даднения домейн максимален размер за един имейл, т.е. имейлът е твърде голям, за да бъде получен.
- **СПАМ** - Често системните администратори настройват системите за които отговарят, да отхвърлят нежелани имейли – СПАМ. Но тъй като не съществуват безгрешни системи за филтриране на СПАМ, е възможно някои имейли да бъдат

[3] User Unknown
[4] ISP – Internet Service Provider
[5] Mailbox Full
[6] Message exceeds size limit
[7] Headers

филтрирани по погрешка, т.е. ако домейна на изпращача е включен в черен списък[8] съдържащ домейни от които се получава СПАМ. Този черен списък може да бъде получен или от външен сервиз, който служи за разпространение на такива списъци. Възможно е и администратора изрично да е забранил работата с домейна на изпращача поради твърде много по обем и/или брой имейли пристигащи от въпросния домейн.

- **Отказана връзка[9] –** още познато като „Connection Timed Out". Обикновено се получава, когато огромно количество имейли се обработват от имейл сървъра в момента на пристигане на проблемният имейл. Това може да се дължи на по-голямо от предвиденото натоварване на сървъра, външна атака на домейна, или вътрешен проблем с настройките на сървъра, който води до отказ на конекции от на имейл сървърите на домейна, или прекъсване на конекциите, преди някое съобщение да бъде напълно изпратено. Имейл сървърите са настроени така, че да приемат за обработка толкова съобщения, с колкото могат да се справят, така че, когато проблемът с натоварването се отстрани, изпращането на имейли отново ще се осъществява безпроблемно.
- **Домашен IP адрес[10]** – означава, че имейл сървърът на получателя не приема съобщения изпратени от домашни IP адреси. Под домашен IP адрес (независимо дали е динамичен или фиксиран), се разбира IP адрес на интернет потребител, ползващ DSL[11], LAN[12], или модем връзка. Ако такъв потребител е настроил собствен имейл сървър и се опитва да изпрати имейл от него, то някои имейл сървъри ще отхвърлят имейла. Примерен такъв сървър е AOL[13].

[8] blacklist
[9] Connection Refused
[10] Residential Internet Protocol address
[11] Digital Subscriber Line
[12] Local Area Network
[13] America Online

- Причини свързани с транспортирането на имейла:
  - **Домейнът не може да бъде открит**[14] - в този случай домейнът на който се изпраща съобщението не съществува. Това най-често става, при грешка при написването на името на домейна. Друга причина водеща до тази грешка, може да е проблем свързан със откриването на домейна.
  - **Прекалено дълъг път на рутиране**[15] - възможно е имейла да е попаднал в цикъл по време на рутиране[16]. Това може да е в резултат или на референция[17] на локален потребител сочеща към рекурсивна референция на отдалечена машина, или две системи взаимно се използват в MX записите си. Има и други причини водещи до този проблем, но тези са най-често срещаните.
  - **Достъпът за препредаване е отказан**[18] – получава се, когато скоро е сменена хостинг компанията за дадения домейн и новият запис за домейна все още не се е разпространил напълно. При тази ситуация, имейлът може да пристигне при старата хостинг компания, която вече не приема имейли за преместения домейн.
- Други причини:
  - **Неизвестни причини**[19] – въпреки, че много от съобщенията за неполучени имейли, съдържат точни и ясно описани причините за непристигането на имейла, за съжаление съществуват и много нестандартни и уникални начини по които някои имейл сървъри оказват доставянето на някои имейли. Когато даден имейл не бъде доставен до адресата по неизвестни причини, това не ни носи информация дали той е

[14] Domain Not Found
[15] Too many hops
[16] Routing loop
[17] alias
[18] Relay Access Denied
[19] Unknown

получен, или не. Понякога това се дължи на временен Интернет проблем (например спрян сървър, прекъсната връзка и т.н.).

2.2. Съобщения за грешки връщани от различните имейл сървъри

Целта на съобщенията за грешки, които се връщат от имейл сървъра на получателя в случай на недоставен имейл, е да опише причината за неполучаването на имейла. Тези съобщения могат да бъдат в най-различен формат, който зависи както от вида на въпросният имейл сървър, така и от въведените от администратора настройки. Нека разгледаме няколко често срещани в практиката основни вида съобщения с кратко разяснение за всеки от тях (повечето от тях са изброени в RFC[20] 2821 [2]):

- **Message rejected**[21] – В този случай съобщението не е получено, тъй като в хедърите на имейла е записано, че той идва от имейл сървър, който не съответства на IP адреса, от който на практика е изпратен. Този тип промяна на хедърите често се използва от автори на СПАМ, поради което повечето имейл сървъри отхвърлят такива имейли. Например ако даден имейл е изпратен с адрес на подателя user@domain.com, но е изпратен чрез имейл сървър с IP адрес, който не отговаря на domain.com, то този имейл най-вероятно ще бъде отхвърлен [3].
- **421 <domain> Service not available, closing transmission channel**[22] – В този случай сървърът на получателя временно или перманентно няма възможност да приема имейли и прекъсва връзката за комуникация.

[20] Request For Comment
[21] Съобщението е отхвърлено
[22] Сервиза не е на разположение, комуникационният канал ще бъде прекъснат

- **450 Requested mail action not taken: mailbox unavailable**[23] – Това съобщение се изпраща от сървъра на адресата, за да уведоми изпращача, че съобщението не е доставено, тъй като имайлът на получателя не е открит, т.е. не е валиден.
- **451 Requested action aborted: local error in processing**[24] **-** Това съобщение се изпраща, в случай че настъпи грешка докато имейл сървърът обработва дадена SMTP[25] заявка.
- **452 Requested action not taken: insufficient system storage**[26] – Това съобщение се изпраща, в случай, че е свършило свободното дисково пространство или оперативна памет на сървъра и той не може да обработи повече SMTP заявки.
- **500 Syntax error, command unrecognized**[27] – Това съобщение означава, че имейл сървърът на адресата на писмото не е успял да разпознае командата изпратена от имейл сървъра на изпращача.
- **501 Syntax error in parameters or arguments**[28] – Това съобщение означава, че имейл сървърът се е опитал да изпълни SMTP команда, но е открил синтактична грешка в нейните параметри.
- **502 Command not implemented**[29] – Това съобщение означава, че командата, която е извикана за изпълнение на имейл сървъра на адресата или е изключена от употреба, или не е имплементирана.
- **503 Bad sequence of commands**[30] – Това съобщение означава, че отдалеченият имейл сървър се опитва да зададе поредица от SMTP транзакции в неправилна последователност.

[23] Заявеното имейл операция не е осъществено: пощенската кутия не е открита
[24] Заявеното действие е прекъснато преди да бъде изпълнено: възникнала е локална грешка по време на обработката
[25] Simple Mail Transfer Protocol
[26] Заявеното действие не е осъществено: недостатъчно системно пространство за съхранение
[27] Синтактична грешка, командата не може да бъде разпозната
[28] Синтактична грешка свързана с параметрите или аргументите
[29] Командата не е имплементирана
[30] Неправилна последователност от команди

- **504 Command parameter not implemented**[31] – Това съобщение означава, че командата, която е извикана за изпълнение е имплементирана, но тя изисква параметър, който от своя страна не е имплементиран.
- **550 Requested action not taken: mailbox unavailable**[32] – Това съобщение означава, че пощенската кутия не може да бъде открита, че няма достъп до нея или че командата извикана за е забранена за изпълнение.
- **551 User not local**[33] – Това съобщение означава, че адресатът на имейла не е локален потребител на имейл сървъра и опциите за препращане не позволяват той да бъде прехвърлен до крайната си дестинация.
- **552 Requested mail action aborted: exceeded storage allocation**[34] – Това съобщение означава, че пощенската кутия на адресата е препълнена и нови имейли не могат да бъдат записани там.
- **553 Requested action not taken: mailbox name not allowed**[35] - Това съобщение означава, че имейл адресът на адресата не е синтактично коректен или е невалиден.
- **554 Transaction failed**[36] – Това съобщение означава, че транзакцията е била неуспешна.

Този списък не е напълно изчерпателен и не покрива напълно списъка с причините за неполучаване на имейли, но дава реална представа за най-често възникващите съобщения за грешки.

[31] Параметър на командата не е имплементиран
[32] Заявената операция не е осъществена: пощенската кутия не е открита
[33] Потребителят не е локален
[34] Заявената имейл операция е прекъсната преди да бъде изпълнена: надхвърлен е максималният размер на пространството за съхранение на имейли
[35] Заявената операция не е изпълнена: използвано е непозволено име на пощенска кутия
[36] Неуспешна транзакция

2.3. Проблеми свързани с отхвърлените и неполучени писма при липса на отговор от страна на имейл сървъра на адресата

В някои случаи е възможно имейл сървърът на получателя на имейла, да е настроен по такъв начин, че да не се върне bounce съобщение на изпращача, въпреки, че адресатът не е получил имейла. В този случай от гледна точка на изпращача, електронното писмо е пристигнало успешно до получателя, въпреки че това не е вярно. Тази ситуация може да води до неприятни последствия както за изпращача, така и за получателя, в зависимост от важността на въпросния имейл.

В случай, че изпращачът на имейла е някаква полуавтоматизирана система, то тя няма как да разбере, дали адресатът го е получил, или не. С настоящата дипломна работа се предлага и решение на този проблем.

3. Решения на проблема свързан с отхвърлените и неполучените електронни писма

Съществуват много приложения опитващи се да се справят с неполучените писма. В настоящата глава ще бъдат разгледани някои от софтуерните решения предлагани на пазара, ще бъдат съпоставени със решението предложено чрез настоящата дипломна работа и ще бъдат посочени предимствата му спрямо тях.

3.1. Съществуващи приложения

При повечето от съществуващите приложения, използваната терминология свързана с описанието на bounce проблема се различава, но по начина си на работа, те са сходни. Основното съревнование при тях е базирано на количеството съобщения за грешки, които могат да разпознават в случай на bounce, количеството на пощенски кутии, от които четат, протоколите с които работят и цената на която се предлагат.

3.1.1. eMail Bounce Handler

Разработва се от компанията Maxprog. Според интернет страницата на продукта [4], основната функционалност на тази програма е филтриране на получени имейли с цел разпознаване на bounce имейлите. След като открие тези имейли, програмата записва проблемните имейли и оставя потребителят да реши дали да препраща отново имейла, който не е пристигнал, или да се откаже изобщо да праща имейли на този адрес. Програмата разглежда всички получени имейли на изпращача и събира информация от тези от тях, които успее да разпознае като bounce. За връзка и четене на имейлите, които

обработва, програмата поддържа POP3 и IMAP протоколи за връзка с пощенските кутии.

Главното приложение на тази програма е прочистването на списъци от имейли[37] от адреси до които не могат да пристигат по една или друга причина имейли. Птенциални потребители на тази програма са автори на СПАМ съобщения, като чрез нея те могат да си "прочистват" периодически имейл листите от имейли, на които не е възможно да се пращат имейли. От интернет страницата на този продукт, твърдят, че разпознават 99,9% от bounce имейлите, което изключва по-екзотичните съобщения за грешки връщани от някои имейл сървъри. Цената на един лиценз за този продукт е на стойност от $25.00, като се разпространяват версии за Windows (включително Vista) и Macintosh компютри.

3.1.2. Bounce eMail Manager

Разработва се от Mailsbroadcast dotcom. По функционалност не се различава значително от вече разгледания по-горе eMail Bounce Handler. Както е посочено на интернет страницата на продукта [5], той се настройва да чете от определен брой имейл акаунти чрез POP3, като сканира за bounce имейли и на база на откритите такива, прави статистика. Основна разлика, е че се твърди, че може да обработва едновременно много имейл акаунти и съответно много имейли. Работи под Windows и цената на един от пакетите достига до $499. Съществува и безплатна версия, но с ограничени възможности и четене от максимум 15 POP3 имейл сървъра. Продукта предлага също така и сканиране за вируси на прикачените към отхвърлените имейли файлове. Резултатите могада да бъдат записвани в текстови, CSV[38], Access файлове, или да се експортират към MSSQL сървър.

[37] E-mail list
[38] Comma Separated Values

3.1.3. Email Processor

Разработва се от G-Lock Software. Това е още едно приложение с подобни на дотук разгледаните продукти функции. На интернет страницата му [6], е обяснено, че то е разработено като 32 битова програма за работа под Windows. Начинът на работа е отново да чете чрез POP3 имейлите от дадена пощенска кутия и да събира информация за bounce имейлите. Приложението струва $99 и отново основното му приложение е да поддържа имейл листите на изпращачите на СПАМ "чисти" от невалидни имейли.

3.1.4. b*Bounce Server 2005

Собсвеник на разработката е b*Bounce Company. На интернет страницата на продукта [7] твърдят, че притежават един от най-големите списъци с проблеми свързани с bounce имейли, които разпознават успешно. Важна е и възможността да се получава обновление на този списък ежедневно и автоматично, без намеса на потребителя. Предлагат и възможност потребителят сам да дефинира свой елемент към този списък. Приложението работи под Windows версии NT, 2000, XP и 2003. На интернет страницата се отбелязва също така и голямата скорост с която работи продуктът и възможността да се настрои да използва по-малко процесорно време, ако компютърът на който е инсталирана е нужен за използване активно и с други приложения. Цената варира от $99 за работа с пет пощенски кутии до $699 за работа с неограничен брой пощенски кутии. Основният принцип на работа на програмата е аналогичен на разгледаните по-горе.

3.1.5. BoogieTools

Продуктът се разработва от BoogieTools Inc. Според интернет страницата на приложението [8], то работи както под Windows, така и под Linux. Като

принцип на действие наподобява останалите приложения, но предлага и допълнителна функционалност. Основното предимство пред разгледаните дотук приложения е предлаганото BoogieBounce API. Чрез него, потребителя може да разпознава различните bounce формати. Подобно на b*Bounce Server 2005, се предлага и обновяване на списъка свързан с разпознатите проблеми свързани с bounce имейлите. Друго предимство на този продукт е поддръжката на различни езици на този списък. Предложеното API се предлага заедно със примерен изходен код[39] написан на C/C++, Visual Basic, VB-Script, PowerBASIC, ColdFusion, Delphi, VB.NET и C#. Предлага се и опционален ActiveX интерфейс. По-голямата предлагана функционалност, води и до по-голямата цена на продукта. Различните версии ценово варират от $99 за поддръжка на пет POP3 имейл пощенски кутии, до $999 за работа с неограничен брой пощенски кутии.

3.1.6. switchemail

Switchemail представлява интернет страница [9], която предлага услуга различна от до момента разгледаните приложения. Нека потребител „А" изпрати имейл на потребител „Б". Ако този имейл не пристигне до адресата, при което „А" получи bounce съобщение, то потребителят „А" може да препрати bounce съобщението на имейл адреса на switchemail, като те от своя страна ще се опитат да открият валиден имейл на „Б" и да го изпратят на „А". При това положение „А" може да изпрати на новиq адрес на „Б" отново своя имейл и проблемът би трябвало да бъде решен по този начин. За целта ако потребителят „Б" смени имейл адреса си, трябва предварително да е информирал switchemail интернет страницата за това и да е въвел стария и новия си адрес. Това решение, обаче помага само в случай, че „Б" е въвел тези данни. Във всички останали ситуации switchemail не може да помогне. Недостатъкът на тази процедура, е че трябва достатъчно много потребители да се регистрират на този интернет адрес и да публикуват там всяка промяна на

[39] Source code

интернет адресите си. Но тъй като самата страница не е набрала голяма популярност, ефектът от нейното използване не е голям.

3.2. Сравнение на съществуващите приложения и решението на проблема предложен с настоящата дипломна работа

Предложеното чрез тази дипломна работа решение на bounce проблема се нарича Email Bounced Manager[40] – за по-кратко в по-нататъшното изложение, ще бъде използвана абревиатурата EBM.

