

«Допущен к защите»

Председатель цикловой комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.Б.Гусятинер

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2015 г.

**Разработка Почтового клиента**

**Пояснительная записка**

**ДП.230115.15.03ПЗ**

**Дипломный проект**

**Королёв, 2015**



**КОЛЛЕДЖ КОСМИЧЕСКОГО МАШИНОСТРОЕНИЯ И ТЕХНОЛОГИЙ**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | |  | «УТВЕРЖДАЮ»  Заместитель директора колледжа  по учебной работе  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.В.Васильева  \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2015 г. | |

**ЗАДАНИЕ**

**на дипломный проект**

студенту группы 46П **Валиеву Ивану Дмитриевичу**

Тема задания **«Разработка почтового клиента»**

Руководитель дипломного проекта **Шаповалов Е.П.**

**Содержание работы:**

Введение.

Глава 1. Теоретические основы разработки почтового клиента

Глава 2. Проектирование почтового клиента

Глава 3. Организационно - экономическая часть

Заключение.

Список использованных источников.

Приложения.

Презентация.

Срок выполнения дипломной работы 15 июня 2015 года.

Руководитель дипломной работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись студента

Дата выдачи задания «\_\_\_\_\_\_\_\_\_» апреля 2015 год

СОДЕРЖАНИЕ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

**ДП.230105.15.03ПЗ**

Дипломант

Валиев И.Д.

Консультант

Шаповалов Е.П.

Рецензент

Исаев В.Г.

Ст.консультант

Гусятинер Л.Б..

Конс.Экон.

Волкова Ю.А.

**«Разработка почтового клиента»**

Лит.

Листов

103

Группа к46П

[ВВЕДЕНИЕ 1](#_Toc422857828)

[1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ПОЧТОВОГО КЛИЕНТА 4](#_Toc422857829)

[1.1. Электронная почта 4](#_Toc422857830)

[1.1.1. Определение 4](#_Toc422857831)

[1.1.2. История возникновения и развития электронной почты 4](#_Toc422857832)

[1.1.3. Архитектура и службы 12](#_Toc422857833)

[1.1.4. Форматы сообщений 15](#_Toc422857834)

[1.1.5. Пересылка сообщений 25](#_Toc422857835)

[1.1.6. Окончательная доставка сообщений 30](#_Toc422857836)

[1.1.7. Веб-почта 40](#_Toc422857837)

[1.1.8. Пользовательские агенты (почтовые клиены) 41](#_Toc422857838)

[1.2. Анализ существующих популярных почтовых клиентов 45](#_Toc422857839)

[1.2.1. Общая информация 45](#_Toc422857840)

[1.2.2. Анализ Microsoft Outlook 2013 46](#_Toc422857841)

[1.2.3. Анализ TheBat! 48](#_Toc422857842)

[1.2.4. Выводы 56](#_Toc422857843)

[2. ПРОЕКТИРОВАНИЕ ПОЧТОВОГО КЛИЕНТА 57](#_Toc422857844)

[2.1.1. Общие сведения 57](#_Toc422857845)

[2.1.2. Возможности и особенности 60](#_Toc422857846)

[2.1.3. Преимущества перед веб-интерфейсом 63](#_Toc422857847)

[2.1.4. Описание структуры программного комплекса 64](#_Toc422857848)

[2.1.5. Описание процедур и функций 66](#_Toc422857849)

[2.1.6. Системные требования 69](#_Toc422857850)

[2.2. Обоснование выбора инструментов, сред и языков 71](#_Toc422857851)

[2.2.3. Задействованные инструменты, языки и технологии 71](#_Toc422857852)

[2.2.4. Описание и обоснование выбранных инструментов, языков и технологий 71](#_Toc422857853)

[3. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ 77](#_Toc422857854)

[3.1. Эксплуатационный раздел 77](#_Toc422857855)

[3.1.1. Сфера и специфика применения программного комплекса 77](#_Toc422857856)

[3.2. Раздел техники безопасности 78](#_Toc422857857)

[3.2.1. Техника безопасности при работе с ЭВМ 78](#_Toc422857858)

[3.3. Раздел экономических расчетов 83](#_Toc422857859)

[3.3.1. Расчет стоимости одного часа работы ПЭВМ 83](#_Toc422857860)

[3.3.2. Расчет затрат на составление и отладку программы 90](#_Toc422857861)

[3.3.3. Расчет показателей эффективности применения ПЭВМ 93](#_Toc422857862)

[3.3.4. Расчет затрат на разработку модуля 95](#_Toc422857863)

[3.3.5. Расчет Интернет-трафика в сравнении с веб-почтой 99](#_Toc422857864)

[3.3.6. Вывод 101](#_Toc422857865)

[ЗАКЛЮЧЕНИЕ 102](#_Toc422857866)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 103](#_Toc422857867)

[ПРИЛОЖЕНИЕ 1. Руководство пользователя 104](#_Toc422857868)

[ПРИЛОЖЕНИЕ 2. Листинги программы 118](#_Toc422857869)

# ВВЕДЕНИЕ

Почта древнее изобретение цивилизации, которое совершенствовалось на протяжении столетий. С появлением Интернета в конце 20 века связано возникновение такого замечательно сервиса, как электронная почта. Интернет обладает многими возможностями и функциями. При помощи Интернета любой человек способен воспользоваться одним из преимуществ Глобальной сети – это электронная почта.

Технология передачи почтового сообщения за многие десятилетия существенно упростилась. Вам уже нет надобности отсылать личного скорохода, выходить к почтовому дилижансу или бежать на почту, чтобы отправить письмо. Благодаря развитию Интернета люди из любой точки мира могут отправить свою почту и затем прочитать ответ. Расстояния и время больше не преграда. Воспользоваться услугами электронной почты в праве любой человек и этому не трудно научиться.

Сегодня переписка по электронной почте становится одной из основных форм общения между людьми, находящимися в разных городах и странах. Вы сможете читать письма и скачивать прикрепленные файлы, пришедшие из любой точки планеты, и отвечать на них, ни на минуту не отрываясь от своих дел, где бы вы ни находились. В наши дни во вложении могут находиться не только изображения, но и исполняемые файлы, видеофильмы, архивы больших объемов и многое другое.

Большинство людей уже используют преимущества электронной почты. Однако эффективность, удобство работы, совместимость и надежность различных почтовых программ сильно различаются. А за некоторые программы придется даже заплатить.

Целью данной работы является разработка почтового клиента или как по-другому еще называют пользовательского агента, удовлетворяющего основным потребностям пользователей электронной почты. Со временем сформировались определенные стандарты, которым я постараюсь следовать, а так же учесть распространенные ошибки и недочеты в современных почтовых программах. Для достижения цели потребуется ознакомиться с архитектурой электронной почтовой системы, протоколами передачи данных и их расширениями, проследить историю возникновения и развития электронной почты, а также произвести анализ уже существующих программных решений.

Работа раскрывается в следующих разделах:

* Теоретические основы разработки почтового клиента

В данном разделе представлена исключительно теория. Главные объекты, над которыми ведется работа – это электронная почта и программы для взаимодействия с ней. Поэтому здесь описываются теоретические понятия, такие как: история электронной почты, определения, основные понятия, протоколы передачи данных (SMTP, POP, IMAP), форматы Интернет-сообщений, архитектура почтой системы, анализ имеющихся программных решений и т.п.

* Проектирование почтового клиента

Данный раздел посвящен исключительно разрабатываемому программному комплексу, а именно – почтовому клиенту «MailV». Здесь написано описание программы, об ее возможностях, преимуществах и особенностях. Раздел содержит описание структуры программного комплекса, описание процедур и функций, обоснование выбора инструментов, сред и языков, описание системных требований к программе.

* Организационно-экономическая часть

Организационно-экономическая часть состоит из трех разделов:

* Эксплуатационный раздел

Содержит информацию о сфере и области применения почтового клиента.

* Раздел техники безопасности

Содержит правила при работе за ЭВМ.

* Расчет экономической эффективности применения данного программного комплекса

Содержит расчеты по формулам для определения экономической эффективности программного комплекса. А также расчет расхода Интернет-трафика в сравнении с веб-почтой.

# ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РАЗРАБОТКИ ПОЧТОВОГО КЛИЕНТА

## Электронная почта

### Определение

Электронная почта – это технология, которая предоставляет нам возможность отправлять и получать электронные сообщения.

Название «электронная почта» в Европе и Америке используют только в двух вариантах: «email» и реже «e-mail». С марта 2011 года англоязычным СМИ рекомендуется писать сокращение, как «email», а не «e-mail». В русском же языке присутствует вариативность, и нет четких законов по сокращению данного названия. Вот такие написания можно встретить на сегодняшний день в русских регионах, обратите внимание, что это названия абсолютно одной и той же технологии:

* Электронная почта, эл. почта, элп. почта
* Интернет-почта, веб-почта
* Имейл, мейл (транскрипция с английского)
* Е-мейл, емейл, емайл (различные варианты транслитерации)

Правильное написание пока не зафиксировано в словарях. Справочное бюро «Грамота.ру» указывает, что Е. Ваулина в словаре «Мой компьютер» предлагает писать «емайл» и «е-мэйл», но замечает, что такое написание не соответствует литературной норме русского языка, в то же время в других словарях советуют писать «e-mail» латиницей.

### История возникновения и развития электронной почты

Электронная почта (e-mail) существует уже более трех десятилетий. Она значительно старше, чем сети «ARPANET» или Интернет. Электронная почта никогда не была изобретена, она эволюционировала из очень простых начинаний. Она стала популярной с первых дней развития Интернета благодаря скорости и дешевизне. До 1990 года она использовалась преимущественно в научных организациях. В девяностые годы она получила широкую известность, и с тех пор количество отправляемых с помощью электронной почты писем стало расти экспоненциально. Среднее число сообщений, посылаемых ежедневно, скоро во много раз превзошло число писем, отправляемых с помощью обычной почты (snail mail), то есть бумажной.

Появление электронной почты можно отнести к 1965 году, когда сотрудниками Массачусетского технологического университета была написана первая почтовая программа «Mail» для операционной системы «CTSS (Compatible Time-Sharing System)».

Программа «Mail» - является первой и простейшей программой для unix-подобных систем, работающей в консольном режиме. Установлена она была на компьютер IBM 7090-7094. Авторы программы Ноэль Моррис и Том Ван Влек сделали возможным обмен сообщениями в пределах одного мейнфрейма (большого компьютера). Данная программа считается устаревшей, однако на просторах интернета можно найти руководство использования уже поистине исторической программы в операционной системе «Linux». Вот один из примеров работы с программой «Mail», который поможет ознакомиться с основными командами и ключами по ссылке: http://linux.about.com/od/commands/a/blcmdl8\_mailxx.htm

«Mail» имеет два режима работы:

* Интерфейсный, т.е. рисующий на экране диалог и ожидающий ввода данных пользователем
* Консольный, т.е. позволяющий использовать программу для автоматизированной отправки почты

Многие считают этот момент началом электронной почты. Но, история современной электронной почты, которой мы пользуемся сейчас, берет свое начало на шесть лет позже.

В конце 1971 года программист Рэй Томлинсон, сотрудник компании «Bolt Beranek and Newman, Inc.» написал программу, которая позволила посылать сообщения на удаленный компьютер по протоколу «CypNet», использующийся ранее для передачи файлов. Полученные данные помещались в файл – «Почтовый ящик». Так и появилась электронная почта, которую мы все сейчас знаем. Рэй Томлинсон так вспоминал о первых электронных письмах: “Я посылал сам себе кучу тестовых сообщений, бегая от компьютера к компьютеру. Что это были за сообщения теперь и не вспомнить.… Вполне вероятно, что самое первое сообщение было QWERTYUIOP (подряд идущие буквы на англоязычной клавиатуре – И.К.) или что-то подобное”.

Практически сразу программа была им усовершенствована, адаптировав ее для использования в сети «ARPANet», предшественнице нынешней сети Интернет. Томлинсон разработал систему организации почтовых адресов: каждому пользователю компьютера присваивался адрес, состоящий из имени пользователя и сетевого имени его компьютера, которые разделялись знаком @ (в просторечии - собака). А дело все в том, что на телетайпном аппарате “модель 33”, который был в распоряжении Рэя Томлинсон, эта клавиша использовалась для пунктуации и обозначения английского предлога at (на). Таким образом, выражение user@machine означало user at machine, т.е. такой-то пользователь на таком-то компьютере. После Томлинсон продолжил совершенствовать программу и в марте 1972 года сделал простой пользовательский интерфейс, позволяющий отправлять и загружать сообщения, используя сеть. Через полгода Лоуренс Робертс, коллега Томлинсон, на основе его программы разработал свою, которая имела не мало сервисных функций.

В июле 1972 года – Ларри Робертс (Larry Roberts) написал первую программу, которая облегчила пользователям работу с электронной почтой. Она позволяла создавать и сортировать списки писем, пользователь мог выбирать и читать требуемое сообщение, сохранять послание в файле, а также пересылать электронные письма на другой адрес или автоматически отвечать на полученное послание. Строго говоря, это была первая программа почтовый клиент, позволившая даже неспециалистам легко управляться с электронной почтой. Как известно, человек ленив, и поэтому за короткий срок программа завоевала громадную популярность среди нарождавшейся сетевой общественности.

В 1973 году завоевав популярность в ARPANET, электронная почта служила для передачи 75% всех данных. А в 1975 году появилась первая почтовая рассылка писем с полезной информацией. Ее основателем стал Стив Уолкер (Steve Walker). Вскоре модератором, то есть по сути дела редактором и человеком, ответственным за выход рассылки, стал Эйнар Стефферуд (Einar Stefferud). Первая почтовая рассылка не была автоматизирована, и все делалось вручную. Самой популярной рассылкой были новости мира научной фантастики.

В этом же году программист Джон Виттал написал программу MSG, которая включала все известные функции для работы с почтой. С этого момента электронная почта вышла на государственный уровень.

В 1976 году, 26 марта. Английская королева Елизавета II впервые отправила письмо по электронной почте, воспользовавшись услугами Королевской службы сигнализации и радиолокации (Royal Signals and Radar Establishment – RSRE) в Малверне (Malvern).

В 1977 году Ларри Лэндвебер (L. Landweber) из Висконсинского университета разработал программу THEORYNET, которая обеспечивала пересылку электронной почты в сети, объединявшей около 100 специалистов по вычислительной технике. Система была предназначена для работы в TELENET.

1979 год, 12 апреля – день рождения смайлика. Именно в этот день Кевин Маккензи (Kevin MacKenzie), один из апологетов «эмоциональных» компьютеров, обратился в «Message Services Group» (MsgGroup) с письмом, в котором предлагалось включать в «сухие» компьютерные тексты некоторые символы, обозначающие эмоции. Например, сочетание :-). Несмотря на завязавшуюся дискуссию среди приверженцев и противников нововведения, «эмоциональные смайлики» вскоре стали очень популярны среди пользователей. Нынешние обитатели Сети, общающиеся как посредством электронной почты, так и в режиме реального времени, вряд ли мыслят свою жизнь без столь привычной вещи, как «смайлики». Существует, например, специальный «The smileys! Server» (<http://www.pop.at/smileys/>), на котором можно ознакомиться со способами выражения при помощи точек и прочих символов самых разнообразных эмоций. И даже посмотреть на анимированные при помощи JAVA «смайлики».

В 1981 году  для того чтобы удовлетворить потребности американских университетов в качественных сетевых сервисах и, в первую очередь, в электронной почте, была создана сеть CSNET (Computer Science Network), более известная теперь под названием Computer and Science Network. Новая сеть создавалась совместными усилиями ученых из нескольких американских университетов (Университет штата Делавэр, Университет штата Висконсин, других университетов), а также компаний «Bolt Beranek and Newman, Inc.» (BBN) и «RAND Corporation». Для этой цели организацией «National Science Foundation» был даже выделен специальный грант, на который собственно и была создана сеть, обеспечившая американских студентов и ученых электронной почтой с выходом в ARPANET.

В 1982 году специалистами EUUG (European UNIX Users Group) создана сеть EUNet (European UNIX Network) специально для того, чтобы обеспечить пользователям UNIX-систем доступ к электронной почте и возможность пользования услугами USENET.

В 1987 году, 20 сентября на основе протокола CSNET установлена электронная почтовая связь между Германией и Китаем. Именно в этот день из Китая в Германию было отправлено первое электронное письмо.

В 1988 году пользователи сети «FidoNet» получили возможность обмениваться электронными письмами с корреспондентами, подключенными к другим сетям.

В 1989 году впервые была установлена связь между коммерческими почтовыми службами и Интернетом. Служба «MCI Mail» была подключена через CNRI (Corporation for National Research Initiative), а «Compuserve» через Университет штата Огайо.

В 1994 году электронная почта впервые была использована для рассылки рекламных объявлений. Позднее такие материалы, «засоряющие» почтовые ящики пользователей ненужной информацией, получили название «спам». Пионером в распространении спама стала адвокатская контора из Аризоны «Canter & Siegel». Первоначально это была реклама лотереи и грин-кард.

1997 году в каталоге электронных почтовых рассылок Liszt ([http://liszt.com](http://liszt.com/)) было зарегистрировано почти 72 тысячи рассылок на самые различные темы.

В 1998 году компания Casio PhoneMate представила на Consumer Electronics Show (CES) автоответчик IT-380 E-Mail Link. По утверждениям компании, это первое устройство такого типа, предоставившее пользователям возможность просматривать полученные сообщения электронной почты.

1998 году, в декабре инженер-программист Лин Хай (Lin Hai) и физик Уонг Юкай (Wang Youcai) осуждены за распространение «виртуальных» листовок по электронной почте среди более 250 тыс. других китайских диссидентов и зарубежных сторонников демократических реформ в стране. Им было предъявлено обвинение в подстрекательстве к свержению государственного строя.

2000 году ранним утром 4 мая по сетям электронной почты начал свое путешествие вирус «I Love You». Он оказался родом с Филиппин и был распространен через четыре взломанных адреса электронной почты. По оценкам экспертов этот вирус причинил ущерб на сумму не менее 7 млрд. долларов, а в США, например, каждая 15-я компания, так или иначе, пострадала от этого вируса. Как ни странно, но столь массовое «размножение» вируса еще раз подтвердило тот факт, что электронная почта стала привычным сетевым инструментом и отнюдь не собирается сдавать своих позиций даже с появлением новых средств сетевого общения. [4]

Хронология возникновений популярных почтовых сервисов:

* 1996 год, 4 июля (день Независимости США) – начало коммерческого функционирования почтового сервиса «Hotmail»
* 1997 год, 8 марта - компания Yahoo! приобретает портал «RocketMail» — один из первых бесплатных почтовых сервисов. Появление сервиса Yahoo! Mail
* 1998 год, 15 октября — заработала бесплатная электронная почта от Mail.Ru
* 2000 год, 26 июня — запущена Яндекс. Почта — бесплатный почтовый сервис от компании Яндекс
* 2004 год, 1 апреля — запущен бесплатный почтовый сервис Gmail от компании Google

Сегодня значение адреса электронной почты встало на один уровень с адресом места жительства или номером мобильного телефона. Появилось множество почтовых сервисов, а вторая часть выражения user@machine интерпретируется иначе. Теперь вместо machine – конкретного компьютера, на котором работало несколько пользователей, подразумевается сервер, предоставляющий услуги электронной почты. Такой сервер обслуживает сотни миллионов пользователей. Так же электронная почта получила широкое применение в промышленности для обмена информацией внутри компаний, что делает совместную работу над сложными проектами далеко удаленных друг от друга сотрудников. В последнее десятилетие широко распространились и другие формы сетевой коммуникации, такие как обмен, мгновенными сообщениями и IP-телефония, но рабочей лошадкой интернет-коммуникации остается электронная почта. К сожалению, так же как и в отношении обычной почты, 9 из 10 сообщений электронной представляют из себя спам (spam) (McAfee, 2010). [1. Стр. 661]

Электронной почте, как и любой форме коммуникаций, присущ определенный стиль и набор соглашений. В частности, общение по электронной почте носит очень неформальный и демократичный характер. Скажем, человек, который никогда бы не осмелился позвонить или даже написать бумажное письмо какой-нибудь Особо Важной Персоне (VIP), запросто может сесть и написать ей небрежное электронное сообщение. Часто переписка по e-mail, фокусируется на содержании, а не на статусе и устраняет большинство проблем, связанных с различиями в должностном положении, возрасте и поле. Благодаря электронной почте блестящая идея, посланная студентом-практикантом по электронной почте, имеет шанс выиграть у идеи не столь блестящей, но высказанной вице-президентом компании.