Основната разлика между EBM и вече съществуващите приложения, се състои в това, че EBM освен да чете и сканира имейлите от дадена пощенска кутия за bounce проблеми, следи и изпратените имейли. Т.е. служи не само за получаване, но и за изпращане на писма. По този начин се появява възможността EBM да се опитва да направи връзка между изпратен имейл и получено bounce съобщение свързано с този имейл. От една страна, това предлага детерминираност на всички имейли и възможност да се следи техния статус. От друга страна това позволява, да се предложи решение на проблема свързан с неполучените и отхвърлени имейли, за които не е върнато bounce съобщение.

EBM е разработен като приложение, което се ползва от клиенти. Всеки клиент може да се се регистрира/отрегистрира, да изпраща и получава имейли чрез EBM. Всеки клиент е необходимо да предлага имплементация на предоставени от EBM интерфейси, за да може да се регистрира успешно. Нека разгледаме ситуация, в която изпращачът на имейл е клиент на EBM, а получателят е човек. Клиентът на EBM очаква на всеки изпратен имейл, да се получи отговор в рамките на предварително дефинирано "време за изчакване". Ако отговор не се получи преди изтичането на това време, изпратеният имейл се счита за изгубен и отговор за него не се очаква от този момент нататък. Това дава детерминираност на имейлите, но въвежда и ограничение в случаите в

[40] Програма за обработка на bounce имейли

които отговор на имейла или bounce съобщение за въпросното писмо се получи след изтичане на предварително дефинираното време за изчакване. Въз основа на това, след изпращане на даден имейл, е възможно:

- Имейлът да пристигне успешно до адресата. В този случай има три варианта:
  - Отговор да се изпрати преди да изтиче времето за изчакване. В този случай комуникацията между изпращач и адресат е осъществена успешно.
  - Отговор да се изпрати след като изтиче времето за изчакване. В този случай въпреки полученият отговор, от гледна точка на ЕВМ и неговия клиент, имейлът се приема за изгубен след изтичане на времето за изчакване.
  - Да не се изпрати никакъв отговор. В този случай, имейлът се счита за изгубен.
- Имейлът да не пристигне успешно до адресата. В този случай съществуват два варианта:
  - Да се върне bounce съобщение до изпращача преди да изтиче времето за изчакване. В такъв случай, ако имаме hard bounce, имейлът не може да бъде доставен. В случай на soft bounce, въз основа на настройките на клиента на ЕВМ, могат да се изпълнят предварително дефинирани действия (най-често свързани с повтороно изпращане на първоначално изпратения и отхвърлен имейл).
  - Да се върне bounce съобщение след като изтече времето за изчакване – в този случай независимо от вида на съобщението, имейлът се счита отново за изгубен.

Въз основа на горното описание на възможните ситуации, които могат да възникнат и съответната обработката на, която ЕВМ извършва, може да се раздели на два вида:

- Обработка на съобщения за грешка от имейл сървъра на получателя, т.е. обработка в случай на bounce.

- Обработка на изгубени имейли – такива, за които не е получен отговор, или bounce съобщение преди изтичане на времето за изчакване.

Именно вторият вид обработка липсва при разгледаните съществуващи приложения представящи решение за bounce проблема (Таблица 1 Съпоставка на ЕВМ с вече съществуващи приложения).

	eMail Bounce Handler	Bounce eMail Manager	Email Processor	b*Bounce Server 2005	BoogieTools	switchemail	Email Bounce Manager
Настройки за действия, които да се изпълнят при откриване на bounce	+	+	-	-	-	-	+
Работа под Windows	+	+	+	+	+	+	+
Работа под Linux	-	-	-	-	+	+	+
Работа под Macintosh	+	-	-	-	-	+	+
Разпознаване на голямо количество bounce съобщения	+	+	-	+	+	+	-
Обработка на изгубени имейли (имейли, за които не е получен отговор, или bounce съобщение преди изтичане на времето за изчакване	-	-	-	-	-	-	+

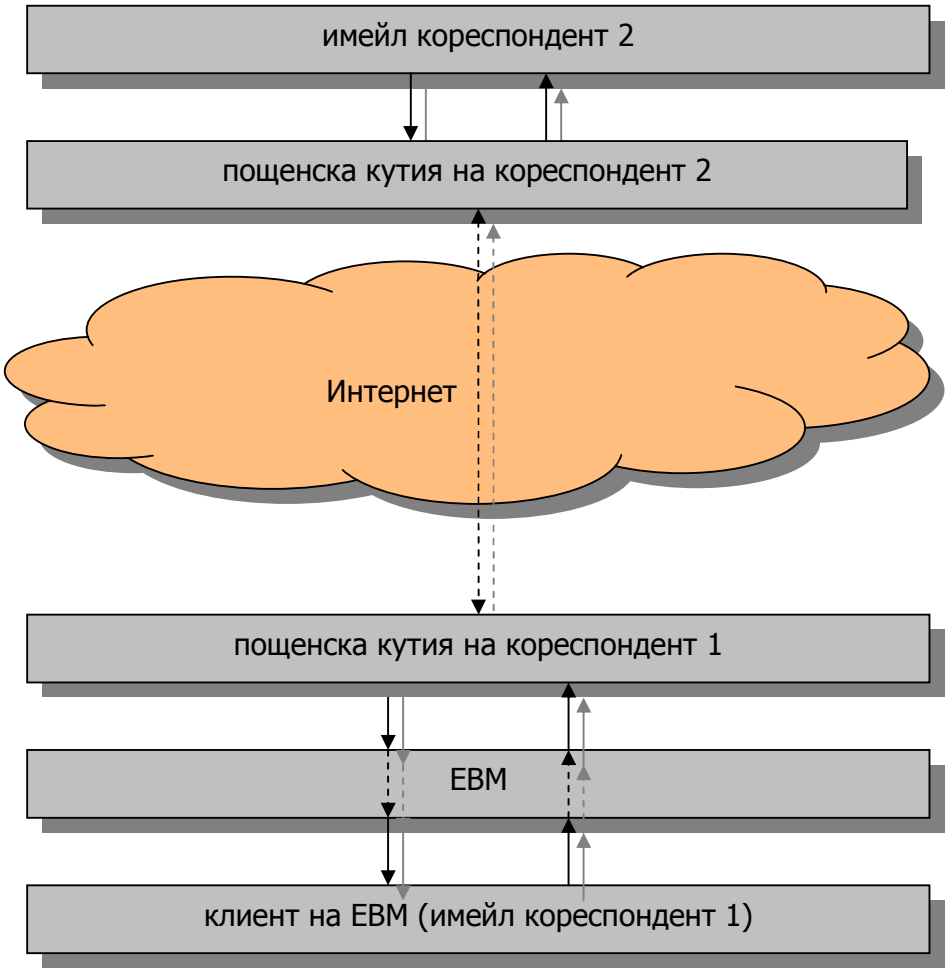
Таблица 1 Съпоставка на ЕВМ с вече съществуващи приложения

4. Реализация на приложението

4.2. Дизайн на приложението

4.2.1. Инфраструктура

В имейл кореспонденцията между клиента на ЕВМ и външен потребител, ЕВМ се явява като буфер на изпратените и получените съобщения (Фиг. 1 Инфраструктура на ЕВМ и комуникацията между имейл кореспондентите).



Фиг. 1 Инфраструктура на ЕВМ и комуникацията между имейл кореспондентите

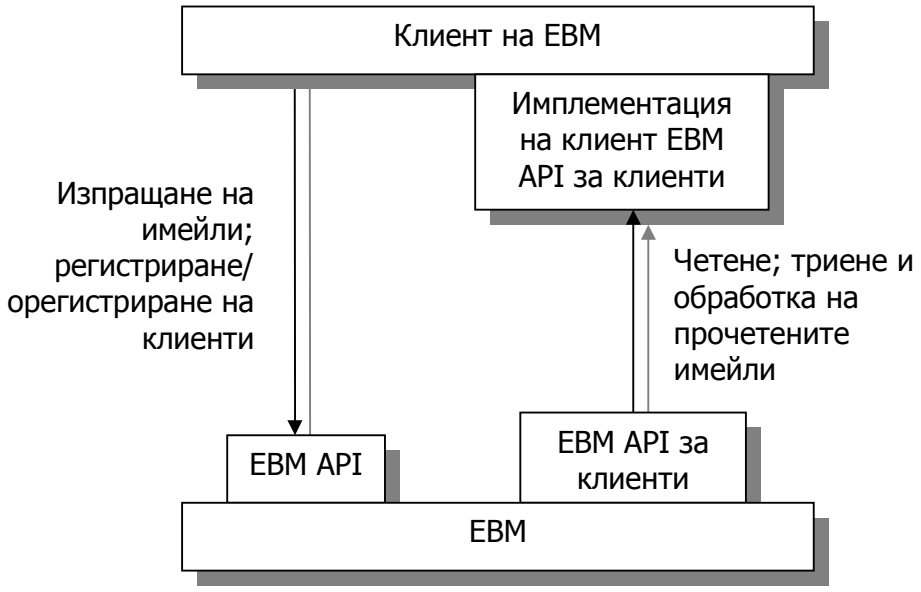
Нека наречем клиента на EBM имейл кореспондент 1, а потребителя, с който той комуникира чрез имейли, имейл кореспондент 2. Когато кореспондент 1 изпраща имейл на кореспондент 2, това става чрез EBM. В този случай, EBM записва в базата си данни информация за изпратеното писмо. Ако кореспондент 2 изпрати отговор на въпросното писмо, то минава през EBM, който от своя страна се опитва да намери съответстващия на това писмо изпратен имейл, след което то се предава на клиента, т.е. на кореспондент 1.

Ако вместо отговор, се върне bounce съобщение от имейл сървъра на кореспондент 2, то преминава отново през EBM, който отново се опитва да открие съответстащия на това писмо изпратен имейл. В този случай, въз основа на настройките на EBM, ако е открит съответният изпратен имейл, той може да бъде изпратен отново (това действие има смисъл да се извършва в случай на soft bounce).

4.2.2. Клиент-сървър комуникация и изисквания към клиента на EBM

EBM предлага два вида API за своите клиенти. Единият вид API служи на клиентите за изпращане на имейли, регистриране и отрегистриране от EBM – нека го наречем EBM API. Другият вид API трябва да бъде имплементиран от клиента – нека го наречем EBM API за клиенти. Той предоставя функционалността свързана както с четене и изтриване на имейлите от пощенската кутия на клиента, така и с обработка на прочетените имейли от страна на клиента (Фиг. 2 Комуникация между EBM и негов клиент).

Задължително условие да се регистрира дадена програма като клиент на EBM, е необходимо тя да предостави на EBM валидна имплементация на EBM API за клиенти. Ако такава имплементация е предоставена, чрез нея и EBM API, клиента може успешно да се регистрира за работа с EBM.

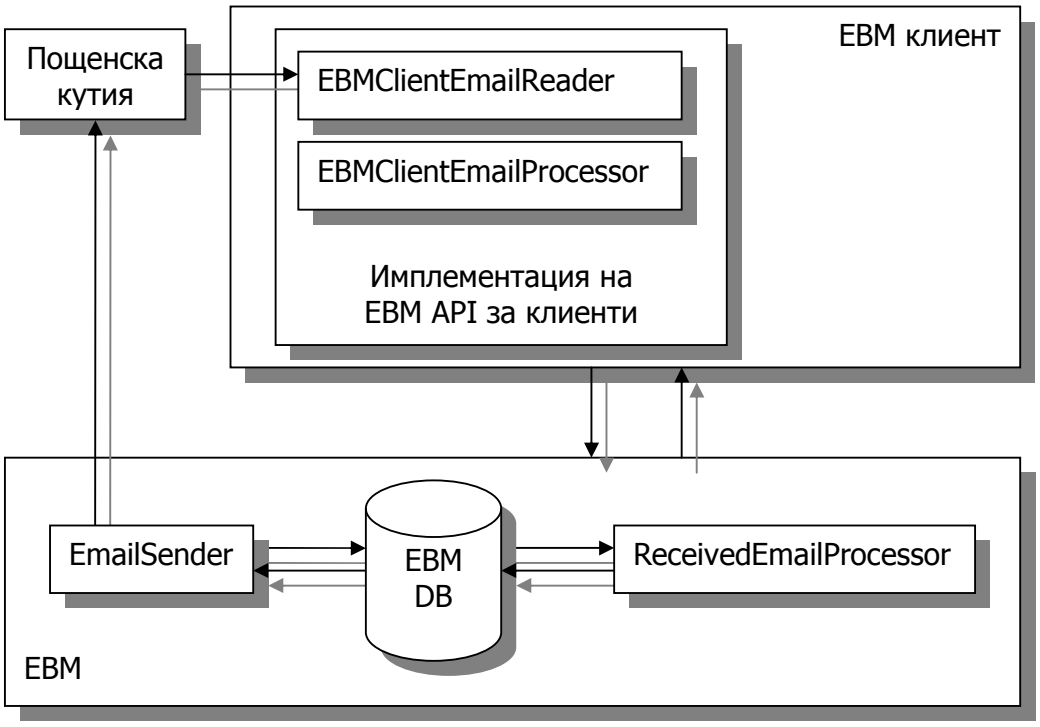


Фиг. 2 Комуникация между ЕВМ и негов клиент

Ако регистрацията протече успешно, клиента може да използва ЕВМ API за изпращане на имейли или отрегистриране от ЕВМ. От своя страна, след протичане на успешна регистрация, ЕВМ започва на предварително дефиниран интервал от време да се опитва да чете имейли чрез ЕВМ API за клиенти от пощенската кутия на новорегистрирания клиент. Когато ЕВМ успее да прочете имейл, той го подава за обработка от клиента, след което го обработва от своя страна и го изтрива чрез ЕВМ API за клиенти.

При изпращане на имейл чрез ЕВМ, се използва EmailSender – клас, който осъществява връзка с имейл сървъра на клиента и изпраща подаденият имейл. Преди изпращане на даден имейл, за него се генерира уникален идентификационен номер, който се записва в него. След изпращането на имейла, информация за него се записва в базата данни на ЕВМ (Фиг. 3 Детайли от структурата на ЕВМ и негов клиент).

При получаване на имейл, той се прочита от пощенската кутия на клиента чрез EBMClientEmailReader – клас от имплементацията на ЕВМ API за клиенти. След това прочетеният имейл се обработва последователно от EBMClientEmailProcessor – клас от имплементацията на ЕВМ API за клиенти и от ReceivedEmailProcessor – клас от ЕВМ за обработка на получени имейли.

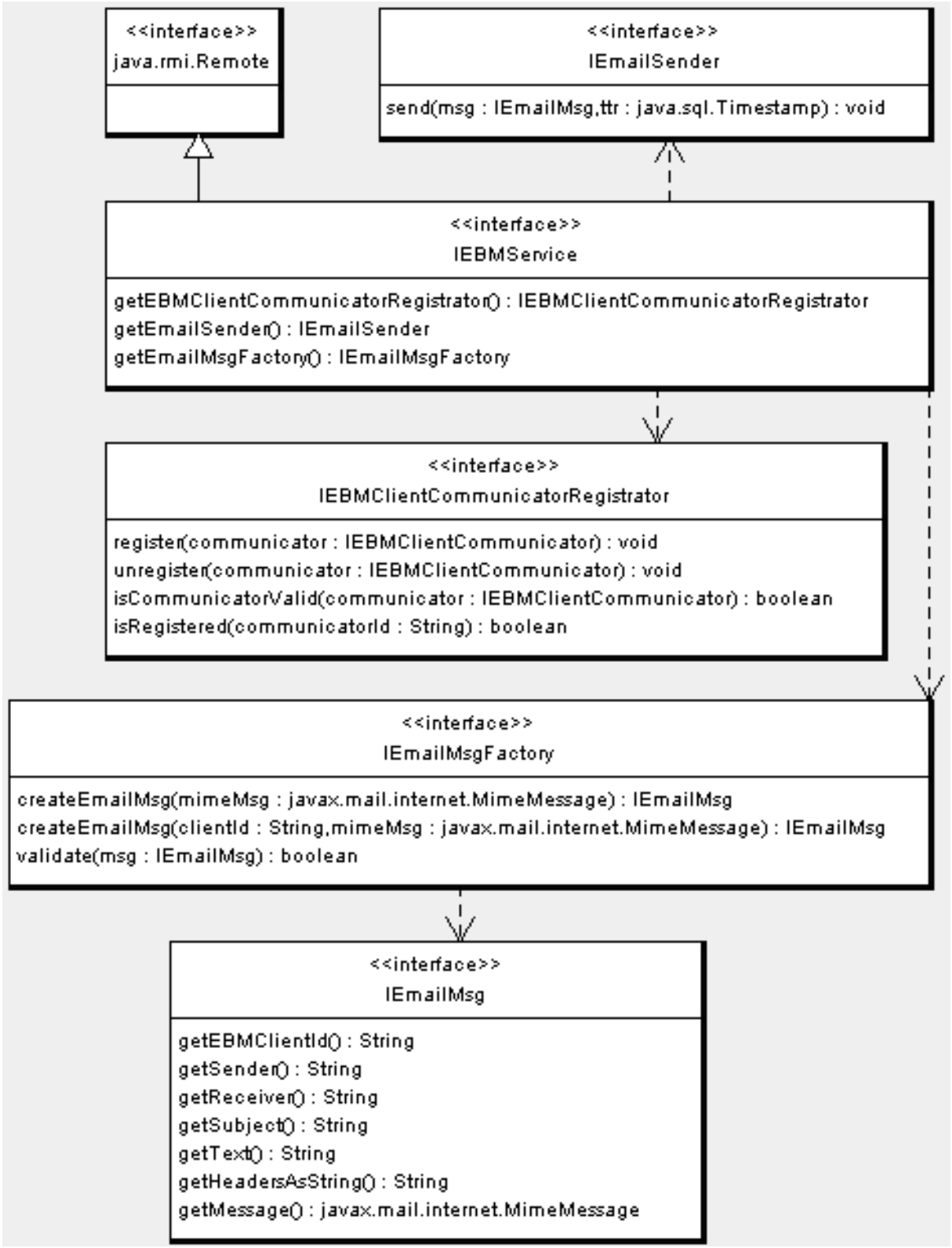


Фиг. 3 Детайли от структурата на EBM и негов клиент

4.2.3. EBM API

Както вече беше споменато, за извършване на клиент-сървър комуникацията, е необходимо да се имплементират EBM API и EBM API за клиенти съответно от EBM и от клиентите на EBM.

EBM API е необходимо да предлага функционалност свързана с изпращане на имейли, регистриране и отрегистриране на клиенти (Фиг. 4 EBM API).



Фиг. 4 EBM API

Нека разгледаме основните интерфейси и класове участващи в тях:

IEBMService – Интерфейсът на EBM сервиза, който предлага достъп до функционалността предоставена от него. Достъпен е чрез RMI.

- public IEBMClientCommunicatorRegistrator **getEBMClientCommunicatorRegistrator**() – Връща регистратора на клиенти.
- public IEmailSender **getEmailSender**() – Връща имплементацията служеща за изпращане на имейли.
- public IEmailMsgFactory **IEmailMsgFactory** – Връща имплементацията служеща за валидиране и създаване на обекти от тип IEmailMsg.

IEBMClientCommunicatorRegistrator – Служи за регистриране и отрегистриране на имплементации на IEBMClientCommunicator.

- public void **register**(IEBMClientCommunicator communicator) – Регистрира имплементация на IEBMClientCommunicator.
- public void **unregister**(IEBMClientCommunicator communicator) – Отрегистрира имплементация на IEBMClientCommunicator.
- public boolean **isCommunicatorValid**(IEBMClientCommunicator communicator) – Проверява, дали дадената имплементация на **IEBMClientCommunicator** е валидна.
- public boolean **isRegistered**(String communicatorId) – Проверява дали вече е регистриран клиент с подаденото communicatorId.

IEmailMsgFactory – Служи за валидиране и създаване на обекти от тип IEmailMsg

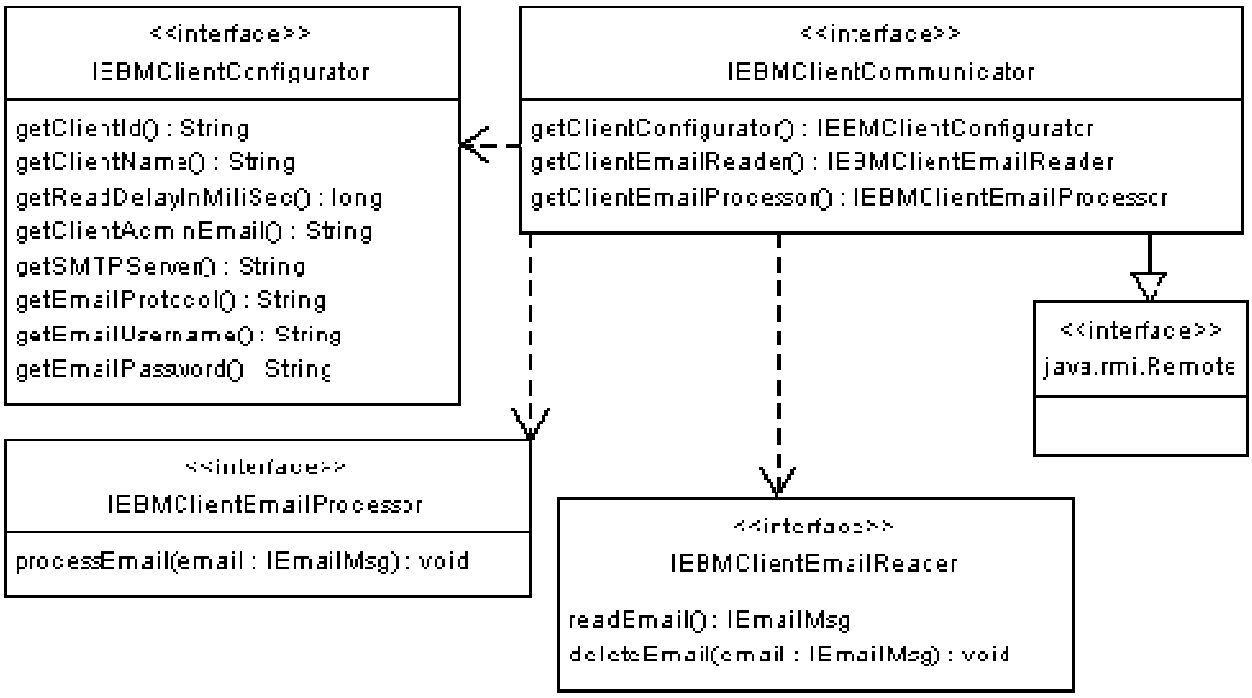
- public IEmailMsg createEmailMsg(javax.mail.internet.MimeMessage mimeMsg) – създава IEmailMsg от зададен MimeMessage
- public IEmailMsg createEmailMsg(String clientId, javax.mail.internet.MimeMessage mimeMsg) – създава IEmailMsg от зададен MimeMessage
- public boolean validate(IEmailMsg msg) – проверява валидността на подаден IEmailMsg

IEmailMsg – Репрезентация на имейл съобщение, използвана от EBM.

- public String **getEBMClientId**() – Връща уникалния идентификационен номер на клиента.
- public String **getSender**() – Връща имейл адреса на изпращача на имейла.
- public String **getReceiver**() – Връща имейл адреса на получателя на имейла.
- public String **getSubject**() – Връща стойността на полето относно (subject) на имейла.
- public String **getText**() – Връща текста на имейл съобщението.
- public String **getHeadersAaString**() – Връща хедърите (headers) на имейла представени като символен низ (String).
- public javax.mail.internet.MimeMessage **getMessage**() – Връща съответстащия имейл.

4.2.4. EBM API за клиенти

EBM API за клиенти е необходимо да се имплементира от всеки клиент на EBM, преди да може да се регистрира за работа с EBM. Той предлага функционалността свързана с четене, триене и обработка на имейли. По този начин не е задължително четенето на имейли да става от пощенска кутия. Тъй като имплементацията на тези интерфейси е част от клиента, те могад да бъдат изпълнени така, че четенето да става например от текстови файл или дори от база данни (Фиг. 5 EBM API за клиенти).



Фиг. 5 EBM API за клиенти

Нека разгледаме основните интерфейси и класове участващи в тях:

IEBMClientConfigurator – Чрез него EBM получава настройките на клиента. Достъпен е чрез RMI.

- public String **getClientId**() – Връща уникалния идентификационен номер (може да съдържа букви и цифри) на клиента.
- public String **getClientName**() – Връща името на клиента.
- public long **getReadDelayInMiliSec**() – Връща времето (в милисекунди), за което ще се спира изпълнението на нишката (the thread), отговаряща за обработката на имейлите на съответния клиент.

IEBMClientEmailProcessor – Използва се за обработка от страна на клиента на всеки получен имейл.

- public void **processEmail**(IEmailMsg email) – Обработва даден имейл.

IEBMClientEmailReader – Използва се за четене на имейлите.

- public IEmailMsg **readEmail**() – Чете и връща прочетения имейл.

- public void **deleteEmail**(IEmailMsg email) – Изтрива подадения като параметър имейл.

IEBMClientCommunicator – Клиентите на EBM трябва да предоставят имплементация на този интерфейс и да регистрират инстанция на тази имплементация чрез **IEBMClientCommunicatorRegistrator**. За всеки клиент трябва да се регистрира точно една инстанция на този клас.

- public IEBMClientConfigurator **getClientConfigurator**() – Връща настройките на клиента.
- public IEBMClientEmailReader **getClientEmailReader**() – Връща имплементацията служеща за четене на имейли.
- public IEBMClientEmailProcessor **getClientEmailProcessor**() – Връща имплементацията служеща за обработка на имейли.

4.2.5. Съответствие между изпратен и получен имейл

Основна част от работата на EBM, както вече беше споменато, е откриването на съответствие между изпратен и получен имейл. За целта е необходимо EBM да предостави функционалност свързана с добавянето на специфична информация към хедърите на имейлите, които изпраща, преди да ги изпрати. Тази информация трябва да съдържа уникален идентификатор[41] на имейла ограден в специфични предварително дефинирани символи. В такъв случай, ако се получи bounce съобщение, то би следвало този уникален идентификатор да се съдържа в неговия хедър и/или съдържание и/или прикачен(и) файлове (в случай, че оригиналният изпратен имейл е прикачен към bounce съобщението). По този начин, чрез този уникален идентификатор, е възможно да се определи на кой изпратен имейл съответства полученият bounce имейл.

В случай че не е позволено използването на уникален идентификатор в хедърите и съдържанието на имейлите, би могло да се използва евристично

[41] Unique ID

търсене на съответствие между изпратен имейл и bounce съобщение. Този подход би могло да се базира на сравнение на изпращач, получател, полето относно, полетата за дата и други полета. В този случай резултата не е гарантирано коректен, тъй като тези хедъри не могат уникално да идентифицират даден изпратен имейл.

Съществуват и ситуации, в които намирането на съответствие е невъзможно – такъв е например случаят със съобщение изпратено автоматично, което уведомява, че адресатът временно не може да отговаря на кореспонденцията си[42].

4.2.6. Предприемане на действия при откриване на bounce проблем

В случай на hard bounce, имейлът не само че не е пристигнал до адресата но в общия случай и не съществуват действия, предприемането на които да води до успешното му изпращане. В такъв случай, при hard bounce, единствентото, което може да се предприеме е да се преустанови по-нататъшно изпращане на имейли до въпросният адресат. Възможно е и да се следят броя на soft bounce проблемите, които са възникнали спрямо даден адресат и след определен брой, този имейл адрес би следвало да се „маркира" като hard bounce.

В случай на soft bounce, изпратеният имейл, би могло да се изпрати отново (определено брой пъти) до адресата. Съответно пощенската кутия на адресата би следвало да се „маркира" като soft bounce.

Много често, ако администраторите на имейл сървърите забележат, че имейлите от някой изпращач твърде често се връщат като bounce от един или повече адресата, които имат акаути при тях, блокират този изпращач да няма достъп до тяхната мрежа. В такъв случай, изпращачът няма да може да изпрати имейли на никой потребител с пощенска кутия от въпросния имейл сървър,

[42] Out-of-office auto reply

включително на тези, които са с валидни адреси и до момента не е имал проблеми с комуникацията с тях.

EBM ще се спрявя с soft bounce проблемите чрез повторно изпращане на проблемния имейл. В по-нови версии на програмата би могло да се добавят и други действия, които да се предприемата в случай на bounce.

4.2.7. Откриване на причина за bounce имейл

За откриване на причината за възникване на bounce проблем, EBM използва регулярни изрази[43], които се прилагат върху съдържанието на даден получен имейл. Това се извършва след като се прочете даден имейл чрез предоставената от клиента имплементация на **IEBMClientEmailReader**. Съществуват няколко варианта на действие в зависимост от упешното извършване на откриването на причина за bounce имейл и намирането на съответствие между изпратен и получен имейл. Нека разгледаме възможните случаи:

1. Успешно откриване на съответствие между изпратен и получен имейл и успешно откриване на причина за bounce имейл. В този случай, може изпратеният имейл да се изпрати наново, ако е на лице soft bounce.
2. Неуспешно откриване на съответствие между изпратен и получен имейл и успешно откриване на причина за bounce имейл. В този случай, въпреки, че е открита причината за възникналия bounce на изпратеният имейл, не можем да направим нищо, освен да уведомим администратора на EBM, при положение, че не знаем полученият bounce имейл на кой изпратен имейл съответства.
3. Успешно откриване на съответствие между изпратен и получен имейл и неуспешно откриване на причина за bounce имейл. В този случай, можем да приемем, че полученият имейл е валиден

[43] Regular expressions

отговор на изпратеното електронно писмо. От страна на ЕВМ не е нужно да се предприемат действия.

4. Неспешно откриване на съответствие между изпратен и получен имейл и неуспешно откриване на причина за bounce имейл. В този случай, няма действия, които ЕВМ може да предприеме, освен евентуално да уведоми администратора. Такъв имейл може да се приеме за несвързан с кореспонденциите извършвани чрез ЕВМ.

4.3. Използвани средства

За разработката на ЕВМ е използван езика за програмиране Java. Той е избран заради лесната си преносимост на различни хардуерни платформи. Интерфейсите ЕВМ API и ЕВМ API за клиенти са също написани на Java. Съответно за осъществяване на комуникацията между ЕВМ и неговите клиенти, се използва Java RMI. Този метод за комуникация е избран, заради тясната си интеграция с Java.

База данни, която ЕВМ използва за работата си, е MySQL, която намира широко приложение при реализирането на подобен род продукти.

Потребителските интерфейси свързани с работата на ЕВМ и неговите клиенти са разработени с помощта на SWING.

4.4. Реализация

Въз основа на разгледаните изисквания и поставени задачи пред ЕВМ, ще бъдат разгледани реализациите на основни примерни функции[44]. Такива са например регситриране, отрегистриране на ЕВМ клиент, изпращане и получаване на имейл чрез ЕВМ. Щв бъде разгледана и структурата на базата данни използвана от ЕВМ.