На сегодняшний день люди в электронной почте обожают использовать особый жаргон и сокращения, такие как BTW (By The Way – между прочим), ROTFL (Rolling On The Floor Laughing – катаюсь по полу от смеха), IMHO (In My Humble Opinion – по моему скромному мнению) и т.д. Кроме того, чрезвычайно популярны смайлы (smileys). Этот и другие эмотиконы помогают передать тон общения. Они заполонили не только переписку по электронной почте, но и обмен мгновенными сообщениями.

За время использования смайликов изменились и протоколы электронной почты. Первые системы электронной почты состояли просто из протоколов передачи файлов и договоренности указывать адрес получателя в первой строке каждого сообщения (то есть файла). Со временем электронная почта отошла от передачи файлов, и были добавлены многие опции, такие как возможность, отослать одно сообщение нескольким адресатам. В 90-ых годах добавилась возможность передавать мультимедиа, то есть посылать сообщения с изображением и другой не текстовой информацией. Программы для чтения почты стали более сложными, переход от основанного на тексте к графическому пользовательскому интерфейсу обеспечил пользователям возможность доступа к почте при помощи ноутбуков и другой мобильной техники, вне зависимости от того, где они находились территориально. Наконец, обилие спама заставило современные программы для чтения почты и протоколы передачи оной информации обратить внимание на поиск и удаление нежелательных сообщений.

Вот такой по истине длинный путь проделала электронная почта с момента создания до наших дней. И если раньше в начале своего пути она использовалась исключительно военными и учеными, то сегодня пользуется популярностью у сотни миллионов человек по всему миру. И понятно почему: из-за простоты, скорости и малой вероятности, что адресат не получит электронное письмо.

### Архитектура и службы

В данном разделе рассматриваются возможности и организация систем электронной почты. Система электронной почты состоит из двух подсистем: пользовательских агентов или почтовых клиентов, позволяющих пользователям читать и отправлять электронную почту, и агентов передачи сообщений, пересылающих сообщения от отправителя к получателю.

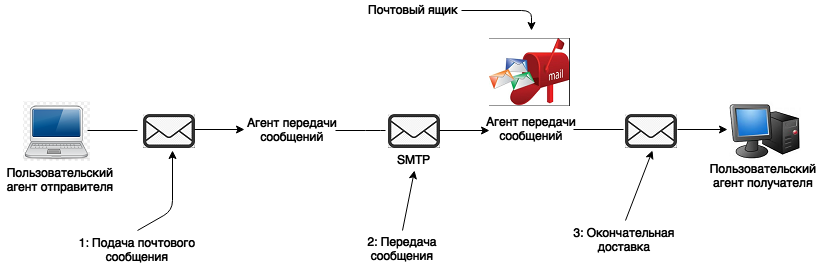


Рис. 1.1. Архитектура системы e-mail

Пользовательские агенты (почтовые клиенты) представляют собой программы, предоставляющие графический интерфейс или иногда интерфейс, базирующийся на тексте или командах, позволяющий пользователям взаимодействовать с системой электронной почты. В него входят средства написания сообщений и ответов на сообщения, отображения входящих сообщений и организации писем при помощи распределения их по папкам, поиска и удаления. Отсылка новых сообщений в почтовую систему для их дальнейшей доставки называется подачей почтового сообщения.

Обработка сообщений может быть частично автоматизирована с учетом желаний пользователя. Например, поступающая почта может фильтроваться, чтобы извлечь или приписать низкий приоритет сообщениям, похожим на спам. Некоторые программы включают дополнительные возможности, такие как автоматическая отправка отчетных сообщений («Я в отпуске, скоро вернусь и отвечу на ваше письмо»). Пользовательский агент работает на том же компьютере, на котором пользователь читает свою электронную почту. Это обычная программа, и она не обязательно должна работать все время.

Агенты передачи сообщений, как правило, являются системными процессами. Они работают в фоновом режиме на машинах почтовых серверов и всегда должны быть доступными. Они должны автоматически перемещать почтовые сообщения по системе от отправителя получателю при помощи SMTP (Simple Mail Transfer Protocol – простого протокола передачи почтовых сообщений). Это шаг, на котором передается сообщение.

SMTP был впервые определен как RFC 821. Далее в него вносились изменения вплоть до текущей редакции RFC 5321. Он отсылает сообщения по соединениями высылает обратно отчеты о статусу доставки и любых возникших ошибках. Существует множество приложений, в которых подтверждение доставки имеет большую важность и даже может иметь юридическую значимость («Ваша честь, моя электронная система не очень надежна, поэтому я полагаю, что повестка с вызовом в суд просто где-то затерялась»).

Агенты передачи сообщений также используют списки рассылки, которые позволяют доставлять идентичные копии сообщения всем, чьи адреса были включены в список адресов электронной почты. Среди других полезных дополнительных функций можно перечислить следующее: рассылка копий писем «под копирку» (Carbon copy или Cc), рассылка копий без уведомления о других получателях (Blind carbon copy или Bcc), письма с высоким приоритетом, секретная (то есть зашифрованная) почта, возможность доставки письма альтернативному получателю, если основной временно недоступен.

За связь пользовательских агентов и агентов передачи сообщений отвечают почтовые ящики и стандартный формат почтовых сообщений. Почтовые ящики хранят почту, которая доставляется пользователю. Они поддерживаются почтовыми серверами. Пользовательские агенты просто предоставляют возможность увидеть содержимое их почтовых ящиков. Чтобы это сделать, пользовательский агент отсылает почтовым серверам команды и получает возможность манипулировать почтовыми ящиками, проверяя их содержимое, удаляя сообщения и т.д. Последний шаг в извлечении почты – это ее доставка конечному пользователю. При такой архитектуре один пользователь может использовать различные пользовательские агенты на различных машинах, чтобы получить доступ к одному и тому же почтовому ящику.

Почта пересылается между агентами передачи сообщений в стандартном формате. В первоначально разработанный формат, RFC 822, вносились изменения. Текущая версия носит название RFC 5322; в нее включена поддержка мультимедиа-контента и международный текст. Данная схема называется MIME.

В основе всех современных систем электронной почты лежит ключевая идея о разграничении конверта и содержимого письма. Конверт заключает в себе сообщение. Он содержит всю информацию, необходимую для доставки письма, - адрес получателя, приоритет, уровень секретности и т.п. Все эти сведения отделены от самого сообщения. Агенты передачи сообщений используют конверт для маршрутизации, аналогично тому, как это делает обычная почтовая служба.

Сообщение внутри конверта состоит из двух отдельных частей: заголовка и тела письма. Заголовок содержит управляющую информацию для пользовательских агентов (почтовых клиентов). Тело письма целиком предназначена для человека – получателя. [1. Стр. 661-664]

### Форматы сообщений

**RFC 5322 – формат интернет-сообщений.** RFC 5322 – это основной ASCII-формат электронного письма, который является последним вариантом оригинального варианта интернет-сообщений, описанного в RFC 822.

Основные поля заголовка, связанные с транспортировкой сообщения, перечислены в таблице 1.1.

Поле *To:* содержит DNS – адрес основного получателя. Возможно наличие и нескольких получателей. В поле *Cc:* указываются адреса дополнительных получателей. С точки зрения доставки, никакой разницы между основным и дополнительными получателями нет. Разница между ними чисто психологическая и, может быть, важна для людей, но совершенно не существует для почтовой системы. Поле *Bcc:* аналогична предыдущему, с той разницей, что в последнем случае строка этого поля удаляется из всех экземпляров сообщения, отправленных как основному, так и дополнительным получателям. Это свойство позволяет рассылать одно письмо одновременно нескольким получателям так, что получатели не будут знать, что это письмо послано еще кому-либо кроме них.

Таблица 1.1. Поля заголовка стандарта RFC 5322, связанные с транспортировкой сообщения

|  |  |
| --- | --- |
| Поле | Значение |
| To: | Электронный адрес (адреса) основного получателя (получателей) |
| Cc: | Электронный адрес (адреса) дополнительного получателя (получателей) |
| Bcc: | Электронный адрес (адреса) слепой копии |
| From: | Автор (автор) сообщения |
| Sender: | Электронный адрес отправителя |
| Received: | Строка, добавляемая каждым агентом передачи сообщений на протяжении маршрута |
| Return-path: | Может быть использован для идентификатора обратного пути к отправителю |

Следующие два поля, *From:* и *Sender:*, сообщают соответственно, кто составил и отправил сообщение. Это могут быть разные люди. Например, написать письмо может руководитель предприятия, а отослать – его секретарь. В этом случае руководитель будет числиться в поле From:, а секретарь – в поле *Sender:*. Поле *From:* является обязательным, тогда как поле Sender: может быть опущено, если его содержимое не отличается от содержимого поля *From:*. Эти поля нужны на случай, если сообщение доставить невозможно и об этом следует проинформировать отправителя. Кроме того, по адресам, указанным в этих полях, может быть отправлен ответ.

Строка, содержащая поле *Received:*, добавляется каждым агентом передачи сообщений на пути следования сообщения. В это поле помещается идентификатор агента, дата и время получения сообщения, а также другая информация, которая может быть использована для исправления неисправностей в системе маршрутизации.

Поле *Return-Path:* добавляется последним агентом передачи сообщений. Предполагалось, что это поле будет сообщать, как добраться до отправителя. Теоретически, эта информация может быть собрана из всех полей Received: заголовка (кроме имени почтового ящика отправителя), однако на практике оно редко заполняется подобным образом и обычно просто содержит адрес отправителя.

Помимо полей, показанных в таблице 1.1 , сообщения стандарта RFC 5322 могут также содержать широкий спектр полей заголовка, используемых пользовательским агентом или самим пользователем. Наиболее часто используемые поля заголовка приведены в таблице 1.2.

Таблица 1.2. Некоторые поля, используемые в заголовке сообщения стандарта RFC 5322

|  |  |
| --- | --- |
| Поле | Значение |
| Date: | Дата и время отправки сообщения |
| Reply-to: | Электронный адрес, на который следует присылать ответ |
| Message-Id: | Уникальный номер для последующей ссылки на это сообщение |
| In-Reply-To: | Идентификатор Message-Id сообщения, в ответ на которое посылается это сообщение |
| References: | Другие важные ссылки (идентификаторы Message-Id) |
| Keywords: | Ключевые слова, выбираемые пользователем |
| Subject: | Краткое изложение темы сообщения для отображения в одной строке |

Поле *Reply-to:* иногда используется, в случае если ни составитель письма, ни его отправитель не хотят получать на него ответ. Например, управляющий отделом сбыта может написать письмо, информирующее клиентов о новом продукте. Это письмо отправляется его секретарем, но в поле *Reply-to:* указан адрес менеджера отдела продаж, который сможет ответить на вопросы и принять заказы. Это поле также может быть полезным, если у отправителя есть два адреса электронной почты, и он хочет, чтобы ответы приходили не на тот, с которого он отправляет сообщение.

*Message-Id:* - это автоматически генерируемое число, которое используется, чтобы связывать сообщения (например, при ответе на письмо) и избежать повторной доставки.

В документе RFC 5322 открыто сказано, что пользователям разрешается изобретать дополнительные заголовки для своих нужд. Начиная с формата RFC 822, такие заголовки начинаются со строки X-. Гарантируется, что в будущем никакие стандартные заголовки не будут начинаться с этих символов, чтобы избежать конфликтов между официальными и частными заголовками.

После заголовка идет тело самого сообщения. Пользователь может разместить в нем все, что угодно.

**MIME – многоцелевые расширения электронной почты в сети Интернет.** На заре существования сети ARPANet электронная почта состояла исключительно из текстовых сообщений, написанных на английском языке и представленных символами ASCII. Для подобного использования первоначально стандарта RFC 822 было вполне достаточно: он определял формат заголовков, он оставлял содержимое сообщения полностью на усмотрение пользователя. В 1990-е годы всемирное использование Интернета и необходимость посылать более разнообразный контент через почтовую систему показали, что такой подход уже не удовлетворяет пользователей, привыкших к Интернету. Требовалось обеспечить возможность отправление сообщений на разных языках с надстрочными знаками (например, на французском и немецком), на языках, использующих алфавиты, отличные от латинского (например, на иврите или русском), или на языках без алфавитов (например, китайском и японском). Также требовалась возможность отсылки сообщений, не являющихся текстом (например, аудио, изображения или бинарные документы и программы).

Решением стала разработка MIME (Multipurpose Internet Mail Extensions – многоцелевые расширения электронной почты в Интернете). Они широко применяются как для почтовых сообщений, посылаемых через Интернет, так и для описания контента в других приложениях, таких как веб-сайты. MIME описаны в RFC 2045-2047, 4288, 4289 и 4049.

Основная идея стандартов MIME – продолжить использование формата RFC 822 (предшественника формата RFC 5322, действовавшего в то время, когда были предложены MIME), но с добавлением структуры к телу сообщения и с определением правил кодировки для передачи не ASCII-сообщений. Не отклоняясь от стандарта RFC 822, MIME-сообщения могут передаваться с помощью обычных агентов передачи сообщений и протоколов (ранее основанных на RFC 821, а сейчас на RFC 5321). Все, что нужно было изменить, - это отправляющие и принимающие программы, которые пользователи могли создать для себя сами.

Стандартами MIME определяются пять новых заголовков сообщения, приведенных в таблице 1.3. Первый заголовок (*MIME-Version:*) просто информирует пользовательского агента, получающего сообщение, что тот имеет дело с сообщением MIME, а также сообщает ему номер версии MIME, используемой в этом сообщении. Если сообщение не содержит такого заголовка, то оно считается написанным на английском языке (или, по крайней мере, на языке, использующем только знаки ASCII) и обрабатывается соответствующим образом.

Таблица 1.3. Заголовки сообщений, добавленные MIME

|  |  |
| --- | --- |
| Заголовок | Описание |
| MIME-Version: | Указывает версию стандартов MIME |
| Content-Description: | Описание содержимого. Строка обычного текста, информирующая о содержимом сообщения |
| Content-Id: | Уникальный идентификатор |
| Content-Transfer-Encoding: | Указывает способ кодировки тела сообщения для его передачи |
| Content-Type: | Тип и формат содержимого сообщения |

Заголовок *Content-Description:* представляет собой ASCII-строку, информирующую о том, что находится в сообщении. Этот заголовок позволяет пользователю принять решение о том, нужно ли ему декодировать и читать сообщение. Если в строке сказано: «Фотография студента с Мадрида », а получивший это сообщение не является любителем этого города, то, вероятнее всего, он сразу удалит это сообщение, а не станет перекодировать его в цветную фотографию высокого разрешения.

Заголовок *Content-Id:* содержит идентификатор содержимого сообщения. В нем используется тот же формат, что и в стандартном заголовке *Message-Id:*.

Заголовок *Content-Transfer-Encoding:* сообщает о способе упаковки тела сообщения для его передачи по сети. Во времена разработки MIME основной проблемой были протоколы передачи сообщений (SMTP), предполагавшие, что сообщение будет в формате ASCII, при этом в строке должно быть не больше 1000 символов. Символы ASCII используют 7 бит из каждого восьмибитного байта. Бинарные данные, такие как исполняемые программы и изображения, используют все 8 бит байта, так же как и расширенные наборы символов. Не было никакой гарантии того, что эти данные будут передаваться безопасно. В результате этого потребовался метод передачи бинарных данных, который заставлял бы их выглядеть как обычное почтовое сообщение ASCII. Расширения SMTP, введенные с момента разработки MIME, на самом деле позволяют передавать восьмибитные бинарные данные, хотя даже сегодня незакодированные бинарные данные не всегда корректно передаются почтовой системой.

MIME обеспечивается пять схем кодирования, предназначенных для передачи данных (кроме того, имеется возможность добавления новых схем, на всякий случай).Простейшая из них заключается в передаче просто текстовых сообщений ASCII. Символы ASCII используют 7 разрядов и могут передаваться напрямую протоколом электронной почты, при условии, что строка не превышает 1000 символов.

Следующая по простоте схема аналогична предыдущей, но использует 8-разрядные символы, то есть все значения байта от 0 до 255 включительно. Сообщения, использующие 8-разрядную кодировку, также должны соблюдать правила о максимальной длине строки.

Еще хуже обстоит дело с сообщениями в настоящей двоичной кодировке. К ним относятся произвольные двоичные файлы, которые не только используют все 8 разрядов в байте, но еще и не соблюдают ограничения на 1000 символов в строке. К этой категории относятся исполняемые программные файлы. Сегодня почтовые серверы могут проверять, есть ли возможность переслать данные в бинарной (или восьмибитной) кодировке, и если на обоих концах не поддерживается данное расширение, данные будут передаваться в ASCII.

Кодировка в бинарных данных в формате ASCII называется base64 (64 символьная кодировка). При использовании данного метода группы по 24 бита разбиваются на четыре единицы по 6 бит, каждая из которых посылается в виде обычного разрешенного ASCII-символа. В этой кодировке 6-разрядный символ 0 кодируется ASCII символом «А», 1 – ASCII-символом «В» и т.д. Затем следуют 26 строчных букв, затем 10 цифр и, наконец, + и / для кодирования 62 и 63 соответственно. Последовательности == и = говорят о том, что последняя группа содержит только 8 или 16 бит. Символы перевода строки и возврата каретки игнорируются, поэтому их можно вставлять в любом месте закодированного потока символов для того, чтобы строки не выглядели слишком длинными. Таким способом можно передать любой двоичный код, хотя и достаточно не эффективно. Эта кодировка была крайне популярна до того, как приобрели широкое распространение почтовые серверы, способные передавать бинарную информацию.

Для сообщений, почти полностью состоящих из символов ASCII, но с небольшими включениями не ASCII-символов, подобный метод несколько неэффективен. Вместо него лучше применять кодировку quoted-printable (цитируемое печатное кодирование). Это просто 7-битный ASCII, в котором символы, соответствующие значениям ASCII-кода свыше 127, кодируются знаком равенства, за которым следуют две шестнадцатеричные цифры – ASCII-код символа. Кроме конечных пробелов также кодируются символы управления, некоторые пунктуационные знаки и математические символы.

Наконец, когда имеются веские причины не использовать эти методы можно указать в заголовке *Content-Transfer-Encoding:* свою кодировку.