[44] Use cases

4.4.1. Регистриране на EBM клиент

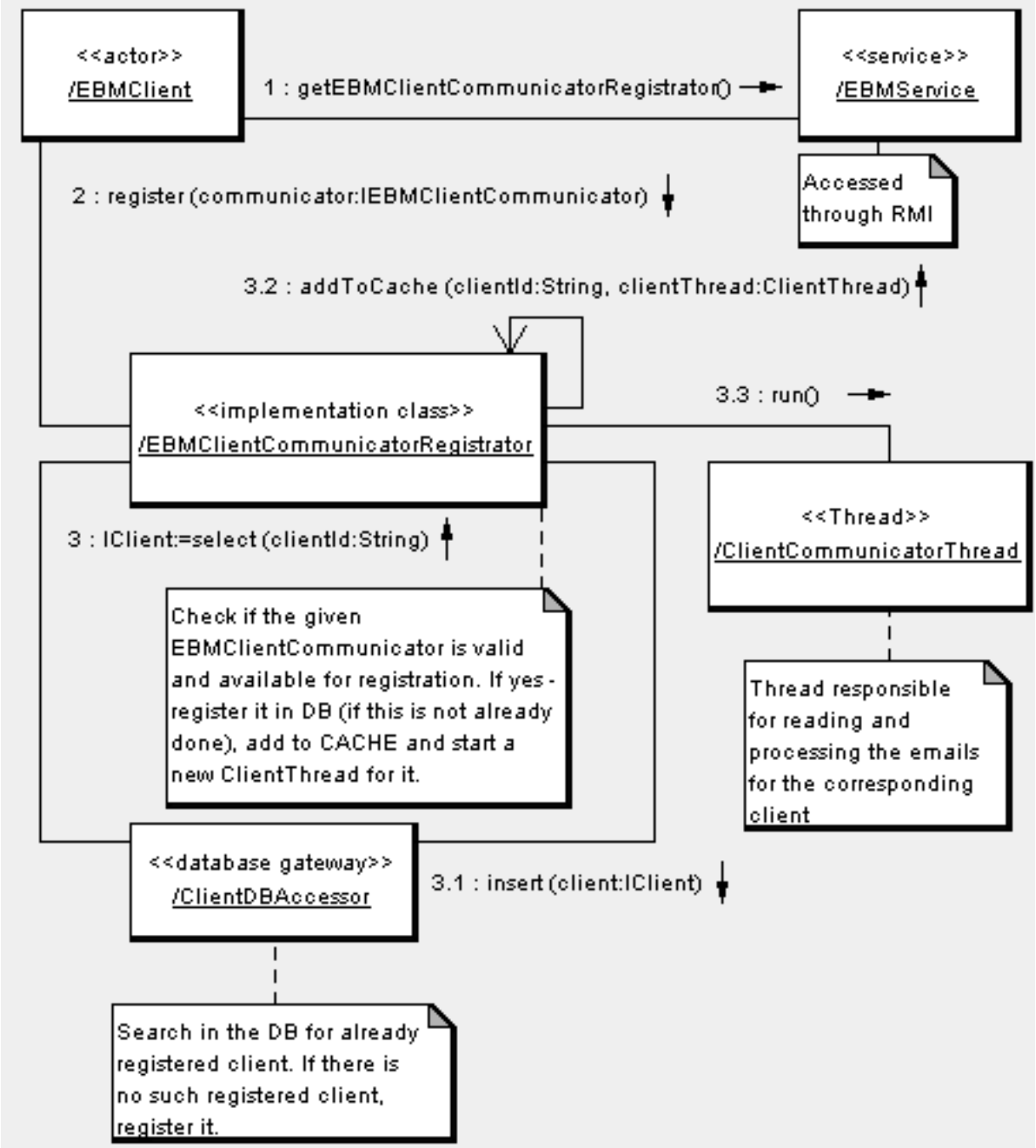
Преди да започне работа с EBM, всеки негов клиент, трябва да предостави имплементация на EBM API за клиенти и чрез нея да се регистрира за работа с EBM. Нека разгледаме няколко основни класа, които участват в процеса на регистрацията (Фиг. 6 Регистрация на EBM клиент):

- **EBMClient** – клас, репрезентиращ клиента на EBM
- **EBMService** – клас, представляващ сървъра на EBM, достъпен чрез RMI
- **EBMClientCommunicatorRegisrator** – клас, чрез който клиента се регистрира за работа с EBM. Той служи за валидиране на дадена имплементация на **IEBMClientCommunicator** и регистриране на валидирана такава имплементация в базата данни и кеш, който той поддържа, съдържащ всички регистрирани клиентски имплементации на този клас
- **ClientCommunicatorThread** – за всеки клиент на EBM, при успешна регистрация, се стартира отделна нишка репрезентирана с този клас. Той е отговорен за четене, обработка и изтриване на имейли съответстващи на дадения клиент през предварително определен от клиента интервал от време
- **ClientCommDBAccessor** – клас, който служи за достъп до таблицата от базата данни, в която се записва информация за клиентите на EBM

Процедурата по регистриране на EBM клиент, се инициира от клиента и протича в следния ред:

1. Клиентът достъпва сървъра на EBM - **EBMService** чрез RMI, след което извиква метода на EBM сървъра **getEBMClientCommunicatorRegistrator**. Чрез този метод клиентът получава референция към класа **EBMClientCommunicatorRegisrator**.

2. Клиентът извиква метода register на получената в т.1 референция на класа **EBMClientCommunicatorRegisrator**, при което му подава имплементацията си на EBM API за клиенти от тип **IEBMClientCommunicator**.
3. **EBMClientCommunicatorRegisrator** проверява валидността на подадената имплементация на EBM API за клиенти и проверява дали в базата данни вече е регистриран клиент с идентификациония номер, който е подаден от клиента чрез **IEBMClientCommunicator**, използвайки **ClientCommDBAccessor** и неговият метод select.
   3.1. Ако не съществува такъв запис в базата данни, **EBMClientCommunicatorRegisrator** чрез **ClientCommDBAccessor** записва запис за новия клиент използвайки метода insert.
   3.2. **EBMClientCommunicatorRegisrator** създава нова инстанция на класа **ClientCommunicatorThread** и я добавя в кеша си, заедно с идентификационния номер на клиента. На по-късен етап, чрез идентификационният номер на клиента, референцията към **ClientCommunicatorThread** може лесно да бъде възстановена
   3.3. **EBMClientCommunicatorRegisrator** извиква **run** метода на новосъздадената инстанция на **ClientCommunicatorThread**, с което стартира тази нишка.



Фиг. 6 Регистрация на EBM клиент

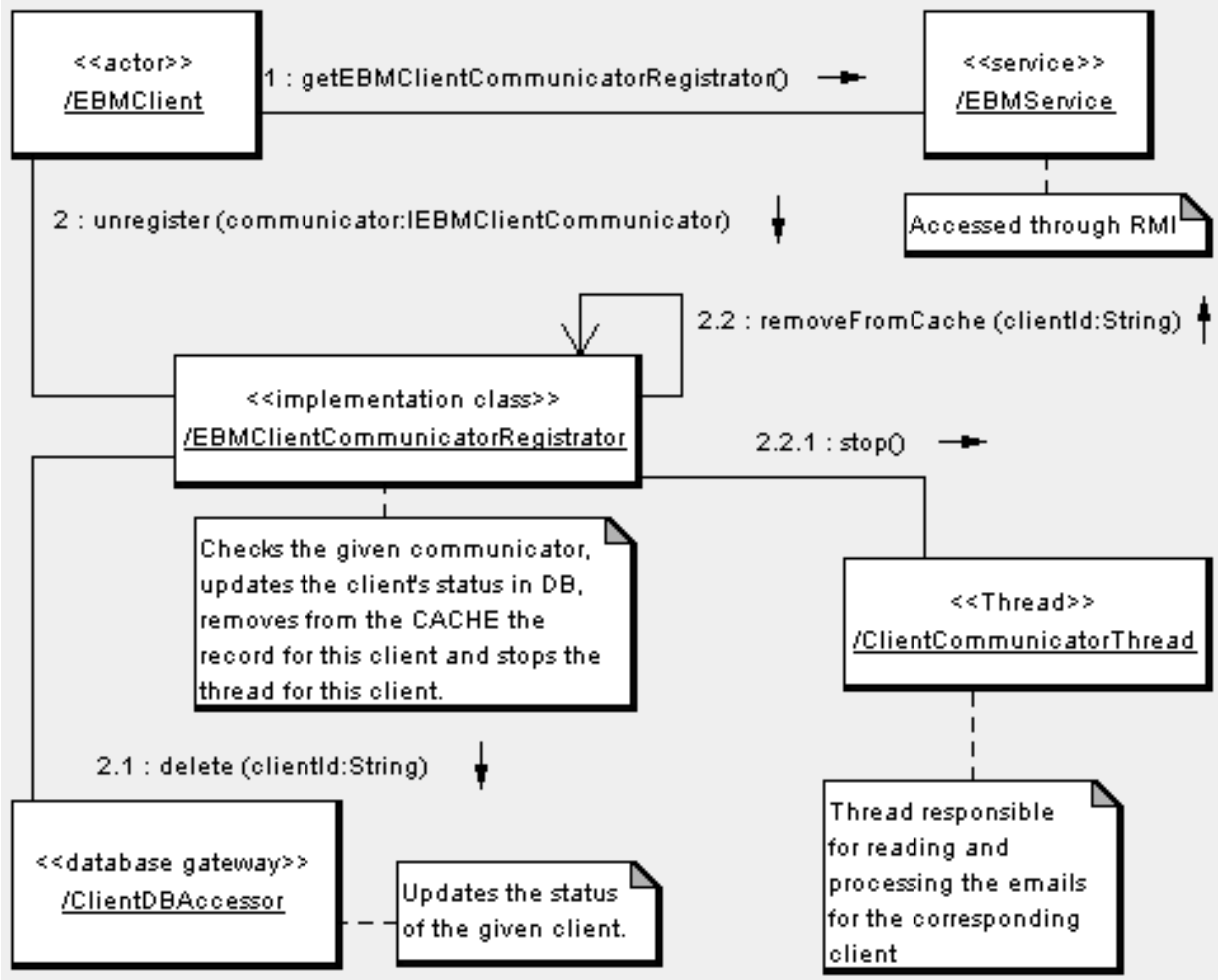
4.4.2. Отрегистриране на EBM клиент

В случай, че даден клиент желае да преустанови работата си с EBM, е необходимо, да се отрегистрира от EBM. Основните класове, които участват в

процеса на отрегистриране на ЕВМ са идентични с класовете използвани при регистрирането (Фиг. 7 Отрегистриране на ЕВМ клиент).

Процедурата по отрегистриране на ЕВМ клиент, се инициира от клиента и протича в следния ред:

1. Клиентът достъпва сървъра на ЕВМ – **EBMService** чрез RMI, след което извиква метода на ЕВМ сървъра **getEBMClientCommunicatorRegistrator**(). Чрез този метод клиента получава референция към класа **EBMClientCommunicatorRegisrator**.
2. Клиента извиква метода **unregister** на получената в т.1 референция на класа **EBMClientCommunicatorRegisrator**, при което му подава имплементацията си на ЕВМ API за клиенти от тип **IEBMClientCommunicator**.
   - 2.1. **EBMClientCommunicatorRegisrator** проверява валидността на подадената имплементация на ЕВМ API за клиенти и извиква метода delete на класа **ClientDBAccessor**, с параметър идентификационния номер на клиента. По този начин записа за клиента в базата данни се изтрива
   - 2.2. **EBMClientCommunicatorRegisrator** изтрива от своят кеш записа съдържащ клиентския идентификатор и референцията към съответният му **ClientCommunicatorThread**.
     - 2.2.1. **EBMClientCommunicatorRegisrator** извиква метода **stop** на съответстващата за отрегистрирания клиент нишка **ClientCommunicatorThread**, с което прекратява нейното действие. По този начин се преустановява четнето, обработката и триенето на имейлите на въпросния клиент.



Фиг. 7 Отрегистриране на EBM клиент

4.4.3. Изпращане на имейл

Една от най-важните функции на EBM е изпращането на имейли. Тя е и основен фактор за разграничаване на функционалността на EBM от съществуващите приложения обработващи bounce имейли.

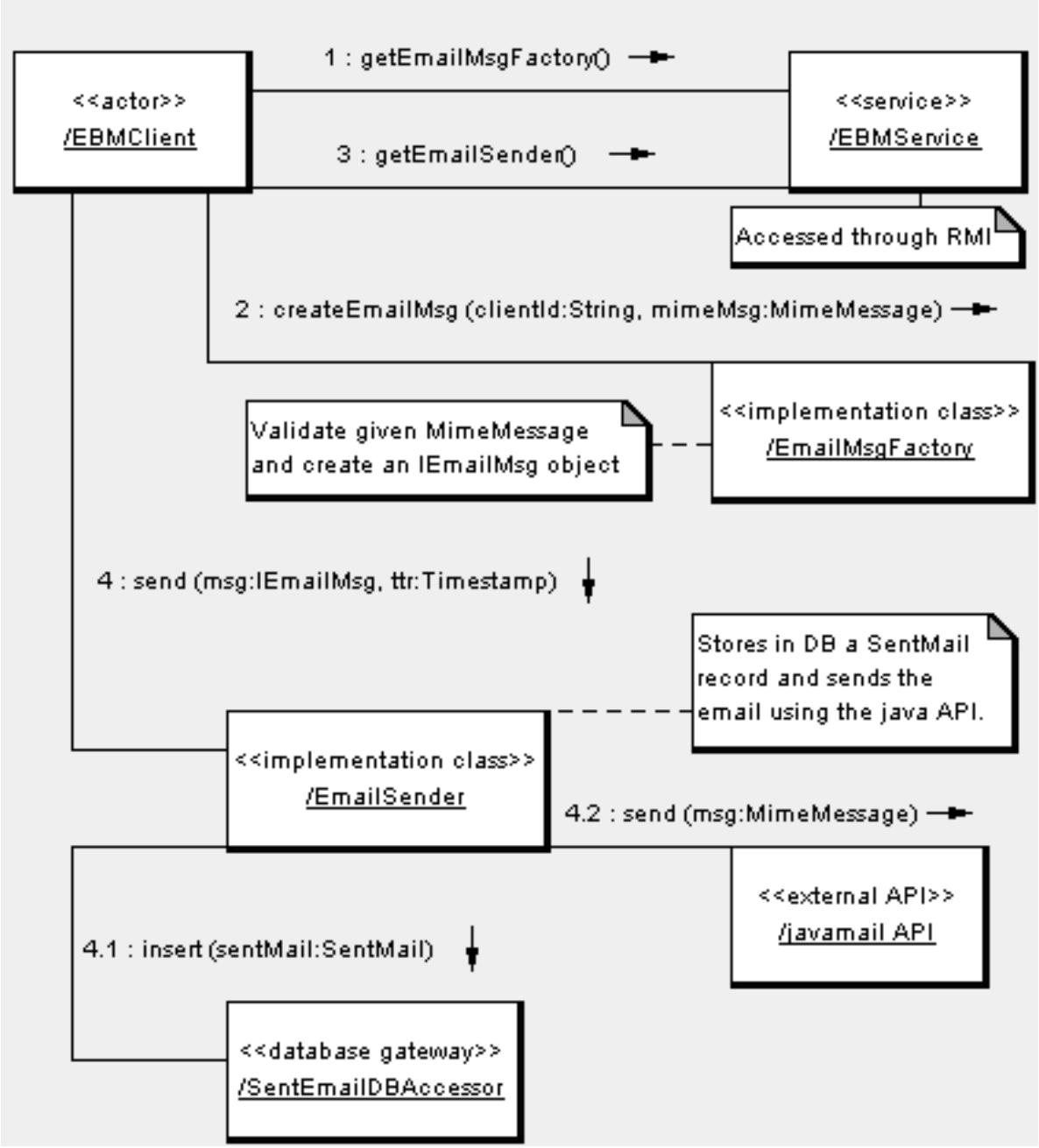
Нека разгледаме няколко основни класа, които участват в процеса на изпращене на имейли (Фиг. 8 Изпращане на имейл):

- **EBMClient** – клас, репрезентиращ клиента на EBM
- **EBMService** – клас, представляващ сървъра на EBM, достъпен чрез RMI

- **EmailMsgFactory** – клас, имплементация на интерфейса **IEmailMsgFactory**, който е част от EBM API. Служи за валидация на **MimeMessage** обекти и създаване на обекти от тип **IEmailMsg**
- **EmailSender** – клас, използван за изпращане на имейли. Представлява имплементация на **IEmailSender**. За всеки от получателите на имейла създава по една инстанция на **MimeMessage** и добавя уникален идентификационен номер към всяка от тях (този номер се използва по-късно за намиране на съответствие между изпратен и получен имейл). Изпраща всички **MimeMessage** инстанции използвайки java mail API.
- **SentEmailDBAccessor** – клас, който служи за достъп до таблицата от базата данни, в която се записва информация за изпратените от EBM имейли

Процедурата по изпращане на имейли, се инициира от клиента и протича в следния ред:

1. Клиентът **EBMClient**, достъпва **EBMService** чрез RMI и извиква метода му **getEmailMsgFactory**, чрез който получава инстанция на класа **EmailMsgFactory**.
2. **EBMClient** създава обект от тип **IEmailMsg**, като извиква метода **createEmailMsg** на **EmailMsgFactory** и му подава уникалния идентификатор на клиента и обект от тип **MimeMessage**.
3. Клиентът **EBMClient**, отново достъпва **EBMService** чрез RMI и извиква метода му **getEmailSender**, чрез който получава инстанция на класа **EmailSender**.
4. **EBMClient** извиква метода send на вече получената в т.3 инстанция на **EmailSender**, като му подава създадения в т.2 обект от тип **IEmailMsg**.
   4.1. **EmailSender** създава обект от тип **SentEmail** и го записва в базата данни, използвайки insert метода на класа **SentEmailDBAccessor**.
   4.2. **EmailSender** изпраща съответния **MimeMessage**, използвайки **javamail API**.