Заголовок *Content-Type:* указывает тип тела сообщения и его применение выходит далеко за пределы электронной почты. Например, контент, загружаемый из Интернета, помечается символами MIME, что позволяет браузеру адекватно отображать информацию. Так обстоит дело с контентом, пересылаемым через потоковое мультимедиа, и передачей в реальном времени, например голоса через IP (VOIP).

Изначально в документе RFC 1521 были определены семь типов содержимого сообщений, каждый из которых распадается на несколько доступных подтипов. Подтип отделяется от типа косой чертой, например «Content-Type: video/mpeg». С тех пор к ним было добавлено много типов и подтипов. В сети его поддерживается IANA, он доступен по адресу www.iana.org/assignments/media-types.

Типы и примеры часто используемых подтипов приведены в таблице 1.4.

Тип *text*. Комбинация *text/plain* служит для обозначения обычного текстового сообщения, которое может быть отображено на экране компьютера сразу после получения. Для этого не требуется дополнительной обработки. Когда веб-технологии стали популярны, был добавлен новый тип *text/html* (в RFC 2854), который позволил пересылать веб-страницы в теле письма RFC 822. В RFC 3023 определен подтип для расширяемого языка разметки (eXtensible Markup Language - XML), *text/xml*.

Следующим типом MIME является *image*. Он используется для передачи неподвижных изображений. На сегодняшний день существует множество различных форматов хранения и передачи изображений, как с использованием сжатия, так и без него. Некоторые форматы, в том числе GIF, JPEG, TIFF, встроены практически во все браузеры, однако существует еще множество других типов.

Типы *audio* и *video* предназначены для передачи звука и двигающегося изображения. Но подтип *video* включает в себя только визуальную информацию, а не звуковую дорожку. Если необходимо передать по электронной почте видеофильм со звуком, то, возможно придется посылать видеоряд и звуковую дорожку отдельно друг от друга. Это зависит от системы кодирования. Первым видеоформатом, который был определен стандартом MIME, стал формат MPEG (Motion Pictures Experts Group – экспертная группа по вопросам движущегося изображения). Так, чтобы позволить пользователям пересылать аудио файлы в формате MP3, в RFC 3003 помимо уже существующего *audio/basic* был добавлен тип *audio/mpeg*. Тип *video/mp4* и *audio/mp4* указывают на видео- и аудио- данные в новом формате MPEG4.

Таблица 1.4. Типы стандарта MIME и примеры подтипов

|  |  |  |
| --- | --- | --- |
| Тип | Подтип | Описание |
| text | plain, html, xml, css | Текст в различных форматах |
| image | gif, jpeg, tiff | Изображение |
| audio | basic, mpeg, mp4 | Звуки |
| video | mpeg, mp4, quicktime | Видеофайлы |
| model | vrml | 3D-модель |
| application | onoctet-stream, pdf, javascript, zip | Данные, производимые приложениями |
| message | http, rfc822 | Инкапсулированное сообщение |
| multipart | mixed, alternative, parallel, digest | Комбинация нескольких типов |

Тип *model* был добавлен позднее, чем остальные типы контента. Он предназначен для обозначения 3D – моделей.

Тип *application* предназначен для всех форматов, которые не покрываются остальными типами и требуют от приложения интерпретации данных. Пользовательские агенты, получившие этот контент, используют для отображения стороннюю библиотеку или внешнюю программу.

Тип *octet-stream* (байтовый поток) представляет собой просто последовательность никак не обрабатываемых байтов. Получив такой поток, пользовательский агент должен, вероятно, предложить пользователю сохранить его в виде файла. Последующая обработка файла целиком зависит от пользователя, который, предположительно, знает, с каким контентом он столкнулся.

Тип *message* позволяет помещать одно сообщение в другое. Это может быть полезно для переадресации письма. Если внутри одного сообщения заключено целое сообщение стандарта RFC 822, следует использовать подтип rfc822. Сходным образом обычно инкапсулируются документы в формате HTML. А подтип *partial* позволяет разбивать сообщение на части и передавать их отдельно. Параметры обеспечивают восстановление сообщения получателем из отдельных фрагментов в правильном порядке.

Тип *multipart* позволяет составлять сообщение из нескольких частей, при этом начало и конец каждой части отчетливо указывается. Подтип *mixed* позволяет создавать сообщение из частей различных форматов. Многие почтовые программы позволяют пользователю прикрепить к текстовому сообщение одно или более приложений. Эти приложения пересылаются с использованием типа *multipart*.

Подтип *alternative* используется для сообщений, посылаемых одновременно из разных языках.

Подтип *parallel* используется, когда все части должны «просматриваться» одновременно. Например, в фильмах часто есть видео-и аудиодорожки.

Подтип *digest* используется, когда несколько сообщений объединяются в одно составное. Например, какая-нибудь куссионная группа в Интернете может собирать сообщения от подписчиков, а затем выслать их в виде единого сообщения типа *multipart/digest*. [1. Стр 672-676]

### Пересылка сообщений

Как агенты передачи сообщений доставляют письма от отправителя получателю? Передача почты осуществляется посредством протокола SMTP. Для этого проще всего установить транспортное соединение от машины-источника до машины-приемника, а затем просто передать по нему сообщение. Так изначально работал SMTP. Однако за последние годы возникло два разных способа использования SMTP. Первый – для подачи почтового сообщения (первый шаг в архитектуре e-mail на рис 1.1). Этим способом пользовательские агенты переправляют сообщения в почтовую систему для их дальнейшей отправки. Второй вариант использования – передача сообщений между агентами передачи сообщений (шаг №2 на рис. 1.1)

Такая последовательность обеспечивает доставку почты на всем пути от посылающего до получающего агента передачи сообщений за один шаг. Окончательная доставка происходит при участии других протоколов.

**SMTP – простой протокол электронной почты и его расширения.** В Интернете для доставки электронной почты машина – источник устанавливает TCP-соединение с портом 25 машины-приемника. Этот порт прослушивается почтовым демоном и их общение происходит с помощью протокола SMTP (Simple Mail Transfer Protocol). Этот сервер принимает входящие соединения, проверяет безопасность и принимает сообщения для доставки. Если письмо невозможно доставить, отправителю возвращается сообщение об ошибке, содержащее первую часть этого письма.

Протокол SMTP представляет собой простой ASCII – протокол. Это не недостаток, а отличительная черта. Использование текста в формате ASCII позволяет легко модифицировать, тестировать и исправлять ошибки в протоколах. Они могут тестироваться отсылкой команд вручную, при этом записи сообщений легко читать. Сейчас так работает большинство интернет-протоколов прикладного уровня (например, HTTP).

Установив TCP-соединение с портом 25, передающая машина, выступающая в роли клиента, ждет запроса принимающей машины, работающей в режиме сервера. Сервер начинает диалог с того, что посылает текстовую строку, содержащую его идентификатор и сообщающего о его готовности (или неготовности) к приему почты. Если сервер не готов, клиент разрывает соединение и повторяет попытку позднее.

Если сервер готов принимать почту, клиент объявляет, от кого поступила почты и кому она предназначена. Если получатель почты существует, сервер дает клиенту добро на пересылку сообщения. Затем клиент посылает сообщение, а сервер подтверждает его получение. Контрольные суммы не проверяются, так как транспортный протокол TCP обеспечивает надежный байтовый поток. Если у отправителя есть еще почта, она также отправляется. После передачи всей почты в обоих направлениях соединение разрывается.

Рассмотрим пример диалога клиента и сервера на листинге 1.1. Сначала клиент посылает приветствие серверу. Таким образом, первая команда клиента выглядит как HELO, что представляет собой сокращение слова HELLO до четырех символов. Зачем все эти команды надо было сокращать до четырех букв, сейчас уже никто не помнит.

Листинг 1.1. Передача сообщения от alice@cs.washington.edu для bob@ee.uwa.edu.au

|  |
| --- |
| S: 220 ee.uwa.edu.au служба SMTP готова  C: HELO abcd.com  S: 250 cs.washington.edu приветствует ee.uwa.edu.au  C: MAIL FROM: <alice@cs.washington.edu>  S: 250 подтверждаю отправителя  C: RCPT TO: <bob@ee.uwa.edu.au>  S: 250 подтверждаю получателя  C: DATA  S: 354 Отправляйте письмо; конец письма обозначается строкой, состоящей из символа "."  C: From: alice@cs.washington.edu  C: To: bob@ee.uwa.edu.au  C: MIME-Version: 1.0  C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au>  C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm  C: Subject: Земля обошла вокруг Солнца целое число раз  C:  C: Это преамбула. Пользовательский агент игнорирует ее. Ку-ку  C:  C: --qwertyuiopasdfghjklzxcvbnm  C: Content-Type: text/richtext  C:  C: Happy birthday to you  C: Happy birthday to you  C: Happy birthday dear <bold> Carolyn </bold>  C: Happy birthday to you  C:  C: --qwertyuiopasdfghjklzxcvbnm  C: Content-Type: message/external-body;  C: access-type="anon-ftp";  C: site="bicycle.cs.washington.edu";  C: directory="pub";  C: name="birthday.snd"  C:  C: content-type: audio/basic  C: content-transfer-encoding: base64  C: --qwertyuiopasdfghjklzxcvbnm  C:  S: 250 сообщение принято  C: QUIT  S: 221 ee.uwa.edu.au разрываю соединение |

В нашем примере сообщение должно быть послано только одному получателю, поэтому используется только одна команда RCPT. Использование этой команды несколько раз позволяет посылать одно сообщение нескольким пользователям. Каждое из них подтверждается или отвергается индивидуально.

Наконец, хотя синтаксис четырехсимвольных команд строго определен, синтаксис ответов не столь строг. Правила определяют только числовой код в начале строки. Все, что следует за этим кодом, может считаться комментарием и зависит от конкретной реализации протокола.

Базовый протокол SMTP не включает аутентификацию. Это значит, что команда FROM в примере может выдать любой адрес отправителя, который ей понравился. Это крайне удобно для отсылки спама. Другое ограничение заключается в том, что SMTP передает сообщения ASCII, а не бинарные данные. Именно для этого нужна была кодировка base64 для передачи контента MIME. Однако с этой кодировкой при передаче почты недостаточно используется пропускная способность, что неприятно, если отправляется объемное сообщение. Третье ограничение заключается в том, что SMTP передает непосредственно сообщение. Оно не шифруется и не защищено.

Чтобы справиться с этими проблемами к SMTP было добавлено расширение. Оно является обязательной частью стандарта RFC 5321. Использование SMTP с расширениями называется ESMTP (Extended SMTP – расширенный SMTP).

Клиенты желающие использовать расширенную версию, в начале высылают EHLO вместо HELO. Если этот вариант отвергается, сервер работает с обычным SMTP. Если EHLO принимается, сервер отвечает, какие расширения он поддерживает. Несколько стандартных расширений показано в таблице 1.5.

Таблица 1.5. Некоторые расширения SMTP

|  |  |
| --- | --- |
| Ключевое слово | Описание |
| AUTH | Аутентификация пользователя |
| BINARYMIME | Сервер принимает бинарные сообщения |
| CHUNKING | Сервер принимает большие сообщения по частям |
| SIZE | Проверка размера сообщения перед попыткой отсылки |
| STARTTLS | Переключение на безопасный канал TLS |
| UTF8SMTP | Интернационализированный адрес |

**Подача почтовых сообщений.** Первоначально пользовательские агенты запускались на том же компьютере, что и агенты передачи сообщений, пересылающие почту. При таком варианте все, что необходимо для отсылки сообщения, - возможность пользовательского агента связаться с локальным почтовым сервером, используя только что описанный диалог. Однако этот вариант уже не так широко распространен.

Пользовательские агенты часто работают на ноутбуках, домашних компьютерах и мобильных телефонах, а они не всегда подключены к Интернету. Агенты передачи сообщений работают на серверах провайдеров и крупных компаний, которые постоянно подключены к Интернету. Также SMTP может задействовать порт 587, а не порт 25, и SMTP-сервер может проверять и исправлять формат сообщений, отосланных пользовательским агентом. Чтобы узнать больше об использовании SMTP при подаче почты, следует посмотреть стандарт RFC 4409.

**Передача сообщений.** Чтобы определить верный почтовый сервер, запрашивается DNS. Запрос DNS делается для записей MX (mail exchanger – запись для обмена почтовыми сообщениями) домена, который написан в адресе после «@». Этот запрос возвращает упорядоченный список имен и IP-адресов одного или более почтовых серверов.

Затем отсылающий агент передачи почты создает TCP –соединение с IP-адресом почтового сервера на порте 25, чтобы связаться с принимающим агентом передачи почты. Для передачи сообщения используется SMTP. Принимающий агент передачи почты затем помещает сообщения для получателя в нужный почтовый ящик.

При помощи этого процесса доставки почта путешествует от начального до конечного агента передачи почтовых сообщений за один шаг. На этапе передачи нет промежуточных серверов. Однако этот процесс может повторяться несколько раз, в случае рассылки, например. [1. Стр. 677-682]

### Окончательная доставка сообщений

Наше почтовое сообщение почти доставлено. Оно прибыло в почтовый ящик. Осталось только передать копию сообщения пользовательскому агенту, чтобы оно могло отобразиться. Это шаг в архитектуре на рис… Агент передачи почты просто писал новые сообщения в конце файла почтового ящика, а пользовательский агент просто проверял файл почтового ящика на наличие новых сообщений. Сегодня пользовательский агент, работающий на персональном компьютере, ноутбуке или мобильном телефоне, вероятно, будет размещаться не на почтовом сервере компании или провайдера. Пользователи хотят иметь доступ к своей почте вне зависимости от того, где они находятся. Им нужен доступ к почте на работе, с домашнего компьютера, с ноутбука в командировках или из интернет-кафе, когда они отдыхают. Им нужна возможность работать без постоянного соединения с сетью, периодически подключаясь к Интернету для получения входящих сообщений и отправки исходящих. Более того, каждый пользователь может запускать несколько агентов в зависимости от того, каким компьютером ему удобнее пользоваться в данный момент. Несколько пользовательских агентов даже могут работать одновременно.

В данном случае пользовательский агент должен отображать содержимое почтового ящика и позволять работать с ним удаленно. Для этой цели может задействоваться несколько разных протоколов, но только не SMTP. SMTP — это протокол, основанный на проталкивании данных. Для передачи сообщения ему необходимо соединение с удаленным сервером. Таким образом, окончательная доставка не может быть осуществлена по двум причинам: из-за того, что почтовый ящик должен храниться на агенте передачи почты, и из-за того, что пользовательский агент может быть не подключен к Интернету, когда SMTP пытается передать сообщение.

**IMAP – протокол доступа к электронной почте в Интернете.** Один из главных протоколов, использующихся для конечной доставки, - IMAP (Internet Mail Access Protocol). Четвертая версия этого протокола определена в RFC 3501. Чтобы использовать IMAP, почтовый сервер запускает IMAP-сервер, который проверяет порт 143. Пользовательский агент запускает IMAP-клиент. Этот клиент соединяется с сервером и начинает передавать команды из перечисленных в табл. 1.6. Сначала клиент запускает защищенный транспорт, если он используется (чтобы сохранить конфиденциальность сообщений и команд), а затем вводит логин и пароль или иначе авторизуется на сервере. После входа в систему можно воспользоваться множеством команд, чтобы просматривать папки и сообщения или даже части сообщений, помечать письма галочками для дальнейшего удаления и разбирать их по папкам. Обратите внимание на то, что во избежание неразберихи мы в этой главе используем слово «папка» ( folder), основываясь на том, что почтовый ящик пользователя состоит из нескольких таких папок. Однако в спецификации IMAP вместо папки используется термин mailbox («почтовый ящик»). Таким образом, получается, что у пользователя есть много почтовых ящиков IMAP, каждый из которых пользователь видит как папку. Протокол IMAP обладает разнообразными возможностями, как, например, способность упорядочивать почту не по порядку ее поступления, а по атрибутам писем (на- пример, «сначала дайте мне письмо от Элис»). На сервере также могут быть поисковые инструменты, чтобы можно было найти сообщения, удовлетворяющие определенным критериям, так что клиент увидит только их. IMAP — это улучшенная версия ранее разработанного протокола окончательной доставки POP3 (Post Office Protocol, version 3 — протокол почтового отделения, версия 3), который определен в RFC 1939. Протокол POP3 проще, но он поддерживает меньше возможностей и является менее безопасным при обычном использовании. Обычно почта загружается на компьютер с пользовательским агентом, а не остается на почтовом сервере. Это облегчает жизнь серверу, но усложняет ее пользователю. Читать почту с нескольких компьютеров становится гораздо сложнее, к тому же, если компьютер с пользовательским агентом сломается, вся почта будет безвозвратно утеряна. Тем не менее, POP3 все еще используется. Кроме того, могут использоваться и частные протоколы, так как протокол работает между почтовым сервером и пользовательским агентом, который может обеспечиваться той же компанией. Примером почтовой системы с частным протоколом является Microsoft Exchange.

Таблица 1.6. Команды IMAP (версия 4)

|  |  |
| --- | --- |
| Команда | Описание |
| CAPABILITY | Перечислить возможности сервера |
| STARTTLS | Запустить безопасный транспорт |
| LOGIN | Войти на сервер, используя имя пользователя и пароль |
| AUTHENTICATE | Авторизоваться иным способом |
| SELECT | Выбрать папку |
| EXAMINE | Выбрать папку, предназначенную только для чтения |
| CREATE | Создать папку |
| DELETE | Удалить папку |
| RENAME | Переименовать папку |
| SUBSCRIBE | Добавить папку к активному набору |
| UNSUBSCRIBE | Удалить папку из активного набора |
| LIST | Перечислить доступные папки |
| LSUB | Перечислить активные папки |
| STATUS | Узнать статус папки |
| APPEND | Добавить сообщение в папку |
| CHECK | Просмотреть состояние выбранной папки (что именно входит в понятие «состояние», зависит от конкретной реализации сервера). Создать контрольную точку для папки |
| FETCH | Просмотреть сообщения, находящиеся в папке |
| SEARCH | Найти сообщения, находящиеся в папке |
| STORE | Изменить метки сообщения |
| COPY | Сделать копию сообщения в папке |
| EXPUNGE | Удалить отмеченные сообщения |
| UID | Вызвать команды, используя уникальные идентификаторы |
| NOOP | Ничего не делать |
| CLOSE | Удалить помеченные сообщения и закрыть папку |
| LOGOUT | Выйти из системы и закрыть соединение |

**POP3 – протокол почтового отделения версии 3.** Post Office Protocol (POP) - протокол доставки почты пользователю из почтового ящика почтового сервера РОР. Многие концепции, принципы и понятия протокола POP выглядят и функционируют подобно SMTP. Команды POP практически идентичны командам SMTP, отличаясь в некоторых деталях. На рис. 1.2 изображена модель клиент-сервер по протоколу POP. Сервер POP находится между агентом пользователя и почтовыми ящиками.