Фиг. 8 Изпращане на имейл

4.4.4. Получаване на имейл

Друга основна функция на EBM е получаването на имейли. Нека разгледаме няколко основни класа, които участват в процеса на изпращене на имейли (Фиг. 9 Получаване на имейл):

- **ClientCommunicatorThread** – клас (нишка), съответстваща на даден клиент на EBM. Той е отговорен за четене, обработка и изтриване на имейли съответстващи на дадения клиент през предварително определен от клиента интервал от време. Всеки имейл бива прочетен от него, подаден за обработка на **ReceivedEmailProcessor** и изтрит.
- **EBMClientEmailReader** – клас имплементация предоставена от клиента, на интерфейса **IEBMClientEmailReader**, който от своя страна е част от EBM API за клиенти. Той служи за четене и триене на имейли.
- **ReceivedEmailProcessor** – клас, който се грижи за обработката на получените имейли. Той играе особено важна роля в работата на EBM.
- **EmailMsgFactory** – имплементация на интерфейса **IEmailMsgFactory**, който е част от EBM API. Служи за валидиране и създаване на обекти от тип **IEmailMsg.**
- **SentEmailMatcher** – клас, който чрез метода си **match**, търси съответствие между полученият имейл и някой от изпратените чрез EBM имейли. При успешно откриване на такова съответствие, този метод връща съответстващият изпратен имейл.
- **ReceivedEmailDBAccessor** – клас, който служи за достъп до таблицата от базата данни, в която се записва информация за получените от EBM имейли
- **EBMClientEmailProcessor** – имплементация предоставена от клиента на **IEBMClientEmailProcessor**, който е част от EBM API за клиенти. Използва се за обработка на всеки получен имейл от страна на клиента.
- **ReceivedEmailHandler** – клас, служещ за допълнителна обработка на получените имейли. Има три **handle** метода, като се извиква един от тях, в зависимост от резултатите от направената до момента обработка, а именно:

- **handleBounceEmail** – служи за обработка на отхвърлените имейли
- **handleValidEmail** – служи за обработка на валидните имейли – тези които се явяват валиден отговор на открит от ЕВМ съответстващ изпратен имейл
- **handleNotRecognizedEmail** – служи за обработка на неразпознатите имейли

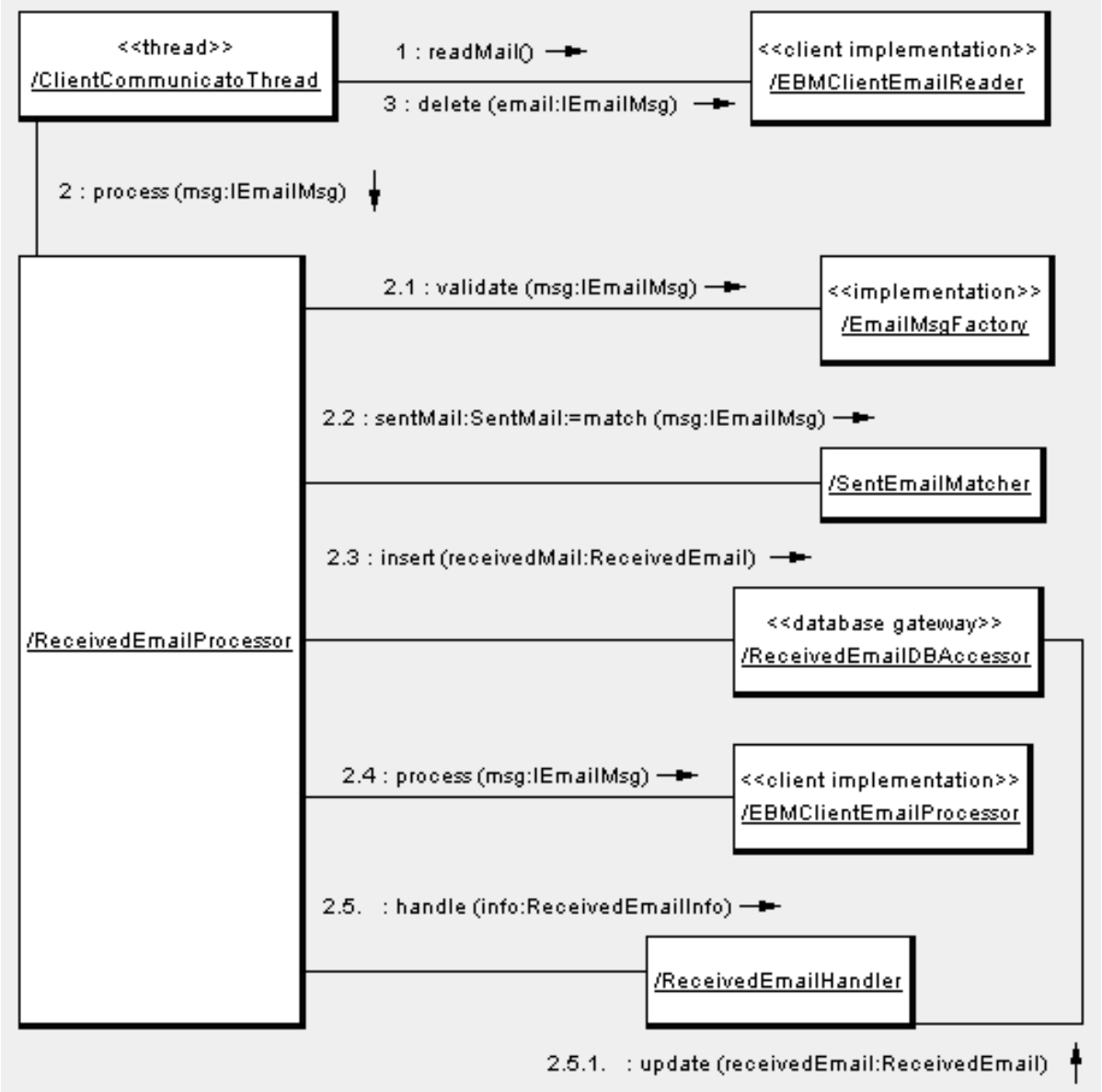
Процедурата по получаване на имейли, се изпълнява периодично през предварително дефиниран в настройките на клиента интервал. Тя се инициира от **ClientCommunicatorThread** - нишката съответстваща на клиента и протича в следния ред:

1. **ClientCommunicatorThread** извиква метода **readMail** на **EBMClientEmailReader**. В случай, че той върне прочетен имейл, се продължава към стъпка 2. В обратен случай – ако не върне имейл, то се изчаква предварителното дефинирано от клиента време, след което се повтаря стъпка 1.
2. **ClientCommunicatorThread** извиква метода **process** на класа **ReceivedEmailProcessor**, който осъществява по-нататъшната обработка на прочетеният имейл.
   - 2.1. **ReceivedEmailProcessor** проверява за валидност прочетения имейл, като използва **EmailMsgFactory** и неговият **validate** метод. В случай, че имейлът е валиден, се продължава със следващата стъпка. В противен случай, се хвърля Exception със съобщение обясняващо възникналата грешка.
   - 2.2. **ReceivedEmailProcessor** извиква **match** метода на класа **SwntEmailMatcher**, с което се опитва да открие съотващ изпратен имейл на обработваният получен имейл.
   - 2.3. **ReceivedEmailProcessor** записва в базата данни запис за получения имейл, използвайки **insert** метода на **ReceivedEmailDBAccessor**.

2.4. **ReceivedEmailProcessor** извиква **process** метода на **EBMClientEmailProcessor**, с който клиента обработва получените имейли.

2.5. **ReceivedEmailProcessor** извиква съответният **handle** метод на **ReceivedEmailHandler** (т.е. един от трите изброени по-горе методи). От своя страна **ReceivedEmailHandler** изпълнява необходимите действия, които са настроени за изпълнение (например опит за изпращане отново на имейла, в случай на soft bounce).

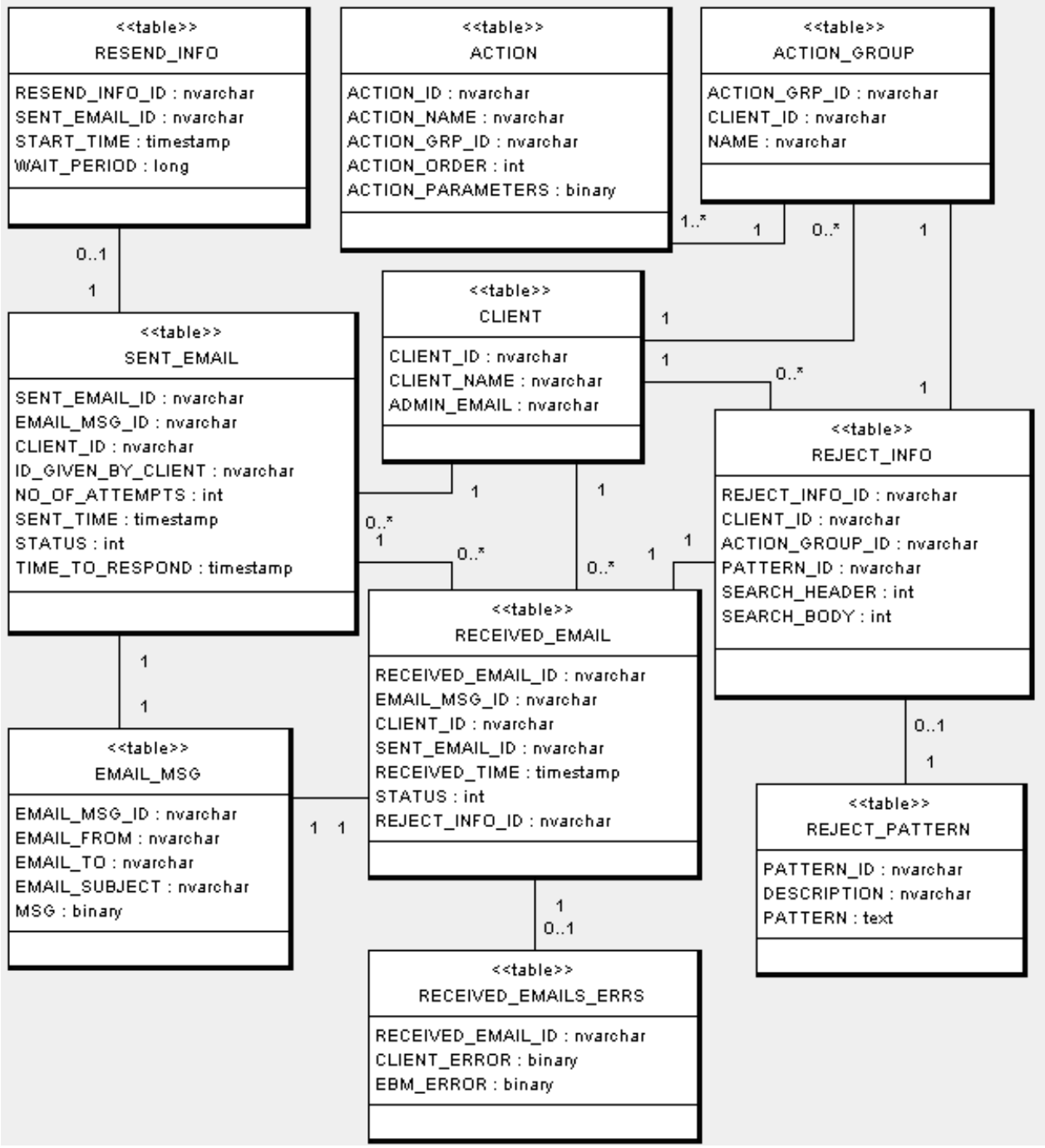
2.5.1. **ReceivedEmailHandler** променя статуса на полученият имейл, в базата данни, след като е извършена обработката му. Това се извършва чрез **update** метода на **ReceivedEmailDBAccessor.** Статуса се определя от това дали имейла е бил отхвърлен, валиден, или неразпознат.



Фиг. 9 Получаване на имейл

4.4.5. Структура на базата данни

Структурата на базата данни използвана от EBM е представена на (Фиг. 10 Структура на базата данни). Тя се състои от десет таблици, като за всяка от тях е разработен клас, чрез който се изпълняват SQL заявки свързани с нея.



Фиг. 10 Структура на базата данни

Нека разгледаме таблиците от базата данни и значението на техните полета:

- **CLIENT** – Съдържа информация за клиентите на EBM (Таблица 2 Таблица от базата данни на EBM - CLIENT)

Колона	Тип данни	Описание	Ограничения
CLIENT_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key[45] Not null[46]
CLIENT_NAME	nvarchar(64)	Име на EBM клиента	Not null
ADMIN_EMAIL	nvarchar(128)	Имейл на администратора на EBM клиента	

Таблица 2 Таблица от базата данни на EBM - CLIENT

- **ACTION_GROUP** – Съдържа описание на групите от действия, които се изпълняват при възникване на bounced mail (Таблица 3 Таблица от базата данни на EBM – ACTION_GROUP)

Колона	Тип данни	Описание	Ограничения
ACTION_GRP_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
CLIENT_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата CLIENT	Foreign key[47] Not null
NAME	nvarchar(64)	Име на групата	Not null

Таблица 3 Таблица от базата данни на EBM – ACTION_GROUP

- **ACTION** – Съдържа описание на действията, които се изпълняват при възникване на bounced mail (Таблица 4 Таблица от базата данни на EBM - ACTION)

[45] Уникален идентификатор за тази таблица
[46] Не може да приема стойност "null", т.е. задължително трябва да има стойност за всеки ред от таблицата
[47] Уникален идентификатор за съответствие с ред от друга таблица

Колона	Тип данни	Описание	Ограничения
ACTION_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
ACTION_NAME	nvarchar(64)	Име на действието	Not null
ACTION_GRP_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата ACTION_GROUP	Foreign key Not null
ACTION_ORDER	Integer	Номер на действието в групата	Not null
ACTION_PARAMETERS	varbinary	Сериализирани параметри за действието	Not null

Таблица 4 Таблица от базата данни на EBM -ACTION

- **EMAIL_MSG** – Съдържа представяне на обектите от тип MimeMessage (Таблица 5 Таблица от базата данни на EBM – EMAIL_MSG)

Колона	Тип данни	Описание	Ограничения
EMAIL_MSG_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
FROM	nvarchar(64)	Имейл адрес на изпращача	Not null
TO	nvarchar(64)	Имейл адрес на получателя	Not null
SUBJECT	nvarchar(128)	Стойността на полето относно[48]	
MSG	varbinary	Хедърите на имейла	Not null

Таблица 5 Таблица от базата данни на EBM – EMAIL_MSG

- **SENT_EMAIL** – Съдържа имейлите изпратени от EBM (Таблица 6 Таблица от базата данни на EBM – SENT_EMAIL)

[48] Subject на имейла

Колона	Тип данни	Описание	Ограничения
SENT_EMAIL_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
EMAIL_MSG_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата EMAIL_MSG	Foreign key Not null
CLIENT_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата CLIENT	Foreign key Not null
ID_GIVEN_BY_CLIENT	nvarchar(32)	Уникален идентификатор даден от клиента	
NO_OF_ATTEMPTS	integer	Брой опити за изпращане на имейла	Not null
SENT_TIME	timestamp	Дата и час на изпращане на имейла	Not null
STATUS	integer	Статус на изпратения имейл	Not null
TIME_TO_RESPOND	timestamp	Дата и час до които се чака отговор на изпратения имейл	Not null

Таблица 6 Таблица от базата данни на EBM – SENT_EMAIL

- **REJECT_INFO** – Съдържа информация за шаблоните за откриване на причината за отхвърлен имейл свързани с даден клиент (Таблица 7 Таблица от базата данни на EBM – REJECT_INFO)

Колона	Тип данни	Описание	Ограничения
REJECT_INFO_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
CLIENT_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата CLIENT	Foreign key Not null
ACTION_GROUP_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата ACTION_GROUP	Foreign key Not null
PATTERN_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата PATTERN	Foreign key Not null
SEARCH_HEADER	integer	Ако е 1, шаблона трябва да се приложи върху хедърите на имейла	Not null
SEARCH_BODY	integer	Ако е 1, шаблона трябва да се приложи върху съдържанието на имейла	Not null

Таблица 7 Таблица от базата данни на EBM – REJECT_INFO

- **RECEIVED_EMAIL** – Съдържа имейлите получени от EBM (Таблица 8 Таблица от базата данни на EBM – RECEIVED_EMAIL)

Колона	Тип данни	Описание	Ограничения
RECEIVED_EMAIL_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
EMAIL_MSG_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата EMAIL_MSG	Foreign key Not null
CLIENT_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата CLIENT	Foreign key
SENT_EMAIL_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата SENT_EMAIL	Foreign key
RECEIVED_TIME	timestamp	Датата и часа в които е получен имейла	Not null
STATUS	integer	Статус на полученият имейл	Not null
REJECT_INFO_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата REJECT_INFO	Foreign key

Таблица 8 Таблица от базата данни на EBM – RECEIVED_EMAIL

- **REJECT_PATTERN** – Съдържа всички шаблони за откриване на причината за отхвърлен имейл (Таблица 9 Таблица от базата данни на EBM – REJECT_PATTERN)

Колона	Тип данни	Описание	Ограничения
PATTERN_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
DESCRIPTION	nvarchar(128)	Описание на причината	
PATTERN	text	Шаблон (регулярен израз)	Not null

Таблица 9 Таблица от базата данни на EBM – REJECT_PATTERN

- **RECEIVED_EMAILS_ERRS** – Съдържа грешките, които са възникнали при четене на имейлите (Таблица 10 Таблица от базата данни на EBM – RECEIVED_EMAILS_ERRS)

Колона	Тип данни	Описание	Ограничения
RECEIVED_EMAIL_ID	nvarchar(32)	Уникален идентификатор за тази таблица, служи и за връзка с таблицата RECEIVED_EMAIL	Primary key Foreign key Not null
CLIENT_ERROR	blob	Съобщение за грешка възникнала при клиента по време на обработка на получен имейл	Not null
EBM_ERROR	blob	Съобщение за грешка възникнала при EBM по време на обработка на получен имейл	

Таблица 10 Таблица от базата данни на EBM – RECEIVED_EMAILS_ERRS

- **RESEND_INFO** – Съдържа информация за имейли, които трябва да се изпратят отново (Таблица 11 Таблица от базата данни на EBM – RESEND_INFO)

Колона	Тип данни	Описание	Ограничения
RESEND_INFO_ID	nvarchar(32)	Уникален идентификатор за тази таблица	Primary key Not null
SENT_EMAIL_ID	nvarchar(32)	Уникален идентификатор за връзка с таблицата SENT_EMAIL	Foreign key Not null
START_TIME	timestamp	Дата и час в които трябва да започне изпращането	Not null
WAIT_PERIOD	long	Време за изчакване между отделните изпращания	Not null

Таблица 11 Таблица от базата данни на EBM – RESEND_INFO

5. Тестване на приложението

За опростяване на тестовете, е имплементиран само един клиент на приложението и условията разгледани в настоящата глава са съобразени с този факт. Тъй като въпросният клиент е създаден с тестови цели и единствената функционалност, която той предлага е имплементацията на изискваните от ЕВМ приложението програмни интерфейси, необходими за изпращане, четене, изтриване на имейли, както и за регистриране и отрегистриране за работа с ЕВМ. Самото тестване на приложението е базирано на основни сценарии на използване на приложението, които са разгледани в тази глава. Тестовете са извършвани като приложението и неговият клиент са стартирани на машина(и), която има достъп до имейл сървър, чрез който се изпращат и получават имейлите.

5.1. Необходими условия за тестване на приложението

За да бъде стартирано приложението и за да се гарантира безпроблемната му работа, е необходимо да са изпълнени следните условия:

- На машините, на които ще бъдат стартирани клиентската и сървърната части от приложението, трябва да има инсталирана Java версия 1.5.0_11 или по-нова версия. На имейл сървъра трябва да са създадени три имейл акаунта : “admin” (с парола “admin”), “test” (с парола “test”) и “client” (с парола “client”). Първият акаунт се използва за имейл на администратора на клиента на ЕВМ при нотификация за възникнал проблем, вторият се използва за получател на тестовите писма, а третият се използва като служебен адрес на клиента на ЕВМ.
- И двете машини трябва да имат достъп до работещ имейл сървър – приложението е тествано с James версия 2.3.1

- Трябва да има мрежова връзка между сървърната и клиентската машини
- На сървърната машина трябва да има инсталиран MySQL сървър. Приложението е тествано с версия 5.0
- Необходимо е да са изпълнени SQL заявките за създаване на таблиците необходими за работата на EBM (Приложение 9.1.)
- Необходимо е да са изпълнени SQL заявките за въвеждане на регулярните изрази свързани с разпознаването на съобщение за грешка в даден получен имейл (Приложение 9.2.)

5.2. Стартиране на приложението и неговият клиент

За стартиране на EBM, е необходимо да се изпълнят следните команди:

java -cp
"D:/EBM/EmailBounceManager/;D:/EBM/EmailBounceManagerAPI/;D:/EBM/lib/jaf/activation.jar;D:/EBM/lib/javamail/mail.jar;D:/EBM/lib/mysql-connector-java-bin.jar" -Djava.security.policy="D:/EBM/EmailBounceManager/security.policy" -Djava.rmi.server.hostname=127.0.0.1 -Djava.rmi.server.codebase="file:/D:/EBM/EmailBounceManager/" ebm.impl.service.EBMServiceController

Където:

- *Djava.security.policy* – указва къде стои файлът съдържащ настройките за сигурност и правата свързани с EBM приложението
- *Djava.rmi.server.codebase* – настройка, която сочи към файл(ове) и/или директорийна(и) структура(а) за класове, които са част от приложението и могат да бъдат достъпвани от разстояние чрез RMI
- *Djava.rmi.server.hostname=127.0.0.1* – указва името или адреса на машината, на която ще се експортират стъбове на отдалечени обекти във Java виртуалната машина. В случай, че EBM и неговият клиент са разположени на различни машини, тази настройка

трябва да бъде променена да сочи към IP адреса или хост името на машината на която е стартирано приложението.

За стартиране на клиента на EBM, е нужно да се изпълнят следните команди:

java -cp "D:/EBM/EmailBounceManagerClient/;D:/EBM/EmailBounceManagerAPI/;D:/EBM/lib/jaf/activation.jar;D:/EBM/lib/javamail/mail.jar;D:/EBM/lib/mysql-connector-java-bin.jar" -Djava.security.policy="D:/EBM/EmailBounceManagerClient/security.policy" -Djava.rmi.server.codebase="file:/D:/EBM/EmailBounceManagerClient/" ebm.client.service.EBMClientUI

Където:

- *Djava.security.policy* – указва къде стои файлът съдържащ настройките за сигурност и правата свързани с клиента на EBM
- *Djava.rmi.server.codebase* – настройка, която сочи към файл(ове) и/или директорийна(и) структура(а) за класове, които са част от клиента и могат да бъдат достъпвани от разстояние чрез RMI

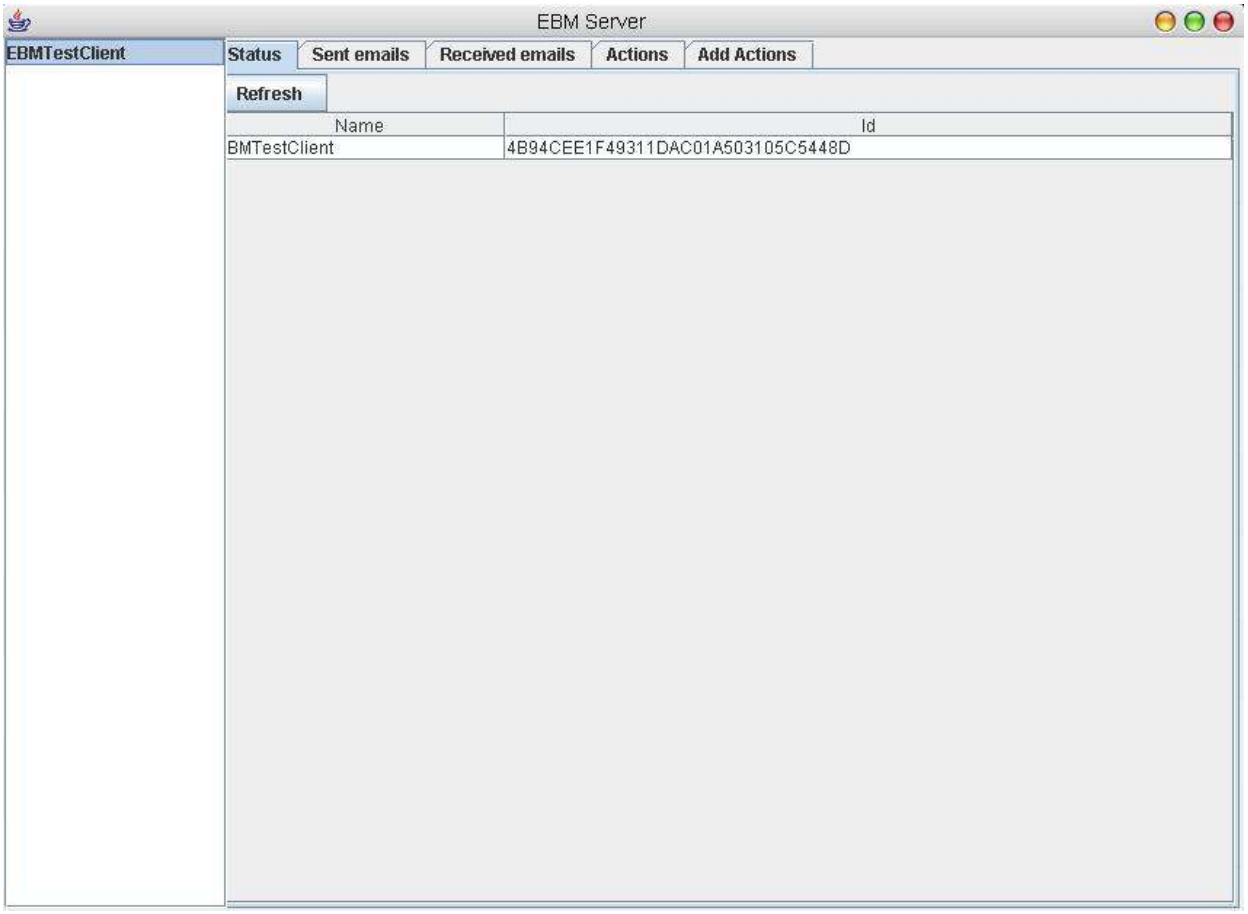
За улеснение при стартирането на приложението и неговият клиент, към тях са добавени два скриптови файла за Windows, които изпълняват горните команди. Имената им са съответно "EBMServer.bat" за стартиране на приложението и "EBMClient.bat" за стартиране на клиента.

5.3. Регистриране на клиент

След като бъде стартиран, всеки клиент на EBM трябва да бъде регистриран за работа с EBM, преди да може да изпраща и получава имейли. За успешното регистриране на даден клиент, е нужно да се изпълнят следните стъпки в указаната последователност:

- Стартиране на RMI registry на сървърната машина

- Стартиране на приложението на сървърната машина (Фиг. 11 Стартиране на EBM)
- Стартиране на клиентското приложение на клиентската машина (Фиг. 12 Стартиране на тестовият клиент на EBM)
- Натискане на бутона "Register client" от клиентското приложение
- Появява се съобщение за потвърждение за извършената регистрация (Фиг. 13 Потвърждение за регистрация)



Фиг. 11 Стартиране на EBM

Фиг. 12 Стартиране на тестовият клиент на EBM

Фиг. 13 Потвърждение за регистрация

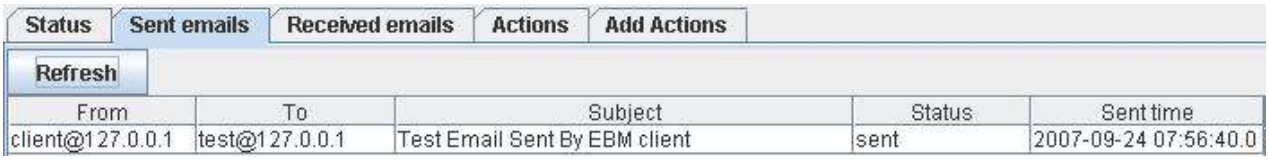
5.4. Изпращане на имейли

Изпращането на имейл от клиента, става след успешното стартиране на приложението и самия клиент и след успешната регистрация на клиента. За изпращане на предварително дефиниран в клиентското приложение тестов имейл, е необходимо да се натисне бутона "Send" (Фиг. 14 Изпращане на имейл). При успешно изпращане на имейл, се появява съобщение за потвърждение на успешното изпращане (Фиг. 15 Потвърждение за изпратен имейл).

Фиг. 14 Изпращане на имейл

Фиг. 15 Потвърждение за изпратен имейл

След успешно изпращане на имейл, информация за него може да се види в менюто на EBM "Sent emails", при натискане на бутона "Refresh", като неговото първоначално състояние е "sent" (Фиг. 16 Изпратен имейл в EBM).