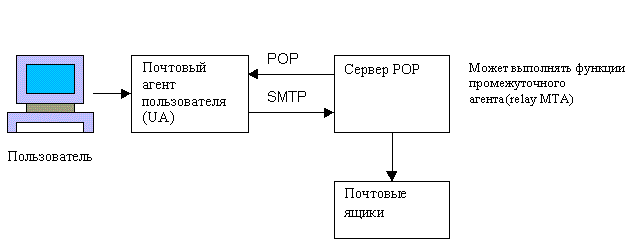


Рисунок 1.2. Конфигурация модели клиент-сервер по протоколу POP (любой версии)

В настоящее время существуют две версии протокола POP - РОР2 и РОРЗ, обладающими примерно одинаковыми возможностями, однако несовместимыми друг с другом. Дело в том, что у РОР2 и РОРЗ разные номера портов протокола. Между ними отсутствует связь, аналогичная связи между SMTP и ESMTP. Протокол РОРЗ не является расширением или модификацией РОР2 - это совершенно другой протокол. РОР2 определен в документе RFC 937 (Post Office Protocol-Version 2, Butler, et al, 1985), a РОРЗ - в RFC 1225 (Post Office Protocol-Version 3, Rose, 1991). Далее кратко рассмотрим POP вообще и более подробно - РОРЗ. PОРЗ разработан с учетом специфики доставки почты на персональные компьютеры и имеет соответствующие операции для этого.

**Назначение протокола POP3.** Ранее почтовые сообщения большинства сетей доставлялись непосредственно от одного компьютера к другому. И если пользователь часто менял рабочие компьютеры или один компьютер принадлежал нескольким пользователям, существовали определенные проблемы. В наши дни общепринята доставка сообщения не на компьютеры пользователя, а в специальные почтовые ящики почтового сервера организации, который круглосуточно работает (включен).

**Описание протокола POP3.** Конструкция протокола РОРЗ обеспечивает возможность пользователю обратиться к своему почтовому серверу и изъять накопившуюся для него почту. Пользователь может получить доступ к РОР-серверу из любой точки доступа к Интернет. При этом он должен запустить специальный почтовый агент (UA), работающий по протоколу РОРЗ, и настроить его для работы со своим почтовым сервером. Итак, во главе модели POP находится отдельный персональный компьютер, работающий исключительно в качестве клиента почтовой системы (сервера). Подчеркнем также, что сообщения доставляются клиенту по протоколу POP, а посылаются по-прежнему при помощи SMTP. То есть на компьютере пользователя существуют два отдельных агента-интерфейса к почтовой системе - доставки (POP) и отправки (SMTP). Разработчики протокола РОРЗ называет такую ситуацию "раздельные агенты" (split UA). Концепция раздельных агентов кратко обсуждается в спецификации РОРЗ.

В протоколе РОРЗ оговорены три стадии процесса получения почты: авторизация, транзакция и обновление. После того как сервер и клиент РОРЗ установили соединение, начинается стадия авторизации. На стадии авторизации клиент идентифицирует себя для сервера. Если авторизация прошла успешно, сервер открывает почтовый ящик клиента и начинается стадия транзакции. В ней клиент либо запрашивает у сервера информацию (например, список почтовых сообщений), либо просит его совершить определенное действие (например, выдать почтовое сообщение). Наконец, на стадии обновления сеанс связи заканчивается. В табл. 1.7 перечислены команды протокола РОРЗ, обязательные для работающей в Интернет реализации минимальной конфигурации.

Таблица 1.7. Команды протокола POP версии 3 (для минимальной конфигурации)

|  |  |
| --- | --- |
| Команда | Описание |
| USER | Идентифицирует пользователя с указанным именем |
| PASS | Указывает пароль для пары клиент-сервер |
| QUIT | Закрывает TCP-соединение |
| STAT | Сервер возвращает количество сообщений в почтовом ящике плюс размер почтового ящика |
| LIST | Сервер возвращает идентификаторы сообщений вместе с размерами сообщений (параметром команды может быть идентификатор сообщения) |
| RETR | Извлекает сообщение из почтового ящика (требуется указывать аргумент-идентификатор сообщения) |
| DELE | Отмечает сообщение для удаления (требуется указывать аргумент - идентификатор сообщения) |
| NOOP | Сервер возвращает положительный ответ, но не совершает никаких действий |
| LAST | Сервер возвращает наибольший номер сообщения из тех, к которым ранее уже обращались |
| RSET | Отменяет удаление сообщения, отмеченного ранее командой DELE |

В протоколе РОРЗ определено несколько команд, но на них дается только два ответа: +ОК (позитивный, аналогичен сообщению-подтверждению АСK) и -ERR (негативный, аналогичен сообщению "не подтверждено" NAK). Оба ответа подтверждают, что обращение к серверу произошло и что он вообще отвечает на команды. Как правило, за каждым ответом следует его содержательное словесное описание. В RFC 1225 есть образцы нескольких типичных сеансов РОРЗ. Сейчас мы рассмотрим несколько из них, что даст возможность уловить последовательность команд в обмене между сервером и клиентом.

**Авторизация пользователя.** После того как программа установила TCP-соединение с портом протокола РОРЗ (официальный номер 110), необходимо послать команду USER с именем пользователя в качестве параметра. Если ответ сервера будет +ОК, нужно послать команду PASS с паролем этого пользователя:

|  |
| --- |
| CLIENT: USER kcope  ERVER: +ОК  CLIENT: PASS secret  SERVER: +ОК kcope's maildrop has 2 messages (320 octets) |

(В почтовом ящике kcope есть 2 сообщения (320 байтов) ...)

**Транзакции POP3.** После того как стадия авторизации окончена, обмен переходит на стадию транзакции. В следующих примерах демонстрируется возможный обмен сообщениями на этой стадии.

Команда STAT возвращает количество сообщений и количество байтов в сообщениях:

|  |
| --- |
| CLIENT:STAT SERVER: +ОК 2 320 |

Команда LIST (без параметра) возвращает список сообщений в почтовом ящике и их размеры:

|  |
| --- |
| CLIENT: LIST  SERVER: +ОК 2 messages (320 octets)  SERVER: 1 120  SERVER: 2 200  SERVER: . ... |

Команда LIST с параметром возвращает информацию о заданном сообщении:

|  |
| --- |
| CLIENT: LIST 2  SERVER: +ОК 2 200 ...  CLIENT: LIST 3  SERVER: -ERR no such message, only 2 messages in maildrop |

Команда TOP возвращает заголовок, пустую строку и первые десять строк тела сообщения:

|  |
| --- |
| CLIENT: TOP 10  SERVER: +ОК  SERVER: <the POP3 server sends the headers of the message,a blank line, and the first 10 lines of the message body> |

(сервер POP высылает заголовки сообщений, пустую строку и первые десять строк тела сообщения)

|  |
| --- |
| SERVER: . ...  CLIENT: TOP 100  SERVER: -ERR no such message |

Команда NOOP не возвращает никакой полезной информации, за исключением позитивного ответа сервера. Однако позитивный ответ означает, что сервер находится в соединении с клиентом и ждет запросов:

|  |
| --- |
| CLIENT:NOOP SERVER: +ОК |

Следующие примеры показывают, как сервер POP3 выполняет действия. Например, команда RETR извлекает сообщение с указанным номером и помещает его в буфер местного UA:

|  |
| --- |
| CLIENT: RETR 1  SERVER: +OK 120 octets  SERVER: <the POPS server sends the entire message here> |

(РОРЗ-сервер высылает сообщение целиком)

Команда DELE отмечает сообщение, которое нужно удалить:

|  |
| --- |
| CLIENT: DELE 1  SERVER: +OK message 1 deleted ...  (сообщение 1 удалено)  CLIENT: DELE 2  SERVER: -ERR message 2 already deleted |

сообщение 2 уже удалено)

Команда RSET снимает метки удаления со всех отмеченных ранее сообщений:

|  |
| --- |
| CLIENT:RSET SERVER: +OK maildrop has 2 messages (320 octets) |

(в почтовом ящике 2 сообщения (320 байтов) )

Как и следовало ожидать, команда QUIT закрывает соединение с сервером:

|  |
| --- |
| CLIENT: QUIT  SERVER: +OK dewey POP3 server signing off  CLIENT: QUIT  SERVER: +OK dewey POP3 server signing off (maildrop empty)  CLIENT: QUIT  SERVER: +OK dewey POP3 server signing off (2 messages left) |

Обратите внимание на то, что отмеченные для удаления сообщения на самом деле не удаляются до тех пор, пока не выдана команда QUIT и не началась стадия обновления. В любой момент в течение сеанса клиент имеет возможность выдать команду RSET, и все отмеченные для удаления сообщения будут восстановлены. [5]

### Веб-почта

Альтернативой IMAP и SMTP для предоставления почтовых услуг, набирающей все большую популярность, стало использование веб-интерфейса для отсылки и получения сообщений. Широко используемые системы веб-почты (Webmail) включают Google Mail, Microsoft Hotmail и Yahoo! Mail (необходимо отметить, что перечень этот далеко не исчерпывающий, а популярность различных сервисов веб-почты может отличаться в разных регионах. Например, в России Яндекс. Почта или Mail.Ru известны, возможно, больше, чем последние два пункта приведенного перечня). Веб-почта — это один из примеров программного обеспечения (в данном случае пользовательский почтовый агент), которое предоставляется как услуга, используя веб-технологии.

При этой архитектуре провайдер управляет почтовыми услугами как обычно, чтобы принимать сообщения для пользователей при помощи SMTP через порт 25. Однако пользовательский агент отличается. Он является не отдельной программой, а пользовательским интерфейсом, который предоставляется через веб-страницы. Это значит, что пользователи могут использовать любой понравившийся им браузер, чтобы получить доступ к своей почте и отсылать новые сообщения.

Когда пользователь заходит на почтовую веб-страницу провайдера, он видит форму, в которой ему нужно заполнить поля Имя пользователя и Пароль. Пароль и имя пользователя отправляются на сервер, где проверяется их правильность. Если вход в систему осуществлен корректно, сервер находит почтовый ящик пользователя и строит веб-страницу, на которой отображается содержание почтового ящика. Эта веб-страница отсылается на браузер клиента. Пользователь может щелкать на многих элементах страницы, отображающей его почтовый ящик, так что сообщения можно читать, удалять и т. д. Чтобы интерфейс лучше откликался на действия пользователя, веб-страницы часто включают программы на JavaScript. Эти программы запускаются локально на машине пользователя в ответ на определенные события (например, клики мышкой) и могут загружать на сервер и подгружать с сервера сообщения в фоновом режиме, чтобы подготовить к показу следующее письмо или подать новое. В этой модели подача почты происходит при использовании обычных веб-протоколов при помощи отправки данных по URL. Веб-сервер помещает сообщения в обычную, уже описанную нами систему доставки почты. Для безопасности также могут использоваться стандартные веб-протоколы. Эти протоколы связаны с шифрованием веб-страниц, а не с тем, является ли содержание страницы почтовым сообщением.

### Пользовательские агенты (почтовые клиены)

Пользовательский агент (почтовый клиент) – это программа (иногда называемая почтовым редактором или «читалкой»), управляемая множеством команд для составления и получения сообщений, а также для ответа на сообщения и управления почтовыми ящиками. Существует много популярных агентов, например Gmail от Google, Microsoft Outlook, Mozilla Thunderbird и Apple Mail. Внешне они сильно отличаются. Графический интерфейс большинства пользовательских агентов основан на меню или значках и требует наличия мышки или, на мобильных устройствах, возможности управления при помощи касания. Более старые программы, такие как Elm, mh и Pine, имеют интерфейсы, основанные на тексте, и работают при помощи ввода с клавиатуры однобуквенных команд. Функциональных различий нет, по крайней мере, в отношении текстовых сообщений.

Когда почтовый клиент запущен, он обычно отображает сводную информацию о сообщениях в почтовом ящике пользователя. Часто одному сообщению соответствует одна строка, а идут они в каком-либо порядке, выбранном при сортировке. Обычно последние сообщения находятся сверху списка. В сводной информации выделены ключевые поля сообщения, извлеченные из конверта или заголовка.

В строках используются поля «От», «Тема» и «Дата получения. Вся информация отформатирована удобным для пользователя образом; она базируется на полях сообщения, но не отображается буквально. Таким образом, те, кто отправляет сообщения без темы, часто сталкивается с тем, что из письма имеют низкий приоритет.

Возможно наличие многих других полей и маркеров. Значки рядом с темой сообщения могут обозначать, например, непрочитанную почту (конверт), прикрепленные файлы (скрепка), важные сообщения, по крайней мере, с точки зрения отправителя (восклицательный знак).

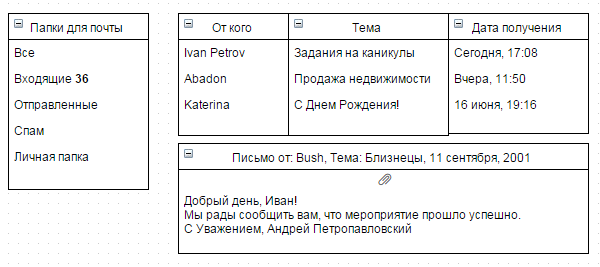


Рис. 1.3. Стандартные элементы интерфейса пользовательского агента (почтового клиента)

Так же возможны несколько вариантов сортировки. Самый распространенный основывается на времени получения письма, сначала более свежие, с маркером того, прочитано ли сообщение получателем или нет. Поля сводной информации и порядок сортировки могут быть настроены в соответствии с желаниями пользователя.

Пользовательские агенты также должны отображать входящие сообщения определенным образом, чтобы почту можно было читать. Часто предоставляется предварительный просмотр письма, чтобы пользователь мог решить, когда прочитать это сообщение. При предварительном просмотре могут использоваться маленькие значки или изображения, чтобы описать содержание письма.

После того как сообщение было прочитано, вы можете решить, что с ним делать. Это называется размещение сообщения. Среди опций есть удаление, написание ответа, пересылка сообщения другому пользователю и оставление сообщения в ящике для дальнейшей работы. Большинство программ снабжено одним почтовым ящиком для входящих писем набором папок для сохраненной почты. Папки позволяют хранить письма в разных местах в зависимости от отправителя, даты, темы или какой-то другой категории.

Распределение по папкам также может автоматически производить пользовательский агент до того, пользователь прочтет сообщение. Многие интернет-провайдеры и компании используют программное обеспечение, помечающее сообщение как важное или спам, так что пользовательский агент может распределить их по соответствующим папкам. Если сотни пользователей в одно и то же время получают одинаковые сообщения, они, вероятно являются спамом. Предварительно сортируя сообщения и помечая часть из них как «предположительно спам», пользовательский агент может избавить пользователей от массы работы по разделению нужного и ненужного.

Но какой же спам является наиболее популярным? Он генерируется набором взломанных компьютеров, которые называются ботнет, а его содержание зависит от страны, в которой вы живете. В Азии самыми популярными являются предложения поддельных дипломов, в США – дешевых лекарств и сомнительной продукции. Невостребованные счета в нигерийском банке также не теряют популярности.

Электронная почта прошла долгий путь с тех пор, как она была просто способом передачи данных. Сегодня почти все провайдеры поддерживают почтовые ящики объемом до 1 гигабайта, сохраняющие переписку пользователя за долгий период времени.

Еще одна полезная опция – это способность определенным образом автоматически отвечать на сообщения. Один из вариантов ответа – пересылка входящего сообщения на другой адрес, например на компьютер, принадлежащий коммерческой пейджинговой компании, чтобы вызвать пользователя, используя радио или спутниковую антенну, и отобразить строку «Тема» на его пейджере. Эти автоответчики должны работать на почтовых серверах, так как пользовательский агент может работать не все время, а также может только иногда извлекать почту. Однако интерфейс для автоматических ответов обычно предоставляется пользовательским агентом.

У сообщений, отсылаемых в почтовую систему – стандартный формат, который должен быть образован при помощи информации, предоставленной пользовательским агентом. Самая важная часть в передаче сообщения – это конверт, а самая важная часть конверта – адрес назначения. Этот адрес должен быть в формате, с которым могут работать агенты передачи сообщений.

Ожидаемый формат адреса – это user@dns-adress. Однако следует, отметить, что существуют и другие формы адресации. В частности, адреса стандарта X.400 абсолютно не похожи на DNS – адреса.

Х.400 – это стандарт ISO для систем обработки сообщений, который оно время соревновался с SMTP. SMTP победил в этом противостоянии, хотя системы Х.400 все еще используются. Адреса стандарта Х.400 состоят из пар атрибут = значение, разделенных слэшами, например: /C=RUS/ST=MOSCOW /L=KOROLEV/PF=16 MEMORIAL DR. /CN=IVAN IVANOV/

В этом адресе указано государство, область, город, личный адрес и имя получателя (IVAN IVANOV). Хотя форма адресации Х.400 значительно менее удобна, чем DNS, для пользовательских агентов этот вопрос является спорным, так как они поддерживают удобные с точки зрения пользователя псевдонимы, с помощью которых он может вводить или выбирать имя адресата и получать корректный электронный адрес. Поэтому пользователю не приходится вводить такую длинную строку.

Еще одна из опций – это списки рассылки, позволяющие пользователю рассылать одно и то же сообщение группе пользователей при помощи одной команды. Есть два варианта того, как может храниться список рассылки. Первый – локальное хранение пользовательским агентом. Второй – этот список может храниться удаленно на агенте передаче сообщений. [1. Стр. 664-668]

## Анализ существующих популярных почтовых клиентов

### Общая информация

В данном разделе будет рассматриваться исключительно десктопные почтовые клиенты, устанавливаемые для операционной системы Windows, которые способны подключаться к почтовому серверу, скачивать почту и отображать результат в собственном интерфейсе. Веб-почта затронута не будет. Так как целью дипломной работы является разработка «настольной» программы, а не веб-приложения.

Настольная программа обладает рядом преимуществ, по сравнению с веб-почтой. Например, почтовые клиенты более функциональны, чем веб-интерфейс почтовых сервисов. Более богатая функциональность проявляется в том, что в почтовом клиенте можно работать с несколькими почтовыми аккаунтами в едином программном интерфейсе, настраивать параметры внешнего вида, интегрироваться с другими программами типа органайзеров, больше инструментов для работы с потоком корреспонденции, настройка оповещения о поступлении новых писем. Ряд функций можно заблокировать и оставить только самое необходимое. Стоит отметить, что почтовые клиенты значительно экономят сетевой траффик за счет того, что подключаются и загружают письма только по мере необходимости, и не требуют постоянного подключения к Интернету. Так же они не загружают рекламу, размещенную на странице. Эта разница заметна при низкой скорости или слабом уровне сигнала.

### Анализ Microsoft Outlook 2013

Microsoft Outlook 2013 – компонент программного пакета Microsoft Office 2013. Это не только почтовый клиент и RSS-ридер для лент новостей, это мощнейший органайзер, который активно используется в деловой сфере. Помимо почтового и RSS-клиента, Microsoft Outlook 2013 – это также планировщик задач, календарь, база контактов, заметки. Рабочая область программы показана на рисунке 1.4

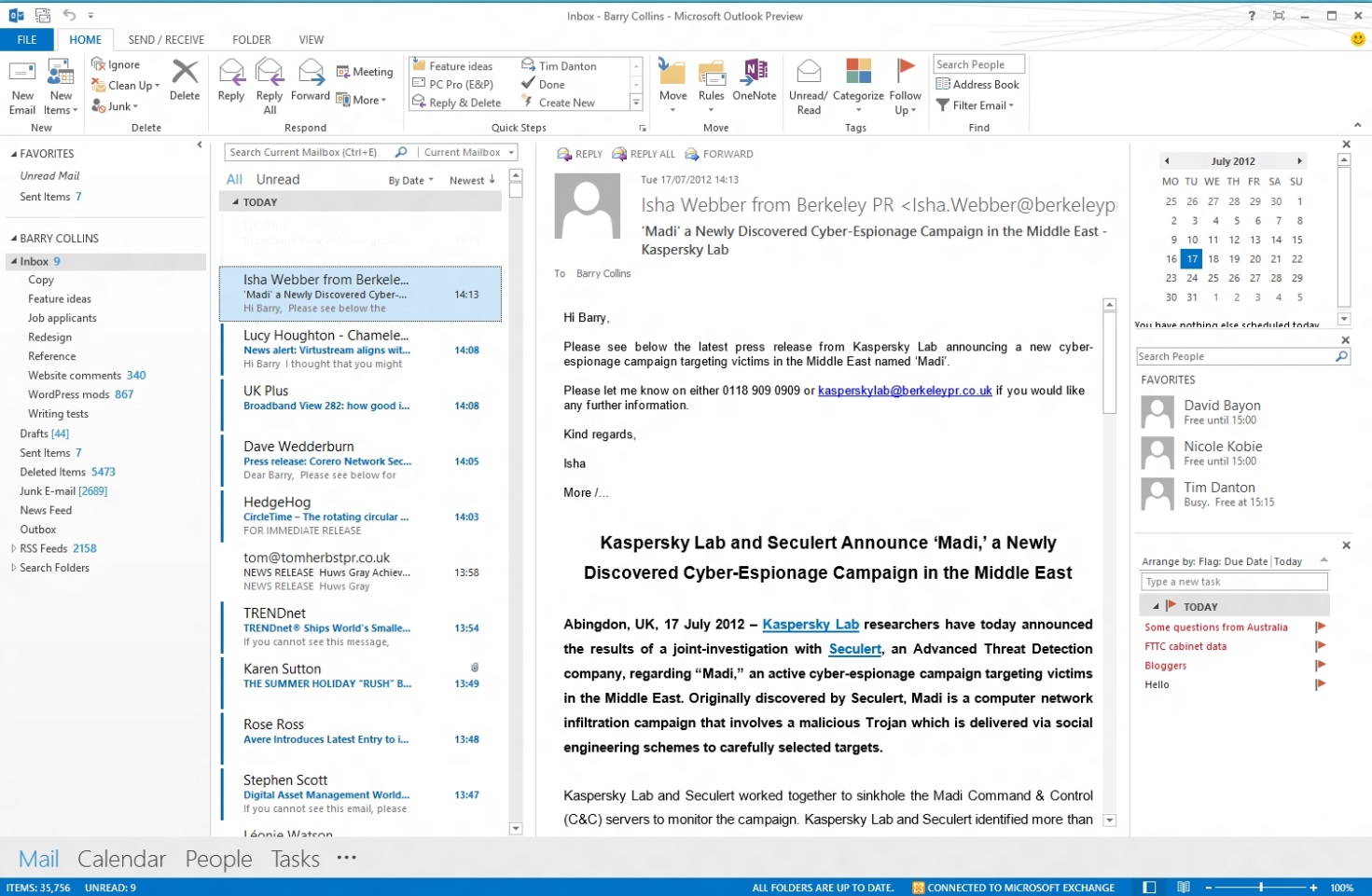


Рисунок 1.4. Главная рабочая область почтового клиента Microsoft Outlook 2013

Не следует путать Outlook с продуктом Outlook Express. Outlook Express — это бесплатный клиент электронной почты, встроенный в большинство версий Windows, выпущенных в период с 1998 по 2003 годы. В Windows 7 Outlook Express отсутствует, и вместо него с сайта можно установить пакет программ Windows Live Essentials, одним из компонентов которого является программа Windows Live Mail — аналог программы Outlook Express. В Windows Vista аналог Outlook Express называется Windows Mail.

Outlook может использоваться как отдельное приложение, так и выступать в роли клиента для почтового сервера Microsoft Exchange Server, что предоставляет дополнительные функции для совместной работы пользователей одной организации: общие почтовые ящики, папки задач, календари, конференции, планирование и резервирование времени общих встреч, согласование документов. Microsoft Outlook и Microsoft Exchange Server являются платформой для организации документооборота, так как они обеспечены системой разработки пользовательских плагинов и скриптов, с помощью которых возможно программирование дополнительных функций документооборота (и не только документооборота), не предусмотренных в стандартной поставке.

Интерфейс Microsoft Outlook 2013 выполнен в общем стиле Microsoft Office 2013. И пользователям, работающим хотя бы с одной программой офисного пакета разобраться с интерфейсом и функционалом Microsoft Outlook 2013 будет несложно.

Настройка для работы с почтой облегчена тем, что не нужно задавать параметры почтовых серверов. Приступать к работе с программой можно, указав всего лишь адрес электронной почты и пароль от неё.

Microsoft Outlook 2013 – интеллектуальный программный продукт. Прямо в форме отправки электронного письма можно выполнить форматирование отправляемого текста, проверку орфографии, отыскать значение слов с помощью встроенного словаря, а также перевести текст на другой язык.

Продукт обладает хорошей организацией контактов. Можно размещать все нужные контакты в одном представлении. Там же будут отображаться сведения о каждом из них и сообщения о последних публикациях в социальных сетях. Так что пользователь ничего не пропустит. Так же сразу будет видно, в каких сетях зарегистрирован интересующийся контакт, и можно удалить дублирующийся данные.

Имеется совместный доступ к календарю. Когда пользователи могут просматривать календари друг друга, им проще составлять расписания встреч и отвечать на приглашения.

Microsoft Outlook 2013 доступен для различных устройств, таких как: смартфоны, планшеты, ноутбуки, стационарные компьютеры. [7]

### Анализ TheBat!

The Bat! - самый защищенный и эффективный клиент электронной почты для Windows.

Главными задачами почтовой программы The Bat! являются: сохранение конфиденциальности переписки, удобство и экономия времени при работе с почтой.

Почтовый агент The Bat! защитит вашу информацию различными методами, в том числе шифрованием данных пользователя на жёстком диске и шифрованием трафика при помощи протоколов SSL/TLS. The Bat! позволяет сохранить тайну переписки, так как может работать без использования глобальных поставщиков услуг электронной почты и веб-интерфейса. Таким образом, ваши письма хранятся на вашем компьютере, а не в облачных хранилищах, где могут быть доступны посторонним лицам.

The Bat! позволяет работать с неограниченным числом почтовых ящиков, обрабатывать и хранить неограниченное количество писем.

**Защита данных.** The Bat! позволяет сохранить тайну переписки, защищая конфиденциальные данные от хищения и мониторинга извне, так как может работать без использования глобальных поставщиков услуг электронной почты и веб-интерфейса. The Bat! помогает вам защитить информацию различными методами, в том числе шифрованием данных пользователя на жёстком диске и шифрованием трафика.

* *Надёжное шифрование*  
  The Bat! предоставляет широкую поддержку протоколов аутентификации и шифрования при работе с почтовыми серверами для защиты от преднамеренного искажения данных, а также утечки конфиденциальной информации при передаче писем через Интернет. Почтовая программа The Bat! защищает конфиденциальность переписки, используя OpenPGP для подписи и шифрования электронной почты. Почтовый клиент предлагает встроенный PGP. Кроме того, можно подключить другие варианты программ, начиная от классической PGP 2.6.3 и до новейших PGP 9 и 10, а также GnuPG. Кроме того, в The Bat! можно использовать внутреннюю реализацию S/MIME или Microsoft CryptoAPI. The Bat! работает с Secure Socket Layer (SSL) v3.0 / Transport Layer Security (TLS) v1.0, v1.1
* *Резервное копирование*  
  Все настройки программы, адресные книги и почтовые папки могут быть сохранены в отдельном файле и затем восстановлены из резервной копии в случае нежелательного изменения информации или потери данных. Архив можно снабдить комментарием и защитить паролем (в этом случае содержимое резервной копии будет зашифровано). Кроме того, The Bat! позволяет создавать отдельную резервную копию для каждого почтового ящика или папки.
* *Шифрование почтовой базы*  
  В The Bat! Professional доступно шифрование почтовой базы, которая хранится на компьютере. При этом используется алгоритм шифрования AES в режиме CBC с размером ключа 128 бит. Шифрование почтовой базы (писем, адресных книг и файлов конфигурации) можно включить при первом запуске программы. При этом необходимо установить мастер-пароль для доступа к программе.
* *Автономная адресная книга*  
  The Bat! использует собственную адресную книгу, не связанную с адресной книгой Microsoft Windows Address Book. Адресная книга Microsoft Windows, используемая программами Microsoft Outlook, Outlook Express и прочими, является основной мишенью всех вирусных атак, так как вирусы извлекают адреса электронной почты из адресной книги Microsoft для дальнейшего распространения.
* *Уникальная функция парковки*  
  С помощью парковки можно закрепить письмо за определенной папкой и предохранить его от случайного удаления или перемещения.
* *Внутренний просмотрщик HTML*  
  Внутренний модуль просмотра HTML, встроенный в The Bat!, не зависит от системного модуля отображения HTML и от Microsoft Internet Explorer. Встроенный модуль The Bat! поддерживает язык гипертекста HTML 4.0 и стили форматирования документа CSS v2.0, игнорирует скрипты и исполняемый код, который является основной причиной вирусов. Модуль просмотра HTML, встроенный в The Bat!, не позволяет вирусам, использующим известные уязвимости HTML от Microsoft Internet Explorer (IFRAME, Automatic Execution of Embedded MIME type, и т.д.), проникнуть в ваш компьютер.
* *Управление загрузкой изображений*  
  В программу встроен инструмент для фонового извлечения изображений из писем в формате HTML. Его появление обусловлено все более частым проникновением вредоносных кодов в компьютеры пользователей через сообщения HTML, содержащие графику. Управление загрузкой изображений позволяет блокировать подозрительные и пропускать безопасные изображения, производя извлечение не в зависимости от папки или адресата сообщения, а исходя из опасности хоста или URL поступившего сообщения.
* *Выборочное скачивание*  
  Выборочное скачивание позволяет удалить письмо или оставить его на вашем POP3 сервере, не дожидаясь загрузки сообщения. Фильтры выборочного скачивания можно использовать для борьбы с вирусами и нежелательной рекламой.
* *Интерфейс к антивирусным сканерам*  
  The Bat! предоставляет интерфейс к известным антивирусным сканерам, за счёт чего реализуется возможность проверять прикреплённые файлы на вирусы.
* *Безопасная работа с прикреплёнными файлами*The Bat! предупреждает пользователя перед попыткой открытия файлов с подозрительными расширениями о возможных вирусах, позволяет запретить запуск файлов определенных типов. Кроме того, основным преимуществом The Bat! является то, что пользователь видит действительное расширение присоединенных файлов. Программа не скрывает тип присоединенных файлов от пользователя. Если присоединенный файл имеет "двойное расширение", например, photo.jpg.exe, The Bat! выдаст пользователю дополнительное предупреждение.
* *ASLR и DEP*The Bat! задействует такие компоненты Windows , как механизм Случайного Размещения в Адресном Пространстве (ASLR) и механизм Предотвращения Выполнения Данных (DEP) для защиты компьютера от вирусов и вредоносных кодов, распространяемых через электронную почту.
* *64-разрядная версия*  
  The Bat! x64 предоставляет владельцам 64-разрядных систем преимущества в скорости и безопасности. С точки зрения производительности, 64-х разрядный The Bat! на 5% быстрее, чем 32-разрядный в 64-битной операционной системе. Более того, использование 64-битных программ в аналогичных версиях Windows является более безопасным, так как угрозы, которые в частности нацелены на 32-битные программы безвредны для 64-битных.

**Повышение производительности.**

* *Фильтрация – одна из сильнейших сторон программы*  
  Для обработки корреспонденции применяется встроенный Сортировщик писем, который автоматически распределяет входящую, исходящую, прочитанную и обработанную почту по соответствующим папкам. Создание правил сортировки займет всего несколько минут, после чего вы можете навсегда забыть про ручное перемещение писем из папки в папку.
* *Автоматическая обработка сообщений*  
  Вышеупомянутый Сортировщик писем позволяет автоматизировать обработку сообщений: настроить автоответ, запуск внешнего приложения, добавить адресатов в адресную книгу, экспортировать или архивировать сообщения, извлечь присоединенные файлы и многое другое.
* *Шаблоны*  
  The Bat! может не только добавить подпись к сообщению, но и позволяет создавать шаблоны писем, используя множество специальных макросов. Любой ящик, папка или адресат могут иметь собственный шаблон для нового письма, ответа или пересылки. Применение шаблонов позволяет существенно ускорить подготовку почты, особенно когда письмо включает в себя стандартный или часто повторяющийся текст.
* *Быстрые шаблоны*  
  Быстрые шаблоны позволяют автоматизировать процесс создания писем и сэкономить рабочее время, вставляя в письмо заранее набранный текст. Если пользователю каждый день приходится отвечать в письмах на одни и те же вопросы, он по достоинству оценит эти шаблоны. В быстрых шаблонах, равно как и в обычных, доступен полный набор макросов, позволяющий задать любую, даже самую сложную последовательность действий.
* *Адресная книга*  
  Адресная книга содержит список адресатов, которые можно объединять в группы для классификации, либо для использования адресов в качестве списка рассылки, т.е. отправить письмо сразу всем членам группы. К каждой адресной записи можно присоединить фотографию, указать личные данные, сведения о месте работы, шаблоны писем к данному адресату, список S/MIME-сертификатов абонента, а также кодировку по умолчанию. В The Bat! присутствует возможность следить за Историей адресов, что позволяет просматривать и быстро находить любое сообщение любого из корреспондентов.
* *Совместимость с Microsoft Exchange*The Bat! подключается к серверам Microsoft Exchange, используя протокол MAPI для вызова или отправки сообщений. Необходимо лишь установить Microsoft Office Outlook или клиент Microsoft Exchange для предоставления компонентов подключения к Exchange, которые будет использовать The Bat!.
* *Национальные кодировки*  
  The Bat! корректно работает со всеми кодировками, включая все русские и восточноевропейские. Гибкая система настроек позволяет устанавливать свои кодировки для каждого ящика и каждого адресата. Повсеместно поддерживается кодировка Unicode. Также поддерживаются все системные шрифты и наборы символов.
* *Диспетчер писем*  
  Диспетчер позволяет обрабатывать почту на сервере, не дожидаясь ее загрузки. Отличное средство для осуществления удаленного контроля над вашей корреспонденцией. Особенно полезно в том случае, если необходимо оставить объемные письма на сервере, чтобы прочесть их позже, или удалить письма, не получая их.
* *Многоязыковой интерфейс*  
  The Bat! позволяет изменять язык интерфейса «на лету», без перезапуска программы.
* *Модуль просмотра изображений*  
  The Bat! оснащен собственным модулем просмотра изображений. Это означает, что доступ к вложенным изображениям гораздо быстрее, чем через внешнее приложение. Пользователи могут переключаться между всеми изображениями, вложенными в письмо. Модуль поддерживает поворот, алгоритмы изменения размера и масштаба, полноэкранный режим, вывод изображения на печать.
* *Импорт писем*  
  The Bat! позволяет импортировать письма из других почтовых клиентов:
  + Mozilla Thunderbird
  + Eudora Lite/Pro
  + Pegasus Mail
  + Netscape 6
  + Netscape Messenger v4.xx
  + Netscape Mail v2.xx, 3.xx
  + Microsoft Outlook Express 5.x / 6.x
  + Microsoft Office Outlook
  + MS Outlook Express 4 / MS Mail News

Импорт адресных книг из файлов:

* + LDIF-файл
  + Электронная визитная карточка (VCard)
  + Список адресов (текст)
  + Разделённый запятыми (текст)
  + Разделённый табуляциями (текст)
  + INI-файл
  + Сертификат X.509
  + Адресная книга Eudora
  + Файл-шаблон Pegasus
  + Контакты MS Outlook
  + MS Outlook Express (Адресная книга Windows)
  + Почта Windows (Vista)
  + Контакты Google
* *Поддержка интернационализированных доменных имен*Начиная с версии 6.0, The Bat! поддерживает работу с IDN - доменными именами. IDN - от английского Internationalized Domain Names - интернационализованные доменные имена, которые содержат символы национальных алфавитов, например, президент.рф, почта.рф и тд. К таким доменным именам относятся имена, составленные из букв не латинских алфавитов планеты: кириллица, арабский , китайский алфавит и др. А также можно использовать имена с символами латинского алфавита с диакритическими знаками (немецкий, французский и т.д.)
* *Встроенная поддержка RSS каналов*  
  RSS – это технология получения новых анонсов, опубликованных на сайте, блоге или форуме. Вы можете подписываться на неограниченное число RSS каналов, а также читать и сортировать информацию, поступающую через эти каналы. Все новости и сообщения с форума будут загружены в почтовый клиент, аккуратно собраны в одной ленте и представлены в удобном списке новостей, точно так же, как и обычные электронные письма. В списке новостей вы увидите не только название новости, но и её анонс, и таким образом сможете выбрать и перейти лишь к тем новостям, которые вас интересуют.
* *Работает с любой операционной системой Windows*  
  Системные требования The Bat! более чем скромны. The Bat! работает в Windows XP и последующих версиях операционной системы Windows с минимальным разрешением экрана 1024X768. Минимальных требований к объему памяти или скорости процессора не существует. [6]

### Выводы

Было рассмотрено только два почтовых клиента: Microsoft Outlook и TheBat!. Разумеется, список этим не ограничивается. Существуют такие продукты, как: Mozilla Thunderbird, Windows Live, Becky!, Eudora, Foxmail, Pegasus и т.д. Все они заслуживают внимания и тщательного анализа, чтобы составить некий рейтинг. Программы разные и у каждой есть свои плюсы и минусы. Для их выявления требуется пользоваться одним почтовым клиентов на протяжении месяца и по возможности «прощупать» все функции, предлагаемые программой. Что касается Microsoft Outlook и TheBat!, то с уверенностью можно сказать, что в функционале им нет равных. У обоих есть официальная русификация интерфейса, возможность установки расширений. Однако Microsoft Outlook входит в состав Microsoft Office, а за TheBat! придется заплатить отдельно. TheBat! Home стоит 631 руб., TheBat! Professional 883 руб. Зато TheBat! обладает по истине могущественной системой защиты, по сравнению со всеми остальными клиентами. В нем присутствует даже биометрическая защита писем.

# ПРОЕКТИРОВАНИЕ ПОЧТОВОГО КЛИЕНТА

* 1. Почтовый клиент «MailV»

Программа «MailV» является почтовым клиентом (пользовательским агентом) для ноутбуков и стационарных компьютеров с операционной системой Windows.

### Общие сведения

Программа обеспечивает прием сообщений формата text, xml, html с любого почтового сервера, их просмотр, мгновенный ответ отправителю, просмотр и загрузка вложений к письму.

Ниже представлены скриншоты окон программы.