Фиг. 16 Изпратен имейл в EBM

5.5. Настройки за действия за изпълнение при откриване на soft bounce

Настройките за действия за изпълнение, при откриване на отхвърлени имейли е базирано на шаблоните за тяхното откриване. За всеки получен от EBM имейл, се проверява дали съвпада с настроените шаблони. Откриване на съответствие с някой от тях, означава, че е възникнал soft bounce и се извличат всички групи от действия и съответните им действия, които трябва да се изпълнят в този случай. Действията в настоящата версия на EBM са два вида:

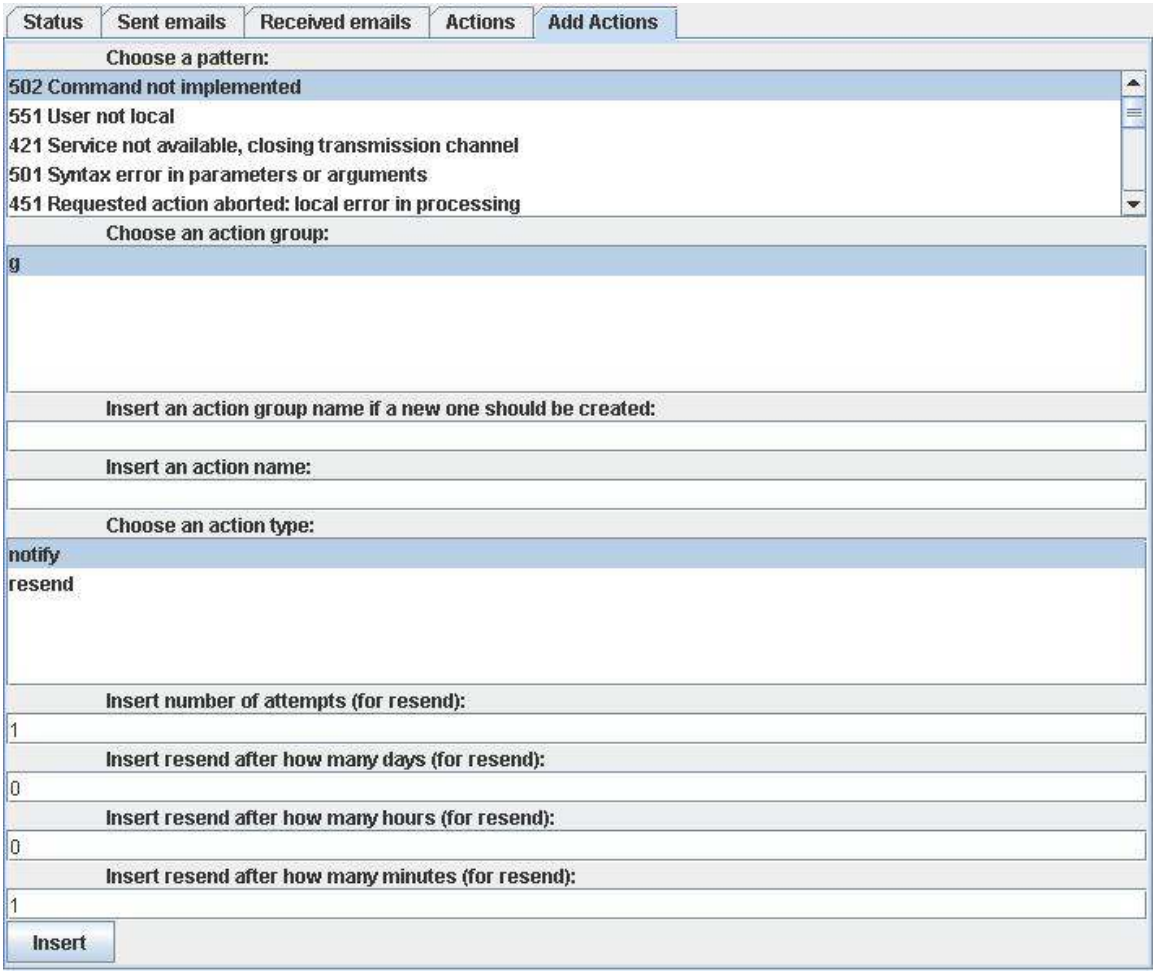
- изпращане на имейл на администратора на клиента, уведомяващ го за възникналият проблем (notify)
- изпращане отново на имейла на който съответства полученият имейл, ако такова съответствие е открито (resend). Има възможност да се настроят броят опити за изпращане (number of

attempts) и на какъв интервал от време да се извършва изпращането – съответно дни, часове и минути

5.5.1. Добавяне на настройки

За добавяне на настройки за действия, които да се изпълнят в случай, че даден получен на адреса на клиента имейл, бъде класифициран от EBM като soft bounce, е нужно да се попълни формулярът, който е част от потребителския интерфейс на EBM. Той се намира в менюто "Add Actions" (Фиг. 17 Добавяне на настройки).

Добавянетона настройки може да се раздели на няколко основни вида.



Фиг. 17 Добавяне на настройки

5.5.1.1. Добавяне на вече съществуваща група от настройки към даден шаблон

За целта е необходимо да съществува поне една създадена група от настройки. Следните параметри трябва да са попълнени:

- Избира се група от настройки от списъка от групи под надписа "Choose an action group"
- Избира се един от шаблоните, с който да се свърже избраната група от списъка с шаблони под надписа "Choose a pattern"

5.5.1.2. Добавяне на нова група от настройки към даден шаблон

За целта се попълват следните параметри:

- Избира се един от шаблоните с който да се свърже новата група от списъка с шаблони под надписа "Choose a pattern"
- Попълва на името на новата група в полето "Insert an action group name if a new one should be created"
- Въвежда се име на действие, което да се изпълни в полето "Insert an action name"
- Избира се едно от възможните предефинирани действия от списъка "Choose an action type" – нотифициране, или препращане отново на вече изпратения имейл

В случай на нотифициране, се настройват:

- Брой нови изпращания
- Брой дни между отделните изпращания
- Брой часове между отделните изпращания
- Брой минути между отделните изпращания

5.5.1.3. Добавяне на действие към вече съществуваща група

За целта се попълват следните параметри:

- Избира се група от настройки от списъка от групи под надписа "Choose an action group"
- Ако се избере един от шаблоните с който да се свърже избраната група от списъка с шаблони под надписа "Choose a pattern", тя ще се с свърже с него. Ако тя вече е свързана с него, никакво действие няма да се предприеме
- Въвежда се име на действие, което да се изпълни в полето "Insert an action name"
- Избира се едно от възможните предефинирани действия от списъка "Choose an action type" – нотифициране, или препращане отново на вече изпратения имейл

В случай на нотифициране се настройват:

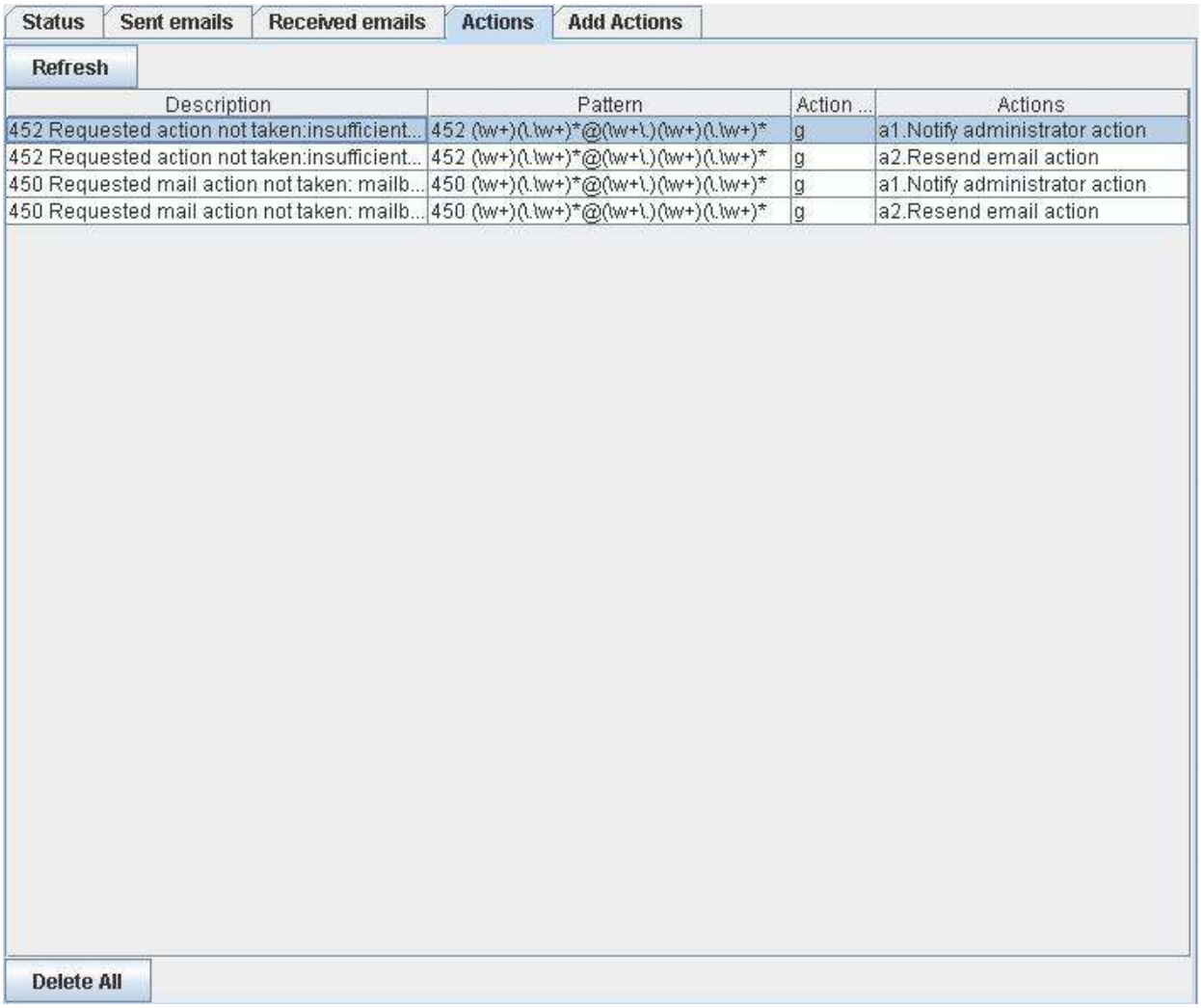
- Брой нови изпращания
- Брой дни между отделните изпращания
- Брой часове между отделните изпращания
- Брой минути между отделните изпращания

5.5.2. Разглеждане на настройки

Разглеждането на настройки се извършва при натискане на таб менюто "Actions" от приложението (Фиг. 18 Разглеждане на настройки). Там могат да се разгледат добавените чрез "Add Actions" настройки. Те са представени в таблица със следните колони:

- Описание на шаблона (Description)
- Регулярен израз на шаблона (Pattern)
- Име на група (Action Group)

- Име на действие от групата и кратко описание на вида действие (Actions)



Фиг. 18 Разглеждане на настройки

5.5.3. Изтриване на настройки

Изтриването на настройки се извършва от таб менюто на приложението за разглеждане на настройки "Actions". За да се изтрият всички въведени настройки, е нужно да се натисне бутона "Delete All" (Фиг. 18 Разглеждане на настройки), като при успешно изтриване, се появява диалогов прозорец с

надпис "All actions information is deleted successfully" (Фиг. 19 Потвърждение за изтриване).

Фиг. 19 Потвърждение за изтриване

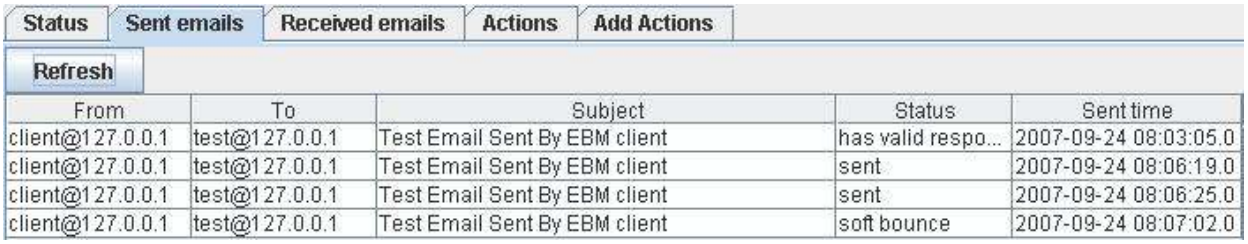
5.6. Разглеждане на списъците с изпратени и получени имейли

При изпращането или получаването на имейл, чрез ЕВМ, той се записва в базата данни на ЕВМ и информация за него може да се разгледа чрез едно от двете таб менюта – "Sent emails" съответно за изпратени имейли и "Received emails" съответно за получени имейли.

5.6.1. Разглеждане на списъка с изпратените имейли

При натискане на таб менюто "Sent emails" (Фиг. 20 Списък с изпратените имейли), могат да се разгледат изпратените чрез ЕВМ имейли, които са подредени в таблица със следните колони:

- Имейл на изпращача (From)
- Имейл на получателя (To)
- Тема на имейла (Subject)
- Статус на имейла (Status) – той може да бъде: изпратен (sent), неполучен (soft bounce), отхвърлен - hard bounce (lost), с получен валиден отговор (has valid response)
- Време на изпращане на имейла (Sent time)

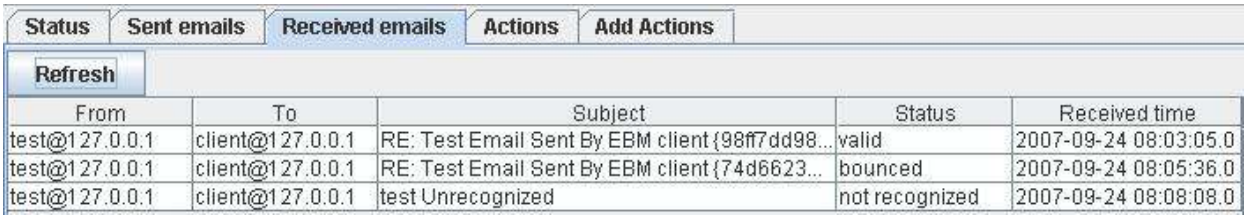


Фиг. 20 Списък с изпратените имейли

5.6.2. Разглеждане на списъка с получените имейли

За разглеждане на списъка с получените имейли, е необходимо да се избере таб менюто "Received emails" (Фиг. 21 Списък с получените имейли) от потребителския интерфейс на приложението. Списъка е подреден в таблица със следните колони:

- Имейл на изпращача (From)
- Имейл на получателя (To)
- Тема на имейла (Subject)
- Статус на имейла (Status) – той може да бъде: в процес на обработка – преходен служебен статус (processing), възникнала грешка (error), валиден – намерен е съответният изпратен имейл и не е открит bounce (valid), изпратеният имейл не е получен – възникнал е soft bounce (bounced), неразпознат – в случай, че не е открит съответен изпратен имейл (not recognized), обработен – преходен служебен статус (processed)
- Време на получаване на имейла (Received time)



Фиг. 21 Списък с получените имейли

5.7. Получаване на имейл

След успешната регистрация на даден клиент за работа с ЕВМ, приложението започва да проверява за нов имейл в настроената пощенска кутия на клиента на предварително дефиниран интервал от време. Ако такъв имейл е пристигнал, ЕВМ го обработва по един от долоизброените начини, след което го изтрива от пощенската кутия, за да не бъде прочетен отново.

5.7.1. Обработка при валиден отговор

При получаване на имейл, ЕВМ се опитва да открие съответният му изпратен имейл и да провери дали полученото писмо не е bounce съобщение. Ако съответен изпратен имейл е получен и полученият имейл не бъде разпознат като неполучен, той се приема за валиден и се записва в списъка с получените имейли със статус валиден (valid). Съответният изпратен имейл преминава в статус "has valid response".

5.7.2. Обработка при неразпознат получен имейл

Ако при получаване на имайл, не бъде открит съответният му изпратен имейл, то той се приема за неразпознат от гледна точка на ЕВМ (not recognized). В този случай писмото се записва за статистически причини и евентуално преглеждане от администратора на приложението или неговият клиент.

5.7.3. Обработка при soft bounce

В случай, че при получаване на имейл, ЕВМ успее да го идентифицира чрез настроените шаблони, като неполучен и съответното изпратено писмо

бъде открито, то полученият имейл преминава в статус "bounced". В този случай, се изпълняват настроените действия за въпросният вид възникнал проблем – изпращане отново на първоначално изпратеният имейл и/или изпращане на администратора на клиента съобщение, за възникналият проблем. Съответният изпратен имейл преминава в статус "soft bounce".

5.7.4. Обработка при hard bounce

В случай, че за даден изпратен имейл, не бъде получен отговор за предварително определено време, то той се счита за изгубен, т.е. възникнал е hard bounce и съответното изпратено писмо преминава в статус "lost".

5.8. Тестване на пиложението при валиден получен имейл

При този сценарий на използване на ЕВМ, се изпълняват следните стъпки:

- Клиента изпраща имейл (Глава 5.4.)
- Изпратеният имейл, може да си види в списъка с изпратените имейли (Глава 5.6.1.), като той има статус "sent"
- Получателят отваря пристигналият имейл, отговаря му (reply) с някакво съобщение и го изпраща
- ЕВМ прочита отговора и го записва в списъка с получени имейли със статус "valid" (Глава 5.7.1.), а изпратеният имейл преминава в статус "has valid response" (може да се види в списъка на изпратените имейли – Глава 5.6.1.)