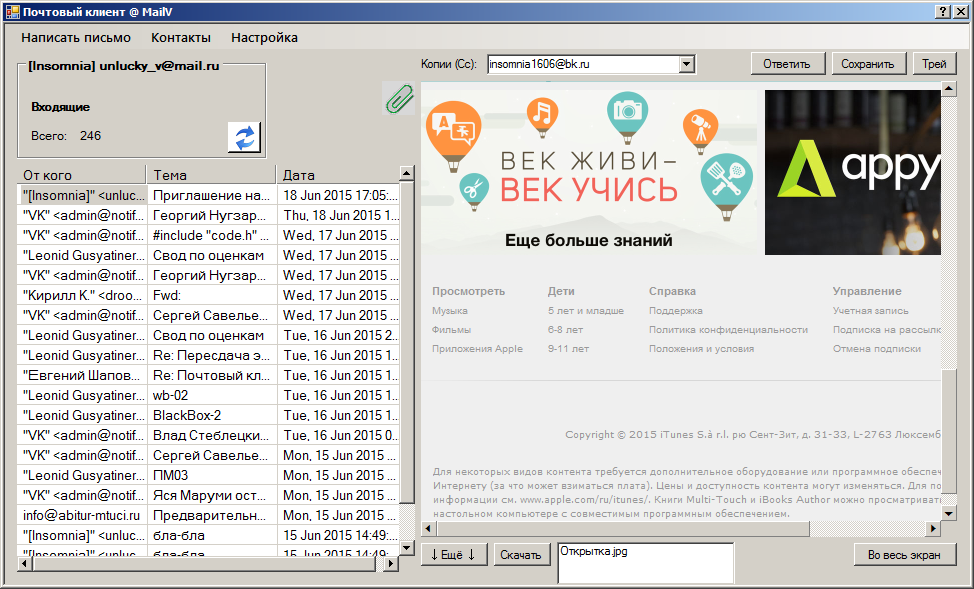


Рисунок 2.1. Главное окно почтового клиента

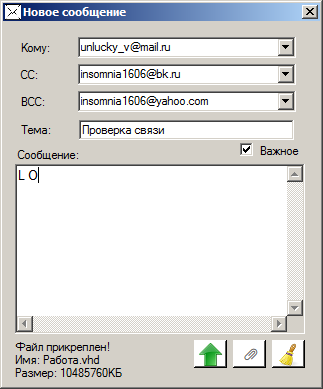


Рисунок 2.2. Написать новое сообщение

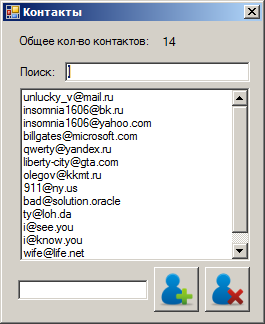


Рисунок 2.3. Контакты

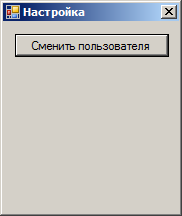


Рисунок 2.4. Настройка

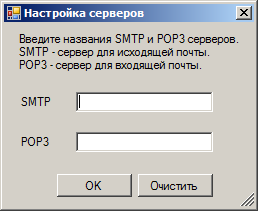


Рисунок 2.5. Настройка серверов

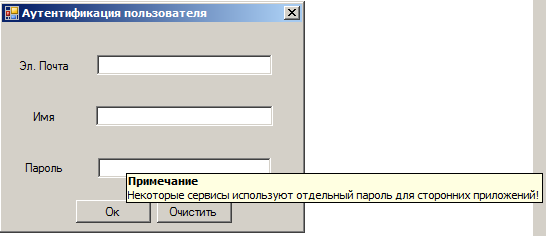


Рисунок 2.6. Аутентификация пользователя

### Возможности и особенности

*Возможности.*

* Поддержка формата письма text, xml, html. Пример сообщения, содержащее только текст показан на рисунке 2.1. Пример рекламной html рассылки показан на рисунке 2.2.
* Удобное распределение скаченных прикреплений по папкам. Например, пользователь с почтовым адресом «truecoder@domen.ru» отправил письмо с прикрепленным к нему текстовым файлом «расписание.txt». В директиве, где находится программа, создается каталог с названием адреса пользователя, причем заменяя все неправильные символы на пробелы. Далее туда загружается файл «расписание.txt». В дальнейшем все прикрепления, отправленные от пользователя с адресом «truecoder@domen.ru» будут помещаться в уже созданный для него каталог. Это обеспечивает некую порядочность в хранении файлов.
* Текст письма подлежащее отправке может содержать html код.
* Отправленное письмо может помечено как важное, т.е. иметь наивысший приоритет.
* Письмо может быть отправлено копией или скрытой копией одному или нескольким получателям.
* К письму может быть прикреплен один файл с любым расширением.
* Контакты автоматически переносятся в список получателей.
* Автоматический выбор smtp и pop серверов для почтовых служб: Mail.Ru, Gmail, Yandex.Почта, Rambler.
* Сохранение письма. Тип выходного файла на усмотрение пользователя.
* Работа программы в фоновом режиме, скрытой в область уведомлений.
* Просмотр письма во весь экран. Удобно использовать при просмотре Интернет-рассылок.
* Проверка новых входящих писем.
* Печать письма на принтере.
* Доступен выбор кодировки, при некорректном отображении сообщения.
* Наличие развертывания приложения в виде установщика (инсталлятора).

*Особенности.* Существует несколько нюансов при работе с почтовыми клиентами, о которых необходимо знать для корректного и полноценного использования:

1. *Знать названия почтовых серверов.* Для скачивания входящей почты по протоколу POP3, необходимо подключаться к серверу, на котором зарегистрирован адрес электронной почты пользователя. Например, для адреса somebody@mail.ru, имя сервера будет такое – pop.mail.ru. У каждого почтового сервиса есть свой почтовый сервер. Названия серверов обычно написаны в справке по настройке к доступу к почте из почтовых программ. Там же есть и руководство по настройке самых распространенных почтовых клиентов.
2. *Пароли для сторонних приложений.* При попытке авторизоваться в клиенте под корректным логином и паролем от аккаунта может возникнуть ошибка, говорящая о неправильности введенных данных. Как правило, это возникает потому, что некоторые почтовые сервисы, такие как Mail.Ru используют отдельный пароль для сторонних приложений. Для этого требуется сгенерировать надежный пароль автоматически в настройках той почты, которой вы пользуетесь. И далее авторизоваться в клиенте вводя новый пароль. Это сделано с целью избежать несанкционированного доступа к ящику злоумышленниками.
3. *Папки под паролем.* Если в ящике присутствует скрытая папка под паролем, то возможно возникнут трудности при доступе к письмам, содержащимся в ней. Папка находится под контролем веб-интерфейса и должно быть данные хранятся по-другому.
4. *Разрешить доступ почтовому клиенту скачивать почту.* Не все почтовые сервисы по умолчанию предоставляют десктопным почтовым клиентам доступ к скачиванию почты. Этот момент необходимо уточнить в настройках каждого отдельного почтового сервиса.

Используемая платформа - .NET Framework 4.0, это значит, что для запуска программы необходима версия платформы .NET Framework не ниже 4.0.

Если требуется отправить группу файлов, то следует соединить их в один архив, так как прикрепление может быть только одно.

Письма открываются в компоненте webbrowser. Webbrowser в свою очередь находится под управлением стандартного браузера Windows - Internet Explorer, это означает, что доступны все функции контекстного меню Интернет-обозревателя. В меню входит, например, печать, просмотр html кода, экспорт в Microsoft Excel, а так же навигация по ссылкам, внутри компонента.

Установщик программы выполнен с помощью инструмента «InstallShield Limited Edition».

InstallShield Limited Edition помогает создать файл установки и распространить его среди пользователей, чтобы они смогли установить классическое приложение или компонент без подключения к сети. InstallShield Limited Edition является бесплатным для пользователей всех версий Visual Studio кроме экспресс-выпусков. Он заменяет технологию установщика Windows, которая больше не поддерживается в Visual Studio. Кроме того, приложения и компоненты можно распространять с помощью технологии развертывания ClickOnce, которая требует подключения к сети. [10]

### Преимущества перед веб-интерфейсом

Использование почтовых клиентов обладает рядом преимуществ:

* + - 1. *Возможно получить доступ к своей почте оффлайн.* При отсутствии Интернета пользователь не сможет зайти через веб-интерфейс и соответственно посмотреть сообщения, которые он уже загружал. А вот использование почтового клиента позволяет сохранять копию загруженного письма у себя на компьютере. И в любой момент можно получить доступ к письму.
      2. *Безопасность.* Как правило, почтовые программы имеют дополнительные системы защиты, а письма располагаются локально на компьютере. И никто кроме хозяина персонального компьютера не сможет найти и расшифровать папки с корреспонденцией.
      3. *Горячие клавиши.* Определенные действия на вкус пользователя можно запрограммировать под сочетания клавиш на клавиатуре. Тогда, как в веб-интерфейсе вообще отсутствуют горячие клавиши или имеют очень ограниченные возможности.
      4. *Расширенный функционал.* Популярные почтовые клиенты такие как: Mozilla Thunderbird и TheBat! обладают колоссальным набором функций сортировки, поиска, фильтрации, организации хранения писем, редактировании текста письма, работы с большим количеством ящиков с разных сервисов и т.д. Всевозможные настройки позволяют подстроится под прихоти любого пользователя.
      5. *Реклама.* Наличие рекламы, ненужных ссылок на мессенджеры, социальные сети значительно тормозят процесс загрузки и мешают сосредоточиться на почте. Загрузка графического оформления тоже для большинства пользователей электронной почты считается излишней. Благодаря использованию почтового клиента значительно экономится Интернет траффик. Например, в случае лимитного тарифа или медленного соединения.

### Описание структуры программного комплекса

Разработчики, работающие в среде Visual Studio называют весь программный проект – решением. В решение добавляются новые формы, библиотеки, расширения (ссылки), подключаются новые файлы и т.д. В решении обязательно должна присутствовать точка входа в приложение.

Рассмотрим решение (или структуру) данного дипломного проекта. Оно представлено на рисунке 2.7.

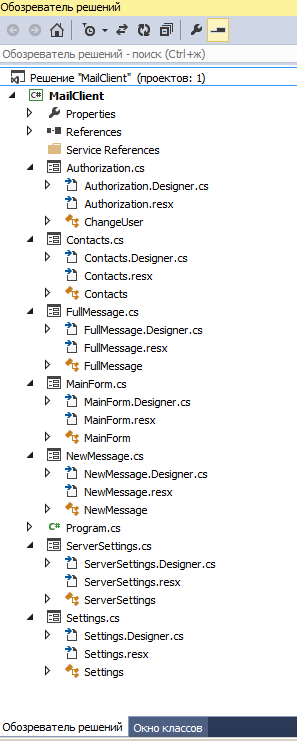


Рисунок 2.7. Структура обозревателя решения дипломного проекта

В обозревателе решений видно не всю иерархию файлов. При раскрытии дерева становится видно все функции, написанные относительно, рассматриваемой формы, а так же визуальные компоненты, расположенные на форме.

Решение состоит из следующих форм:

* + 1. Окно главной формы – MailForm.cs. Возникает при запуске программы.
    2. Окно контактов – Contacts.cs
    3. Окно отправки сообщений – NewMessage.cs
    4. Окно авторизации пользователя – Authorization.cs
    5. Окно настройки серверов – ServerSettings.cs
    6. Окно настройки – Settings.cs
    7. Окно просмотра письма во весь экран – FullMessage.cs

Взаимосвязь между экранными формами представлена на рисунке 2.2.

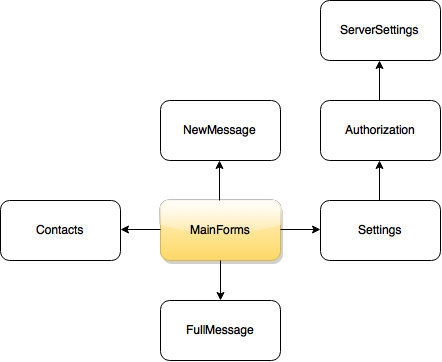


Рисунок 2.8. Взаимосвязь экранных форм

Так же к структуре можно отнести добавленную библиотеку OpenPop.NET.

В ресурсах проекта присутствуют иконки для кнопок и картинка, обозначающая вложение в письме.

### Описание процедур и функций

Описывается в порядке следования в листинге программы.

*Settings*

1. ChangeUser\_Click(). Обрабатывает нажатие на кнопку ChangeUser, следит за значением диалогового окна

*ServerSettings*

* 1. Clear\_Click(). Очищает поля для ввода названий серверов
  2. OK\_Click(). Возвращает значение диалогового окна
  3. ReturnSMTP(). Возвращает значение типа string, содержащее имя smtp сервера
  4. ReturnPOP3().Возвращает значение типа string, содержащее имя pop3 сервера

*NewMessage*

1. NewMessage\_Load(). Возникает при загрузке формы, загружает список отправителей
2. SendMail(). Функция отправки письма
3. SendMessage\_Click(). Возникает при нажатии на кнопку SendMessage, вызывает функцию SendMail()
4. AddFile\_Click(). Прикрепить файл (вложение) к письму
5. Clear\_Click(). Очистить поля темы и тела сообщения
6. label6\_MouseEnter(). Выдает справку при наведении на лейбл копии письма (Cc)
7. label7\_MouseEnter(). Выдает справку при наведении на лейбл скрытой копии письма (Bcc)

*FullMessage*

1. Close\_Click(). Закрывает окно. Выход из полноэкранного режима

*Contacts*

1. ContactsCount(). Присваивает лейблу количество контактов в списке
2. Contacts\_Load(). Возникает при загрузке формы. Загружает список контактов из текстового файла, если файла не существует, то создает его.
3. Search\_KeyUp(). Обеспечивает поиск контакта
4. AddUser\_Click(). Добавить контакт
5. DeleteUser\_Click(). Удалить один или несколько контактов
6. Contacts\_FormClosing(). Возникает при закрытии формы. Обновление файла с контактами

*Authorization*

1. Clear\_Click(). Очистить поля ввода логина пароли и имени
2. Ok\_Click(). Совершает авторизацию пользователя, проверяет корректность введённых данных
3. textBox2\_Enter(). Вызывает справку при заполнении пароля
4. textBox3\_Enter(). Вызывает справку при заполнении имени

*MainForm*

1. новоеСообщение\_Click(). Открывает окно для отправки нового сообщения
2. контакты\_Click(). Открывает окно для работы с контактами
3. настройка\_Click(). Открывает окно с пользовательскими настройками
4. MainForm\_HelpButtonClicked(). Вызов справки. Краткой информации о программе и способе связи с разработчиком
5. internetconnection(). Проверка наличия Интернет соединения. Возвращает true или false
6. userpopconnect(). Соединение с почтовым сервером и аутентификация
7. GetAllMess(). Скачать входящие письма
8. MainForm\_Load(). Возникает при запуске программы. Загружает информацию, проверяет, авторизован ли пользователь. Обеспечивает доступность того или иного компонента, а так же его видимость
9. InBox\_Click(). Возникает при нажатии кнопки Входящие. Вызывает функцию GetAllMess(), работает с прогрессбаром и лейблами
10. InBox\_MouseEnter(). Возникает при наведении курсора на Входящие
11. InBox\_MouseLeave(). Возникает, когда курсор находится не в фокуса Входящие
12. listView1\_MouseClick(). Возникает при нажатии ПКП по элементу в листвью
13. listView1\_ItemActivate(). Открывает выбранное письмо
14. удалить\_Click(). Удалить письмо
15. Save\_Click(). Сохранить письмо
16. Answer\_Click(). Ответить отправителю
17. Tray\_Click(). Скрыть программу в трей (область уведомлений)
18. notifyIcon1\_MouseClick(). Вернуть из трея в исходный вид
19. ShowFullMess\_Click(). Показать письмо целиком во весь экран
20. MainForm\_FormClosing(). Возникает при закрытии формы. Освобождение ресурсов. Удаление писем помеченных, как удаленные
21. UpdateB\_Click(). Обновить входящие письма
22. More\_Click(). Показать еще письма. Расширить список
23. DownloadB\_Click(). Скачать прикрепления к письму при их наличии

### Системные требования

Системные требования для программы сведены в таблицу 2.1.

Требования к системе для .NET Framework 4.0 показаны в таблице 2.2.

Таблица 2.1. Системные требования для программы

|  |  |
| --- | --- |
| Описание | Критерии |
| Наличие пространства на жестком диске для программы и ее составляющих | Не меньше 1 МБ |
| Наличие установленной платформы .NET Framework (требования см. в таблице 2.2.) | Версия 4.0 или выше |
| Оперативная память (ОЗУ) | 50 Мб или больше |
| Поддерживаемая операционная система | Windows 7; Windows 7 Service Pack 1; Windows Server 2003 Service Pack 2; Windows Server 2008; Windows Server 2008 R2; Windows Server 2008 R2 SP1; Windows Vista Service Pack 1; Windows XP Service Pack 3 |
| Наличие Интернет соединения | Рекомендуемая скорость получения – 3.80 Мбит/с, скорость передачи – 6.70 Мбит/с |

Таблица 2.2. Системные требования для .NET Framework 4.0

|  |  |
| --- | --- |
| Описание | Критерии |
| Поддерживаемая операционная система | Windows 7; Windows 7 Service Pack 1; Windows Server 2003 Service Pack 2; Windows Server 2008; Windows Server 2008 R2; Windows Server 2008 R2 SP1; Windows Vista Service Pack 1; Windows XP Service Pack 3 |
| Поддерживаемые архитектуры | x86, x64, ia64 (некоторые возможности не поддерживаются на ia64, например WPF) |
| Процессор | Рекомендуемый минимум: процессор Pentium с тактовой частотой 1 ГГц или выше |
| Оперативная память | 512 МБ или больше |
| Наличие дискового пространства | Для x86 минимум 850 МБ, для x64 минимум 2 Гб |
| Предварительные требования | Установщик Windows 3.1 или более поздней версии, Internet Explorer 5.01 или более поздней версии |

## Обоснование выбора инструментов, сред и языков

### Задействованные инструменты, языки и технологии

Дипломная работа была выполнена в интегрированной среде Microsoft Visual Studio Professional 2013, которая является одним из линейки продуктов компании Microsoft.

Компонентом среды Microsoft Visual Studio Professional 2013 был выбран Visual C#.

Языком программирования соответственно стал C#.

Для разработки программы с графическим интерфейсом, я воспользовался поддержкой технологии Windows Forms, являющийся частью платформы Microsoft .NET Framework.

Так как стандартные библиотеки компонента Visual C# не содержат возможностей работы с протоколом POP3, я решил воспользоваться сторонней библиотекой OpenPop.NET, которую нашел в Интернете.

### Описание и обоснование выбранных инструментов, языков и технологий

*Microsoft Visual Studio Professional 2013*

Это единая среда разработки конкурентоспособных приложений для пользовательских устройств и облака для домашнего и корпоративного применения.

Данный продукт позволяет разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio Professional 2013 включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Стоимость продукта: 27 345,00 руб. Однако для студентов и преподавателей некоторые продукты из линейки Microsoft можно приобрести бесплатно, доказав, что вы являетесь студентом колледжа или ВУЗа фотографией вашего студенческого билета. Заказать ключ можно на сайте dreamspark.ru [8,3]

*Системные требования*

Таблица 2.3. Системные требования к системе для среды VS Pro 2013

|  |  |
| --- | --- |
| Процессор | 1,60 Ггц или быстрее |
| Операционная система | Windows 7 с пакетом обновления 1 (SP1) (32- и 64-разрядные версии), Windows 8 (32- и 64-разрядные версии), Windows 8.1 (32- и 64-разрядные версии), Windows Server 2008 R2 с пакетом обновления 1 (SP1) (64-разрядная версия), Windows Server 2012 (64-разрядная версия), Windows Server 2012 R2 (64-разрядная версия) |
| Память | 1 ГБ ОЗУ (1,5 ГБ при работе на виртуальной машине) |
| Пространство на жестком диске | 10 ГБ |
| Браузер | Эта версия Visual Studio лучше всего работает с Internet Explorer 10 или последующими версиями. Если браузер Internet Explorer 10 на компьютере не установлен, то некоторые функции могут не действовать надлежащим образом. |
| Связь | Обязательный доступ к Интернету |
| Видеоадаптер | Видеоадаптер с поддержкой DirectX 9 и разрешения экрана 1024 x 768 или выше |
| Другие требования к системе | Жесткий диск 5400 об/мин |

*Visual C#*

C# (произносится "Си-шарп") является языком программирования, который разработан для создания множества приложений, работающих в среде .NET Framework. Язык C# прост, типобезопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой разработки приложений, но при этом сохраняет выразительность и элегантность, присущую С-подобным языкам. Так же язык считается кроссплатформенным.