5.9. Тестване на приложението при soft bounce

При този сценарий на използване на ЕВМ, се изпълняват следните стъпки:

- Настройва се действие за изпълнение при откриване на soft bounce препращане (resend) и/или нотификация (notify) за някой от шаблоните. Нека изберем например шаблона "450 Requested mail action not taken: mailbox unavailable".
- Клиента изпраща имейл (Глава 5.4.)
- Изпратеният имейл, може да си види в списъка с изпратените имейли (Глава 5.6.1.), като той има статус "sent"
- Получателят отваря пристигналият имейл, отговаря му (reply), като в съобщението записва "450 test@test.com". По този начин би следвало настроеният в първата стъпка шаблон да разпознае имейла като soft bounce
- EBM прочита отговора и го записва в списъка с получени имейли със статус "bounced", а съответният изпратен имейл преминава в статус "soft bounce" (Глава 5.7.3.)

5.10. Тестване на приложението при изгубен имейл

За осъществяване на този сценарий, е необходимо да се изпрати имейл (Глава 5.4.) и да не се отговаря на този имейл достатъчно дълго време, така че той да бъде определен от EBM като изгубен (Глава 5.7.4.).

5.11. Тестване на приложението при неразпознат имейл

За осъществяване на този сценарий, е необходимо да се подаде произволен имейл, който не съдържа уникален идентификатор, на EBM. По този начин EBM няма да може да разпознае това съобщение и то ще бъде определено като неразпознато (Глава 5.7.2.).

6. Заключение и насоки за бъдещо развитие на приложението

Предложеното с настоящата дипломна работа решение за bounce проблема (EBM), за разлика от наличните на програмни продукти предлага и справяне с проблма свързан с неполучените и отхвърлени имейли, за които не е върнат отговор. В настоящата си версия, EBM изпълнява основната си цел, но съществуват редица подобрения, които биха могли значително да повишат качеството на работата с него.

В следващи версии на програмата би могло да се работи в следните насоки за подобрение:

- Добавяне на други действия, които да се предприемат в случай на bounce, освен препращане на изпратения имейл и нотификация на администратора на клиента на EBM чрез имейл.
- Добавяне на настройка указваща, дали да се трият или не имейлите и данните за тях, чрез IEBMClientEmailReader, след като бъдат прочетени. Възможно е да се маркират като изтрити, но да остават като записи в базата данни с цел статистика.
- Подобряване на сигурността свързана със съхранението на настройките. В текущата версия някои настройки се пазят като полета в сорс кода без да бъдат криптирани.
- Извеждане на основни настройки в конфигурационен файл и/или в базата данни с цел по-удобна конфигурация.
- Добавяне на списък с имейли от и към които EBM съответно да не получава и изпраща имейли. Целта на този списък е там да се записват адресите, при комуникацията с които е възникнал hard bounce.
- Добавяне на статистика в която да се съхраняват стари данни.
- Прибавяне на функционалност за изтриване на отделни настройки за действия за изпълнение при възникване на soft

bounce в менюто "Actions" на EBM. В настоящата версия е възможно само изтриването на всички настройки.

- Добавяне на поддръжка на лог файлове.
- Подобряване на сигурността при ползване на RMI. В настоящата версия, сигурността е значително занижена.
- Добавяне на възможност за четене на имейли от повече от една пощенска кутия за клиент.
- Добавяне на възможност за сортиране на списъците с изпратени и получени имейли по съответните им колони.
- Улесняване на инсталацията на EBM, чрез автоматизиране на създаването на таблиците в базата данни и въвеждането на шаблоните за разпознаване на bounced имайли и записа(ите) за клиента(ите) на EBM.

Като заключение, въпреки предимството на EBM спрямо предлаганите на пазара приложения, за да може предложеното решение да се конкурира с тях, е необходимо да се направят редица подобрения в имплементацията му. Настоящата му версия, обаче е една стабилна основа за по-нататъшно развитие, като предлага в основата си функционалност различна от другите решения.

7. Списък на използваните съкращения и термини

В настоящата дипломна работа са използвани следните съкращения и термини:

1. domain – домейн
2. SPAM – непоискан от получателят имейл
3. User Unknown – неразпознат потребител
4. ISP – Internet Service Provider – интернет доставчик
5. Mailbox Full – пълна пощенска кутия
6. Message exceeds size limit – съобщението надхвърля допустимия размер
7. Headers – служебна информационна част от имейл
8. Domain Not Found – домейнът не е открит
9. blacklist – черен списък от хора, IP адреси и т.н. с които е забранена комуникацията
10. Too many hops – Прекалено дълъг път на рутиране
11. Routing loop – цикъл по време на рутиране
12. alias – псевдоним
13. Connection Refused – Отказана връзка
14. Relay Access Denied – Достъпът за препредаване е отказан
15. Residential Internet Protocol address – Домашен IP адрес
16. Digital Subscriber Line – DSL – Цифрова линия за връзка
17. Local Area Network – Локална мрежа
18. America Online - AOL
19. Unknown – непознат, неизвестен
20. Request For Comment – Документ с пореден номер от серия такива документи за публикуване на стандарти
21. Message rejected – Съобщението е отхвърлено
22. Service not available, closing transmission channel – Сервиза не е на разположение, комуникационният канал ще бъде прекъснат

23. Requested mail action not taken: mailbox unavailable – Заявеното имейл операция не е осъществено: пощенската кутия не е открита
24. Requested action aborted: local error in processing – Заявеното действие е прекъснато преди да бъде изпълнено: възникнала е локална грешка по време на обработката
25. Simple Mail Transfer Protocol – SMTP – протокол за изпращане и получаване на имейли
26. Requested action not taken: insufficient system storage – Заявеното действие не е осъществено: недостатъчно системно пространство за съхранение
27. Syntax error, command unrecognized – Синтактична грешка, командата не може да бъде разпозната
28. Syntax error in parameters or arguments – Синтактична грешка свързана с параметрите или аргументите
29. Command not implemented – Командата не е имплементирана
30. Bad sequence of commands - Неправилна последователност от команди
31. Command parameter not implemented – Параметър на командата не е имплементиран
32. Requested action not taken: mailbox unavailable – Заявената операция не е осъществена: пощенската кутия не е открита
33. User not local – Потребителят не е локален
34. Requested mail action aborted: exceeded storage allocation – Заявената имейл операция е прекъсната преди да бъде изпълнена: надхвърлен е максималният размер на пространството за съхранение на имейли
35. Requested action not taken: mailbox name not allowed – Заявената операция не е изпълнена: използвано е непозволено име на пощенска кутия
36. Transaction failed – Неуспешна транзакция
37. E-mail list – списък от имейли

38. Comma Separated Values – CSV – файлов формат в който информацията е разделена със запетая
39. Source code – изходен код
40. EBM – Email Bounce Handler – Програма за обработка на bounce имейли
41. Unique ID – уникален идентификатор
42. Out-of-office auto reply – съобщение изпратено автоматично, което уведомява, че адресата временно не може да отговаря на кореспонденцията си
43. Regular expressions – регулярни изрази
44. Use cases – примерни начини на употреба
45. Primary key – Уникален идентификатор за дадена таблица от база данни
46. Not null – Не може да приема стойност "null", т.е. задължително трябва да има стойност за всеки ред от таблицата
47. Foreign key – Уникален идентификатор за съответствие с ред от друга таблица
48. Subject of an email – стойност на полето относно на даден имейл

8. Използвана литература

1. Understanding Bounce E-mails, http://www.maxprog.com/support/us/emailbouncehandler/understanding_bounce_e-mails.html
2. Simple Mail Transfer Protocol, http://www.rfc-editor.org/rfc/rfc2821.txt
3. Information on bounced messages, http://help.excite.com/email/bounce.html
4. eMail Bounce Handler™, http://www.maxprog.com/EmailBounceHandler.html
5. Bounce eMail Manager, http://www.bouncemailmanager.com
6. G-Lock Email Processor, http://www.glocksoft.com/ep/index.htm
7. b*Bounce Server 2005, http://www.bbounce.com/Server/About.aspx
8. BoogieTools, http://www.boogietools.com
9. switchemail, http://www.switchemail.com/index.asp

9. Приложение с SQL заявки необходими за работата на приложението

9.1. SQL заявки необходими за създаването на таблиците в базата данни на EBM

За създаване на таблиците, необходими за работата на EBM, е нужно да се изпълнят следните SQL заявки на MySQL сървъра:

- CREATE TABLE `ACTION_GROUP` (`ACTION_GRP_ID` VARCHAR(32) NOT NULL, `CLIENT_ID` VARCHAR(32) NOT NULL, `NAME` VARCHAR(256) NOT NULL, PRIMARY KEY (`ACTION_GRP_ID`));

- CREATE TABLE `ACTION` (`ACTION_ID` VARCHAR(32) NOT NULL, `ACTION_NAME` VARCHAR(32) NOT NULL, `ACTION_GRP_ID` VARCHAR(32) NOT NULL, `ACTION_ORDER` INT NOT NULL, `ACTION_PARAMETERS` VARBINARY(512), PRIMARY KEY (`ACTION_ID`));

- CREATE TABLE `RESEND_INFO` (`RESEND_INFO_ID` VARCHAR(32) NOT NULL, `SENT_EMAIL_ID` VARCHAR(32) NOT NULL, `START_TIME` TIMESTAMP NOT NULL, `WAIT_PERIOD` BIGINT NOT NULL, PRIMARY KEY (`RESEND_INFO_ID`));

- CREATE TABLE `CLIENT` (`CLIENT_ID` VARCHAR(32) NOT NULL, `CLIENT_NAME` VARCHAR(256) NOT NULL, `ADMIN_EMAIL` VARCHAR(256), PRIMARY KEY (`CLIENT_ID`));

- CREATE TABLE `SENT_EMAIL` (`SENT_EMAIL_ID` VARCHAR(32) NOT NULL, `EMAIL_MSG_ID` VARCHAR(32) NOT NULL, `CLIENT_ID` VARCHAR(32) NOT NULL, `ID_GIVEN_BY_CLIENT` VARCHAR(32), `NO_OF_ATTEMPTS` INT NOT NULL, `SENT_TIME` TIMESTAMP,

`STATUS` INT NOT NULL, `TIME_TO_RESPOND` TIMESTAMP, PRIMARY KEY (`SENT_EMAIL_ID`));

- CREATE TABLE `REJECT_INFO` (`REJECT_INFO_ID` VARCHAR(32) NOT NULL, `CLIENT_ID` VARCHAR(32) NOT NULL, `ACTION_GROUP_ID` VARCHAR(32), `PATTERN_ID` VARCHAR(32) NOT NULL, `SEARCH_HEADER` INT NOT NULL, `SEARCH_BODY` INT NOT NULL, PRIMARY KEY (`REJECT_INFO_ID`));

- CREATE TABLE `RECEIVED_EMAIL` (`RECEIVED_EMAIL_ID` VARCHAR(32) NOT NULL, `EMAIL_MSG_ID` VARCHAR(32) NOT NULL, `CLIENT_ID` VARCHAR(32) NOT NULL, `SENT_EMAIL_ID` VARCHAR(32), `RECEIVED_TIME` TIMESTAMP, `STATUS` INT NOT NULL, `REJECT_INFO_ID` VARCHAR(32), PRIMARY KEY (`RECEIVED_EMAIL_ID`));

- CREATE TABLE `REJECT_PATTERN` (`PATTERN_ID` VARCHAR(32) NOT NULL, `DESCRIPTION` VARCHAR(256) NOT NULL, `PATTERN` TEXT NOT NULL, PRIMARY KEY (`PATTERN_ID`));

- CREATE TABLE `RECEIVED_EMAILS_ERRS` (`RECEIVED_EMAIL_ID` VARCHAR(32) NOT NULL, `CLIENT_ERROR` BLOB, `EBM_ERROR` BLOB, PRIMARY KEY (`RECEIVED_EMAIL_ID`));

- CREATE TABLE `EMAIL_MSG` (`EMAIL_MSG_ID` VARCHAR(32) NOT NULL, `EMAIL_FROM` VARCHAR(256) NOT NULL, `EMAIL_TO` VARCHAR(256) NOT NULL, `EMAIL_SUBJECT` VARCHAR(256) NOT NULL, `MSG` VARBINARY(4096), PRIMARY KEY (`EMAIL_MSG_ID`));

9.2. SQL заявки необходими за въвеждане на регулярни изрази свързани с различните кодове за грешка

ЕВМ разпознава различните кодове за грешка получени чрез имейл(и) вместо отговор, използвайки регулярни изрази. За целта е необходимо да се въведат тези изрази в базата данни преди пускането на ЕВМ, чрез следните SQL заявки:

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("4d8540b9e1634a03a8b0fd135488cce5", "421 Service not available, closing transmission channel", '421 (\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("d5d121044f2e400393c34eca70db8e61", "450 Requested mail action not taken: mailbox unavailable", '450 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("63cc3a1495114967867dcd257f5d31e8", "451 Requested action aborted: local error in processing", '451 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("c0731d3f10ca4ddcb31dfaab316b99f3", "452 Requested action not taken: insufficient system storage", '452 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("b6a973234236497eb614df708d0be193", "500 Syntax error, command unrecognized", '500 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("4e1fad7191d84ad6828792ca9464c3a8", "501 Syntax error in parameters or arguments", '501 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("0e2ec64470be49da8ac4e6fc43281f76", "502 Command not implemented", '502 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("ba1f04475ecf4077b50b3c021a3a95be", "503 Bad sequence of commands", '503 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("d7557918bcd34a87b9187329509de2cd", "504 Command parameter not implemented", '504 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("daf9dd0fecf347ddb260d926c772b05f", "550 Requested action not taken: mailbox unavailable", '550 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("2415ea0672784c6094ae429d3afb4fb3", "551 User not local", '551 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("ee96c1819a0f4a65b710677d47bbf43e", "552

Requested mail action aborted: exceeded storage allocation", '552 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("e78e9c4731f848e49be6c4da184c8500", "553 Requested action not taken: mailbox name not allowed", '553 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

- INSERT INTO REJECT_PATTERN (PATTERN_ID, DESCRIPTION, PATTERN) VALUES ("d75ffc87f5fe4a94962749bd0a1b1047", "554 Transaction failed", '554 (\\w+)(\\.\\w+)*@(\\w+\\.)(\\w+)(\\.\\w+)*');

9.3. SQL заявки необходими за успешната работа на EBM с неговият клиент

За да е възможно регистрирането на тестовия клиент за работа с EBM, е необходимо да се изпълни следната SQL заявка:

INSERT INTO CLIENT (CLIENT_ID, CLIENT_NAME, ADMIN_EMAIL) VALUES ("4B94CEE1F49311DAC01A503105C5448D", "EBMTestClient", "admin@127.0.0.1");

10. Списък на разгледаните фигури

Фиг. 1 Инфраструктура на ЕВМ и комуникацията между имейл кореспондентите 24
Фиг. 2 Комуникация между ЕВМ и негов клиент 26
Фиг. 3 Детайли от структурата на ЕВМ и негов клиент 27
Фиг. 4 ЕВМ API 28
Фиг. 5 ЕВМ API за клиенти 31
Фиг. 6 Регистрация на ЕВМ клиент 38
Фиг. 7 Отрегистриране на ЕВМ клиент 40
Фиг. 8 Изпращане на имейл 42
Фиг. 9 Получаване на имейл 46
Фиг. 10 Структура на базата данни 47
Фиг. 11 Стартиране на ЕВМ 58
Фиг. 12 Стартиране на тестовият клиент на ЕВМ 59
Фиг. 13 Потвърждение за регистрация 59
Фиг. 14 Изпращане на имейл 59
Фиг. 15 Потвърждение за изпратен имейл 60
Фиг. 16 Изпратен имейл в ЕВМ 60
Фиг. 17 Добавяне на настройки 61
Фиг. 18 Разглеждане на настройки 64
Фиг. 19 Потвърждение за изтриване 65
Фиг. 20 Списък с изпратените имейли 66
Фиг. 21 Списък с получените имейли 66