Visual C# — это реализация языка C# корпорацией Майкрософт. Поддержка Visual C# в Visual Studio обеспечивается с помощью полнофункционального редактора кода, компилятора, шаблонов проектов, конструкторов, мастеров кода, мощного и удобного отладчика и многих других средств. Библиотека классов .NET Framework предоставляет доступ ко многим службам операционной системы и к другим полезным, хорошо спроектированным классам, что существенно ускоряет цикл разработки. [3]

По ссылке https://msdn.microsoft.com/ru-ru/library/vstudio/kx37x362.aspx доступны справочник по языку C#, примеры и спецификации.

*Windows Forms*

Этоинтерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причем управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке MFC, изначально написанной на языке C++. С другой стороны, WF не предлагает парадигму, сравнимую с MVC. Для исправления этой ситуации и реализации данного функционала в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является User Interface Process Application Block, выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

Внутри .NET Framework, Windows Forms реализуется в рамках пространства имён System.Windows.Forms. [3,8]

*Библиотека OpenPOP.NET*

OpenPop.NET – это библиотека реализации клиента POP3 и MIME парсер с открытым исходным кодом, написанной на C#.

Библиотека проста в использовании и имеет достаточно возможностей для разработки почтового клиента, который сумеет работать с входящими письмами.

Сам проект находится в активном состоянии и постоянно обновляется на GitHub. По ссылке https://github.com/foens/hpop можно просмотреть исходный код библиотеки. Открытость проекта позволяет лучше понять работу библиотеки и исправить найденные ошибки.

По ссылке http://hpop.sourceforge.net/ можно ознакомиться с данной библиотекой. На сайте имеется очень подробная документация не только по электронной почте, но и примеры работы с библиотекой. Пример того, как скачать все сообщения с сервера приведен в листинге 2.1. [2]

Листинг 2.1. Скачать все входящие сообщения с сервера

|  |
| --- |
| Here is an example of how you can download all messages from one POP3 server. It takes the various connection settings as arguments.  /// <summary>  /// Example showing:  /// - how to fetch all messages from a POP3 server  /// </summary>  /// <param name="hostname">Hostname of the server. For example: pop3.live.com</param>  /// <param name="port">Host port to connect to. Normally: 110 for plain POP3, 995 for SSL POP3</param>  /// <param name="useSsl">Whether or not to use SSL to connect to server</param>  /// <param name="username">Username of the user on the server</param>  /// <param name="password">Password of the user on the server</param>  /// <returns>All Messages on the POP3 server</returns>  public static List<Message> FetchAllMessages(string hostname, int port, bool useSsl, string username, string password)  {  // The client disconnects from the server when being disposed  using(Pop3Client client = new Pop3Client())  {  // Connect to the server  client.Connect(hostname, port, useSsl);  // Authenticate ourselves towards the server  client.Authenticate(username, password);  // Get the number of messages in the inbox  int messageCount = client.GetMessageCount();  // We want to download all messages  List<Message> allMessages = new List<Message>(messageCount);  // Messages are numbered in the interval: [1, messageCount]  // Ergo: message numbers are 1-based.  // Most servers give the latest message the highest number  for (int i = messageCount; i > 0; i--)  {  allMessages.Add(client.GetMessage(i));  }  // Now return the fetched messages  return allMessages;  }  }  Notice that the for loop starts at one, and goes up to and includes the messageCount number. This is because POP3 is 1-based. This is the case for all methods taking a message number as an argument. |

# ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

## Эксплуатационный раздел

### Сфера и специфика применения программного комплекса

Почтовый клиент предназначен для работы за ноутбуком или стационарным компьютером под управлением операционной системы Windows.

Допустимо использование в любой сфере деятельности, где есть нужда в общении посредством электронной почты.

Служит для отправки, получения и некоторого манипулирования с письмами.

Рассчитан на поддержку одной учетной записи пользователя. Однако возможно настроить переадресацию корреспонденции их разных аккаунтов в один ящик.

Программа не подходит твердо под офисный или домашний тип пользования. Ее можно использовать как для офиса, так и для дома.

Почтовый клиент является портативным (переносным), т.е. для работы на другой машине, в случае если пользователь хочет оставить свои контакты и сохранить учетные данные, достаточно скопировать три позиции: исполняемый файл с именем «MailV.exe», текстовый файл «Contacts.txt» и текстовый файл «Change.txt».

Для овладения программой не требуется специальных знаний. Уровень рассчитан на новичков. Пользоваться почтовым клиентом в праве тот, у кого уже есть зарегистрированный ящик на любом из почтовых сервисов, известен адрес электронный почты и логин / пароль от аккаунта.

## Раздел техники безопасности

### Техника безопасности при работе с ЭВМ

Персональный компьютер — электроприбор. От прочих электроприборов он отличается тем, что для него предусмотрена возможность длительной эксплуатации без отключения от электрической сети. Кроме обычного режима работы компьютер может находиться в режиме работы с пониженным электропотреблением или в дежурном режиме ожидания запроса. В связи с возможностью продолжительной работы компьютера без отключения от электросети следует уделить особое внимание качеству организации электропитания.

Недопустимо использование некачественных и изношенных компонентов в системе электроснабжения, а также их суррогатных заменителей: розеток, удлинителей, переходников, тройников. Недопустимо самостоятельно модифицировать розетки для подключения вилок, соответствующих иным стандартам. Электрические контакты розеток не должны испытывать механических нагрузок, связанных с подключением массивных компонентов (адаптеров, тройников и т. п.).

Все питающие кабели и провода должны располагаться с задней стороны компьютера и периферийных устройств. Их размещение в рабочей зоне пользователя недопустимо.

Запрещается производить какие-либо операции, связанные с подключением, отключением или перемещением компонентов компьютерной системы без предварительного отключения питания.

Компьютер не следует устанавливать вблизи электронагревательных приборов и систем отопления.

Недопустимо размещать на системном блоке, мониторе и периферийных устройствах посторонние предметы: книги, листы бумаги, салфетки, чехлы для защиты от пыли. Это приводит к постоянному или временному перекрытию вентиляционных отверстий.

Запрещается внедрять посторонние предметы в эксплуатационные или вентиляционные отверстия компонентов компьютерной системы.

Особенности электропитания монитора. Монитор имеет элементы, способные сохранять высокое напряжение в течение длительного времени после отключения от сети. Вскрытие монитора пользователем недопустимо ни при каких условиях. Это не только опасно для жизни, но и технически бесполезно, так как внутри монитора нет никаких органов, регулировкой или настройкой которых пользователь мог бы улучшить его работу. Вскрытие и обслуживание мониторов может производиться только в специальных мастерских.

Особенности электропитания системного блока

Все компоненты системного блока получают электроэнергию от блока питания. Блок питания ПК — это автономный узел, находящийся в верхней части системного блока. Правила техники безопасности не запрещают вскрывать системный блок, например, при установке дополнительных внутренних устройств или их модернизации, но это не относится к блоку питания. Блок питания компьютера — источник повышенной пожароопасности, поэтому вскрытию и ремонту он подлежит только в специализированных мастерских. Блок питания имеет встроенный вентилятор и вентиляционные отверстия. В связи с этим в нем неминуемо накапливается пыль, которая может вызвать короткое замыкание. Рекомендуется периодически (один - два раза в год) с помощью пылесоса удалять пыль из блока питания через вентиляционные отверстия без вскрытия системного блока. Особенно важно производить эту операцию перед каждой транспортировкой или наклоном системного блока.

Система гигиенических требований

Длительная работа с компьютером может приводить к расстройствам состояния здоровья. Кратковременная работа с компьютером, установленным с грубыми нарушениям гигиенических норм и правил, приводит к повышенному утомлению. Вредное воздействие компьютерной системы на организм человека является комплексным. Параметры монитора оказывают влияние на органы зрения. Оборудование рабочего места влияет на органы опорно-двигательной системы. Характер расположения оборудования в компьютерном классе и режим его использования влияет как на общее психофизиологическое состояние организма, так и им органы зрения.

Требования к видеосистеме

В прошлом монитор рассматривали в основном как источник вредных излучений, воздействующих, прежде всего на глаза. Сегодня такой подход считается недостаточным. Кроме вредных электромагнитных излучений (которые на современных мониторах понижены до сравнительно безопасного уровня) должны учитываться параметры качества изображения, а они определяются не только монитором, но и видеоадаптером, то есть всей видеосистемы в целом.

Монитор компьютера должен удовлетворять следующим международным стандартам безопасности:

* по уровню электромагнитных излучений — ТСО 95;
* по параметрам качества изображения (яркость, контрастность, мерцание, антибликовые свойства и др.) — ТСО 99.

Узнать о соответствии конкретной модели данным стандартам можно в сопроводительной документации. Для работы с мониторами, удовлетворяющими данным стандартам, специальные защитные экраны не требуется. На рабочем месте монитор должен устанавливаться таким образом, чтобы исключить возможность отражения от его экрана в сторону пользователя источников общего освещения помещения.

Расстояние от экрана монитора до глаз пользователя должно составлять от 50 до 70 см. Не надо стремиться отодвинуть монитор как можно дальше от глаз, опасаясь вредных излучений (по бытовому опыту общения с телевизором), потому что для глаза важен также угол обзора наиболее характерных объектов. Оптимально, размещение монитора на расстоянии 1,5 D от глаз пользователя, где D — размер экрана монитора, измеренный по диагонали. Сравните эту рекомендацию с величиной 3...5 D, рекомендованной для бытовых телевизоров, и сопоставьте размеры символов на экране монитора (наиболее характерный объект, требующий концентрации внимания) с размерами объектов, характерных для телевидения (изображения людей, сооружений, объектов природы). Завышенное расстояния от глаз до монитора приводит к дополнительному напряжению органов зрения, сказывается на затруднении перехода от работы с монитором к работе с книгой и проявляется в преждевременном развитии дальнозоркости.

Важным параметром является частота кадров, которая зависит от свойств монитора, видеоадаптера и программных настроек видеосистемы. Для работы с текстами минимально допустима частота кадров 72 Гц. Для работы с графикой рекомендуется частота кадров от 85 Гц и выше.

Требования к рабочему месту

В требования к рабочему месту входят требования к рабочему столу, посадочному месту (стулу, креслу), подставкам для рук и ног. Несмотря на кажущуюся простоту, обеспечить правильное размещение элементов компьютерной системы и правильную посадку пользователя чрезвычайно трудно. Полное решение проблемы требует дополнительных затрат, сопоставимых по величине со стоимостью отдельных узлов компьютерной системы, поэтому и в быту и на производстве этими требованиями часто пренебрегают. Несмотря на то, что школьники проводят в компьютерном классе сравнительно немного времени, обучить их правильной гигиене труда на достойном примере очень важно, чтобы полезные навыки закрепились на всю жизнь. Это не просто требование гигиены, а требование методики.

Монитор должен быть установлен прямо перед пользователем и не требовать поворота головы или корпуса тела.

Рабочий стол и посадочное место должны иметь такую высоту, чтобы уровень глаз пользователя находился чуть выше центра монитора. На экран монитора следует смотреть сверху вниз, а не наоборот. Даже кратковременная работа с монитором, установленным слишком высоко, приводит к утомлению шейных отделов позвоночника.

Если при правильной установке монитора относительно уровня глаз выясняется, что ноги пользователя не могут свободно покоиться на полу, следует установить подставку для ног, желательно наклонную. Если ноги не имеют надежной опоры, это непременно ведет к нарушению осанки и утомлению позвоночника. Удобно, когда компьютерная мебель (стол и рабочее кресло) имеют средства для регулировки по высоте. В этом случае проще добиться оптимального положения.

Клавиатура должна быть расположена на такой высоте, чтобы пальцы рук располагались на ней свободно, без напряжения, а угол между плечом и предплечьем составлял 100° — 110°. При использовании обычных письменных столов добиться одновременно правильного положения и монитора, и клавиатуры практически невозможно. Для работы рекомендуется использовать специальные компьютерные столы, имеющие выдвижные полочки для клавиатуры.

При длительной работе с клавиатурой возможно утомление сухожилий кистевого сустава. Известно тяжелое профессиональное заболевание — кистевой туннельный синдром, связанное с неправильным положением рук на клавиатуре. Во избежание чрезмерных нагрузок на кисть желательно предоставить рабочее кресло с подлокотниками, уровень высоты которых, замеренный от пола, совпадает с уровнем высоты расположения клавиатуры.

При работе с мышью рука не должна находиться на весу. Локоть руки или хотя бы запястье должны иметь твердую опору. Если предусмотреть необходимое расположение рабочего стола и кресла затруднительно, рекомендуется применить коврик для мыши, имеющий специальный опорный валик. Нередки случаи, когда в поисках опоры для руки (обычно правой) располагают монитор сбоку от пользователя (соответственно, слева), чтобы он работал вполоборота, опирая локоть или запястье правой руки о стол. Этот прием недопустим. Монитор должен обязательно находиться прямо перед пользователем.

Экран монитора — не единственный источник вредных электромагнитных излучений. Разработчики мониторов достаточно давно и успешно занимаются их преодолением. Меньше внимания уделяется вредным побочным излучениям, возникающим со стороны боковых и задней стенок оборудования. В современных компьютерных системах эти зоны наиболее опасны.  
Монитор компьютера следует располагать так, чтобы задней стенкой он был обращен не к людям, а к стене помещения. В компьютерных классах, имеющих несколько компьютеров, рабочие места должны располагаться по периферии помещения, оставляя свободным центр. При этом дополнительно необходимо проверить каждое из рабочих мест на отсутствие прямого отражения внешних источников освещения. Как правило, добиться этого для всех рабочих мест одновременно достаточно трудно. Возможное решение состоит в использовании штор на окнах и продуманном размещении искусственных источников общего и местного освещения.

Сильными источниками электромагнитных излучений являются устройства бесперебойного питания. Располагать их следует как можно дальше от посадочных мест пользователей. [9]

## Раздел экономических расчетов

### Расчет стоимости одного часа работы ПЭВМ

Стоимость одного часа работы ПЭВМ производится по формуле:

, (3.1)

где:

* – затраты на содержание и обслуживание ПЭВМ,
* – полезный фонд времени работы на ПЭВМ,
* – количество машин.

, (3.2)

где:

* – зарплата обслуживающего персонала,
* – страховые взносы, зачисляемые в государственные внебюджетные фонды,
* – накладные расходы,
* – амортизационные отчисления,
* – затраты на ремонт оборудования,
* – затраты на электроэнергию,
* – затраты на материалы.

Расчет зарплаты обслуживающего персонала

Таблица 3.1

Обслуживающий персонал

|  |  |  |  |
| --- | --- | --- | --- |
| Обслуживающий персонал | Заработная плата в месяц (руб.) | Количество работников (чел.) | Фонд заработной платы (руб.) |
| Оператор ЭВМ | 32000 | 1 | 32000 |
| ИТОГО | | | **32000** |

, (3.3)

, (3.4)

где:

* – заработная плата работника в месяц,
* – количество обслуживаемых работником машин,
* – количество машин, на которых установлена программа.

= 1 (шт.),

= 1 (шт.),

= (32000 / 1) \* 1 = 32000 (руб.),

= 32000 (руб.)

Так как дополнительная заработная плата отсутствует, то:

. (3.5)

Расчет страховых взносов, зачисляемых в государственные внебюджетные фонды

, (3.6)

где:

* – взносы на социальное страхование и обеспечение (%).

Таблица 3.2

Ставки страховых взносов, зачисляемых в государственные внебюджетные фонды

|  |  |
| --- | --- |
| Государственные внебюджетные фонды | Ставка налога (%) |
| Пенсионный фонд РФ | 22 |
| Фонд социального страхования РФ | 2,9 |
| Фонд обязательного медицинского страхования РФ | 5,1 |
| Страховые взносы на социальное страхование от несчастных случаев на производстве и профзаболеваний | 0,2 |
| **ИТОГО** | **30,2** |

= (32000 \* 30,2) / 100% = 9664 (руб.)

Расчет накладных расходов

, (3.7)

где:

* – ставка накладных расходов.

= 150

(32000 \* 150) / 100 = 48000 (руб.)

Расчет амортизационных отчислений

, (3.8)

где:

* – первоначальная стоимость оборудования,
* – ставка годовой нормы амортизации (%), = 12,5%.

Таблица 3.3

Состав оборудования

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование оборудования | Цена за единицу (руб.) | Количество единиц оборудования | Стоимость оборудования (руб.) |
| Компьютер | 19000 | 1 | 19000 |
| ИТОГО | | | **19000** |

= (19000 \* 12,5) / (100 \* 12) = 197,91 (руб.)

Расчет на ремонт оборудования

, (3.9)

где:

* – ставка годовых плановых затрат на ремонт оборудования (%).

= 10%,

= (19000 \* 10) / (100 \* 12) = 158,33 (руб.)

Расчет затрат на электроэнергию

, (3.10)

где:

* – общая мощность оборудования,
* – полезный фонд времени работы ПЭВМ,
* – цена 1 Квт/ч (руб.).

, (3.11)

где:

* – режимный фонд работы оборудования (час),
* – время на профилактику и ремонт оборудования (час).

, (3.12)

где:

* – количество календарных дней за отчетный период,
* – количество выходных дней за отчетный период,
* – количество праздничных дней за отчетный период,
* – количество предпраздничных дней за отчетный период,
* – длительность смены (час),
* – количество смен,
* – количество часов, на которые сокращается рабочий день в предпраздничные дни.

, (3.13)

где:

* – ставка плановых потерь на профилактику и ремонт оборудования (%).

Таблица 3.4

Общая мощность оборудования

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование оборудования | Мощность единицы оборудования (Квт/час) | Количество оборудования | Суммарная мощность (Квт/час) |
| Компьютер | 0,45 | 1 | 0,45 |
| ИТОГО | | | **0,45** |

Расчет ведется за май,

= 3,93,

= 10,

= 8 (часов),

= 2,

= 1 (час),

= (31 – 5 – 2) \* 8 \* 2 – 0 \* 1 = 384(часов),

= (384 \* 10) / 100 = 38,4 (часов),

= 384 – 38,4 = 345,6 (часов),

= 0,45 \* 345,6 \* 3,93 = 611,19 (руб.)

Расчет затрат на материалы

По данным бухгалтерии расходы на материалы составили 295 руб.

= 295.

Расчет стоимости часа работы на ПЭВМ

= 32000 + 9664 + 48000 + 197,91 + 158,33 + 611,19 + 295 = 90926,43 (руб.),

= 90926,43 / (345,6 \* 1) = 263,1(руб.)

Таблица 3.5

Структура стоимости 1 часа работы на ПЭВМ

|  |  |  |  |
| --- | --- | --- | --- |
| № | Элемент затрат | Сумма (руб.) | Удельный вес  (%) |
| 1 | Заработная плата обслуживающего персонала | 32000 | 35,19 |
| 2 | Страховые взносы, зачисляемые в государственные внебюджетные фонды | 9664 | 10,63 |
| 3 | Накладные расходы | 48000 | 52,79 |
| 4 | Амортизационные отчисления | 197,91 | 0,22 |
| 5 | Затраты на ремонт оборудования | 158,33 | 0,17 |
| 6 | Затраты на электроэнергию | 611,19 | 0,68 |
| 7 | Затраты на материалы | 295 | 0,32 |
| ИТОГО | | 90926,43 | **100,00** |

### Расчет затрат на составление и отладку программы

Расчет стоимости решения задачи на ПЭВМ

Стоимость решения задачи на ПЭВМ рассчитывается по формуле:

, (3.14)

где:

* – заработная плата руководителя,
* – заработная плата студента,
* – страховые взносы, зачисляемые в государственные внебюджетные фонды,
* – накладные расходы,
* – затраты на материалы, используемые при составлении и отладке программы,
* – затраты времени на отладку программы и решения ее на ПЭВМ.

Расчет заработной платы руководителя

, (3.15)

где:

* – время, отводимое на консультации по дипломному проектированию,
* – часовая тарифная ставка руководителя,
* – количество студентов у руководителя.

= 17 (часов),

= 300 (руб.),

= 1 (чел.),

= (17 \* 300) / 1 = 5100 (руб.)

Расчет заработной платы студента

, (3.16)

где:

* – стипендия студента,
* – количество учебных часов в месяце,
* – затраты времени на отладку программы и решение задачи на ПЭВМ.

, (3.17)

где:

* – средняя занятость в течении дня,
* – количество дней для отладки программы.

, (3.18)

где:

* Др – количество рабочих дней в месяце.

= 1000

= 6 (часа),

= 5 (дн.),

= 24 \* 6 = 144 (часов),

= 6 \* 5 = 30 (часов),

= (1000 \* 30) / 144 = 208,33 (руб.)

Расчет страховых взносов, зачисляемых в государственные внебюджетные фонды

, (3.19)

где:

* – взносы на социальное страхование и обеспечение (%).

= 30,2;

= (5100 \* 30,2) / 100 = 1540,20 (руб.)

Расчет накладных расходов

, (3.20)

где:

* – ставка накладных расходов.

= 150,

= ((5100 + 208,33) \* 150) / 100 = 7962,49 (руб.)

Затраты на материалы, используемые при составлении и отладке программы

Таблица 3.6

Затраты на материалы, используемые при составлении и отладке программы

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование материалов | Цена за единицу (руб.) | Количество | Стоимость (руб.) |
| Бумага | 1 | 100 | 100 |
| Ручки, карандаши и т.п. | 20 | 5 | 100 |
| Линейка | 15 | 1 | 15 |
| Калькулятор | 80 | 1 | 80 |
| ИТОГО | | | **295** |

= 295 (руб.),

= 5100 + 208,33 + 1540,20 + 7962,49 + 295 + 20 \* 263,1 = 20368,02 (руб.)

Расчет стоимости решения задачи вручную

Стоимость решения задачи вручную рассчитывается по формуле:

, (3.21)

где:

* – затраты времени на решение задачи вручную,
* – часовая ставка студента, решающего задачу (руб.).

, (3.22)

= 1000 / 144 = 6,95 (руб.),

= 8 (часа),

= 4 \* 6,95 \* (1 + (30,2 / 100) + (150 / 100)) = 155,8 (руб.)

### Расчет показателей эффективности применения ПЭВМ

Расчет экономии труда за счет применения ПЭВМ

, (3.23)

где:

* – предполагаемое число обращений к программе,
* – затраты времени на составление и отладку программы на ПЭВМ,
* – время решения задачи на ПЭВМ.

, (3.24)

где:

* – количество дней на составление и отладку программы на ПЭВМ.

, (3.25)

где:

* – время ввода исходной информации,
* – машинное время,
* – время вывода информации на печать.

= 0,02 (часа),

= 0,00043 (часа),

, = 0,33 (часа),

= 0,02 + 0,00043 + 0,33 = 0,35043(часа),

= 4 \* 30=120 (часов),

,

= (375 х 1) + (375 / 2) х 1 = 563,

= 8 \* 563 - (120 + 563 \* 0,35043) = 4186,71 (руб.)

Расчет денежной экономической эффективности применения ПЭВМ

, (3.26)

где:

* – затраты на работу программы.

, (3.27)

= 0,35043 \* 263,1 = 92,2 (руб.),

= 155,8 \* 563 – (20368,02 + 563 \* 92,2) = 15438,78 (руб.)

Расчет роста производительности труда от применения ПЭВМ

, (3.28)

= (4186,71 / (8 \* 563)) \* 100% = 92,96 %.

### Расчет затрат на разработку модуля

Затраты на разработку программного продукта рассчитываются по следующей формуле:

, (3.29)

* Где:
* – общий фонд оплаты труда разработчиков ПП;
* – начисления на заработную плату разработчиков ПП во внебюджетные фонды;
* – затраты, связанные с эксплуатацией техники;
* – затраты на специальные программные продукты, необходимые для разработки ПП;
* – затраты на хозяйственно-операционные нужды (бумага, литература, носители информации и т.п.);
* – накладные расходы (РН = 30% от ЗФОТР).

При разработке программного продукта общее время разработки составило 1 месяц. Из них машинное время (непосредственная работа с вычислительной и оргтехникой) составляет 0,5 мес.

Фонд оплаты труда за время работы над программным продуктом:

, (3.30)

где:

* – оклад *j*-го разработчика. В разработке участвовал 1 человек, его оклад составляет 3000 руб.;
* – общее время работы над ПР в месяцах, ;
* – коэффициент дополнительной зарплаты, = 0,1;
* – районный коэффициент, = 0,15.

= 3000 \* 1 \* (1 + 0,1) \* (1 + 0,15) = 3795 руб.

Отчисления во внебюджетные фонды складываются из ЕСН и взносов на обязательное социальное страхование от несчастных случаев на производстве и профессиональных заболеваний.

Так как годовой фонд заработной платы разработчика не превышает 415 тыс. руб., то используются максимальные ставки ЕСН. Ставка страхования от несчастных случаев в соответствии с классом профессионального риска составляет 0,2 %. Значения всех используемых ставок приведены в табл. 3.7

Таблица 3.7

Значения ставок ЕСН и СНС

|  |  |  |
| --- | --- | --- |
| № | Наименование внебюджетного фонда | Размер ставок, % |
| 1. | Федеральный бюджет | 6% |
| 2. | Пенсионный фонд | 14% |
| 3. | Фонд социального страхования | 3,2% |
| 4. | Федеральный фонд обязательного медицинского  страхования | 0,8% |
| 5. | Территориальный (региональный) фонд обязательного медицинского страхования | 2% |
| 6. | Страховой тариф на обязательное социальное страхование от несчастных случаев на производстве и профзаболеваний (для 2 класса профессионального риска) | 0,2 |
|  | ИТОГО | 26% |

Сумма начислений на заработную плату во внебюджетные фонды составляет:

, (3.31)

= 0,262 \* 3795 = 994,29 руб.

Затраты, связанные с использованием вычислительной и оргтехники:

, (3.32)

где:

* – коэффициент готовности ЭВМ, = 0,95;
* – количество единиц техники, равно 1;
* – машинное время работы над программным продуктом, равно 0,5 мес.

Перевод рабочего времени в часы осуществляется по формуле:

, (3.33)

где:

* – рабочее время, ч,
* – рабочее время, мес., ( = 1),
* – число рабочих дней, ( = 22),
* *Тсм* – продолжительность рабочей смены, (*Тсм* = 4 ч),
* – количество рабочих смен, ( = 1).

Таким образом, время на разработку ПП с использованием ЭВМ составляет:

= 1 \* 22 \* 4 \* 1 = 88 часов,

= 88 \* 0,95 \* 1 \* 263,1 = 21995,16 руб.

Затраты на специальные программные продукты, необходимые для разработки ПП рассчитываются по формуле:

, (3.34)

где:

* – цена -го специального программного продукта.

Перечень программных продуктов специального назначения приведен в табл. 3.8

Таблица 3.8

Программные продукты специального назначения

|  |  |  |
| --- | --- | --- |
| №п/п | Название ПП | Цена, руб. |
| 1 | Visual Studio Professional 2013 (студенческая) | 0 |
|  | Итого | 0 |

= 0 рублей.

Затраты на хозяйственно-организационные нужды приведены в табл. 3.9 и вычисляются по формуле:

, (3.35)

где:

* – цена -го товара, руб.,
* – количество -го товара.

Таблица 3.9

Затраты на хозяйственно-организационные нужды

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Цена за единицу (руб.) | Кол-во (шт.) | Всего (руб.) |
| Диск CD-R | 25 | 3 | 75 |
| Бумага | 1 | 138 | 138 |
| Итого |  |  | 213 |

= 25 \* 3 + 1 \* 138 = 213 руб.

Накладные расходы:

, (3.36)

= 3795 \* 0,3 = 1138,5 руб.

Таким образом, затраты на разработку программного продукта составят:

= 3795 + 994,29 + 21995,16 + 0 + 213 + 1138,5 = 28135,95 руб.

### Расчет Интернет-трафика в сравнении с веб-почтой

Для анализа была выбрана простая программа для перехвата сетевого трафика – NetWorx. Она показывает, сколько получено и отправлено информации в килобайтах (Кб). Способна отслеживать работу приложений. В данном случае почтового клиента «MailV».

В исследовании будет участвовать отправка сообщения с одинаковым содержимым, а так же загрузка входящих писем.

Пусть тема письма – «Тест №1», тело письма – «Анализ Интернет-трафика». К письму прикреплен файл в виде картинки, размер которой равен 0,97 Мб.

Результаты расхода трафика в веб-интерфейсе показаны в таблице 4.0

Результаты расхода трафика в почтовом клиенте показаны в таблице 4.1

Таблица 4.0 Расход трафика при отправке письма под средством веб-интерфейса

|  |  |
| --- | --- |
| Отправка письма (веб-почта) | |
| Получено: | 67,2 Кб |
| Отправлено: | 1,06 Мб |
| Всего: | 1,13 Мб |

Таблица 4.1 Расход трафика при отправке письма под средством почтового клиента

|  |  |
| --- | --- |
| Отправка письма (почтовый клиент) | |
| Получено: | 4,12 Кб |
| Отправлено: | 1,10 Мб |
| Всего: | 1,10 Мб |

Рассмотрим расход на загрузку 285 писем в веб-интерфейсе в таблице 4.2, а расход в почтовом клиенте в таблице 4.3.

Таблица 4.2. Загрузка входящих писем в веб-интерфейсе

|  |  |
| --- | --- |
| Загрузка входящих писем (веб-почта) | |
| Получено: | 0,96 Мб |
| Отправлено: | 122 Кб |
| Всего: | 1,08 Мб |

Таблица 4.3. Загрузка входящих писем в почтовом клиенте

|  |  |
| --- | --- |
| Загрузка входящих писем (почтовый клиент) | |
| Получено: | 39,3 Кб |
| Отправлено: | 1,17 Кб |
| Всего: | 40,4 Кб |

Примечание. Во время загрузки в почтовом ящике содержится 20 писем.

Таблица 4.1. показывает, что при отправлении письма трафик одинаков, однако почтовый клиент выигрывает в получении. За счет отсутствия рекламы и другой лишней информации в почтовом клиенте было израсходовано в 16 раз меньше трафика.

Таблица 4.3. показывает, что загрузка 20 писем из ящика требует в 2 раза меньше сетевого трафика.

### Вывод

Стоимость разработки почтового клиента составила 28135,95 руб. Использование электронной почты позволяет не тратить время на покупку конверта и марок. Письма доставляются моментально в любую точку планеты. И всегда можно проверить дошло письмо до получателя или нет.

Анализ Интернет-трафика эксперимента показал, что использование почтового клиента по сравнению с веб-почтой значительно сокращает расход траффика. Таким образом почтовый клиент данной дипломной работы позволяет уменьшить затраты на использование лимитного Интернета.

# ЗАКЛЮЧЕНИЕ

В данном дипломном проекте была выполнена разработка почтового клиента (пользовательского агента), обладающего основными функциями данной категории программного обеспечения.

Объектом изучения стал принцип работы электронной почтовой системы. Как взаимодействуют между собой почтовый клиент и сервер получателя, функционируют основные протоколы передачи данных в Интернете. Произведя анализ существующих почтовых клиентов, я сделал соответствующие выводы, что помогло мне в разработке дипломной работы.

Программа позволяет принимать входящие письма, обеспечивает их просмотр, автоматизирует создание ответных сообщений, имеет поддержку адресной книги, умеет скачивать прикрепления и многое другое.

Основной особенностью данного почтового клиента является: организация скаченных вложений и значительное сокращение сетевого трафика при загрузке входящей корреспонденции за счет отсутствия рекламы и обильного графического оформления.

Отправка сообщений осуществляется протоколом SMTP, а прием сообщений протоколом POP версии 3.

Программа обладает дружественным интерфейсом. Чтобы ею пользоваться не требуется владеть специальными знаниями.

Разработка производилась в среде Microsoft Visual Studio, которая позволяет создавать приложения на профессиональном уровне. Среда обладает богатым набором инструментов для программиста.

Трудность разработки данного программного комплекса заключалась в том, что стандартная библиотека платформы .NET не содержит собственных средств для получения входящих писем по протоколам IMAP и/или POP. Таким образом, потребовалось найти или разработать библиотеку самостоятельно.

В дальнейшем планируется наращивание функционала, минимизируя при этом расход сетевого трафика.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* + - 1. Компьютерные сети. Э. Таненбаум, Д. Уэзеролл. 5-ое издание
      2. http://hpop.sourceforge.net/
      3. https://www.microsoft.com/ru-ru/default.aspx
      4. http://www.white-windows.ru/history-created-email/
      5. http://www.icmm.ru/~masich/win/lexion/mail/pop3.html
      6. https://www.ritlabs.com/ru/
      7. https://www.microsoft.com/ru-ru/outlook-com/
      8. https://www.dreamspark.com/
      9. http://ohrana-tryda.com/
      10. http://www.flexerasoftware.com/

# ПРИЛОЖЕНИЕ 1. Руководство пользователя

*Развертывание программы.* Прежде, чем начать использовать программу, требуется сперва ее установить. В этом поможет установщик InstallShield, который самостоятельно установит все необходимые компоненты.

Чтобы запустить установщик нажмите двойным нажатием по ярлыку «setup.exe», setup.exe – это настройщик запуска (setup launcher). Ярлык показан на рисунке 1.1.

C:\Users\Insomnia\Desktop\картинки\установщик.png

Рисунок 1.1. Ярлык запуска установщика

После запуска следуйте инструкциям, который предлагает установщик. Особых разъяснений не требуется, действия выполняются на интуитивном уровне.

Загрузка показана на рисунке 1.2.

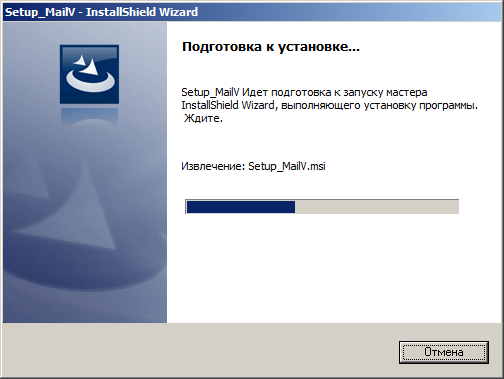


Рисунок 1.2. Подготовка к установке

После завершения загрузки появится окно приветствия, как на рисунке 1.3.

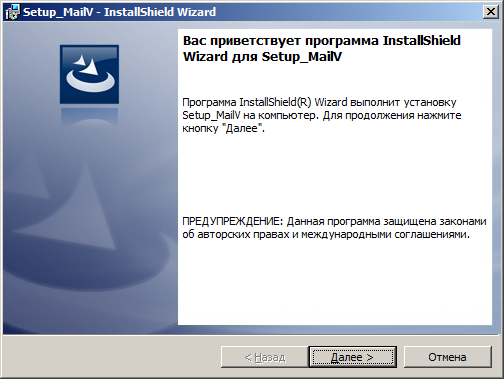


Рисунок 1.3. Приветствие установки

Нажмите кнопку далее.

После появится окно выбора папки назначения для установки всех компонентов. Окно показано на рисунке 1.4. Чтобы изменить стандартный путь нажмите кнопку «Изменить…» и выберите предпочитаемое место назначения.

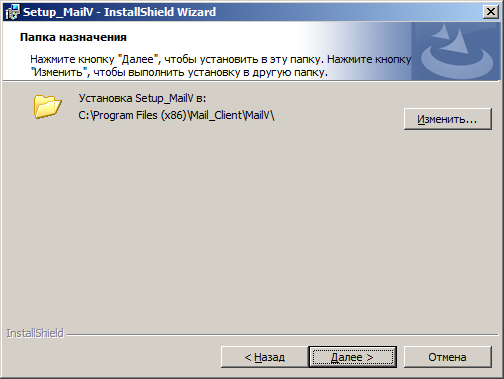


Рисунок 1.4. Папка назначения

После выбора назначения, нажмите далее.

Откроется окно, как на рисунке 1.5., показывающее информацию об установке.

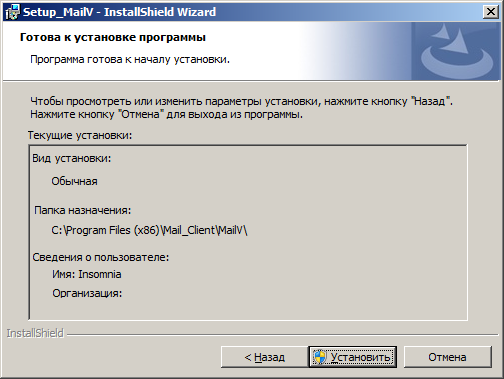


Рисунок 1.5. Информация и сообщение о готовности к установке

Нажмите кнопку «Установить». Потребуются права администратора. Нажмите да, чтобы подтвердить установку. Чтобы изменить папку назначения нажмите кнопку «< Назад» и установщик вернет вас на окно выбора папки.

Далее при успешном завершении работы установки появится окно, как на рисунке 1.6. Нажмите кнопку «Готово» для выхода из установщика.

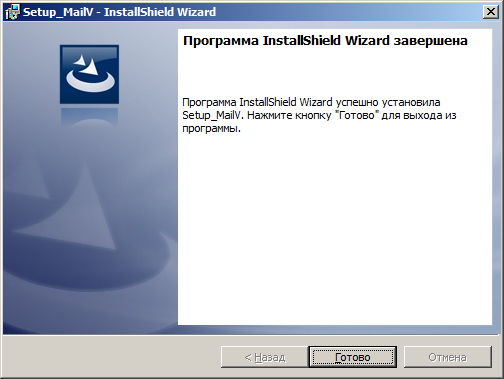


Рисунок 1.6. Завершение установки

После завершения установки на рабочем столе появится ярлык запуска программы «MailV». Ярлык показан на рисунке 1.7.



Рисунок 1.7. Ярлык почтового клиента на рабочем столе

Если программа уже установлена на компьютер, то при повторном запуске установщика будет доступно окно, показанное на рисунке 1.8.

На нем есть три функции: изменить, исправить, удалить. Описание этих функций можно посмотреть напротив каждого пункта.

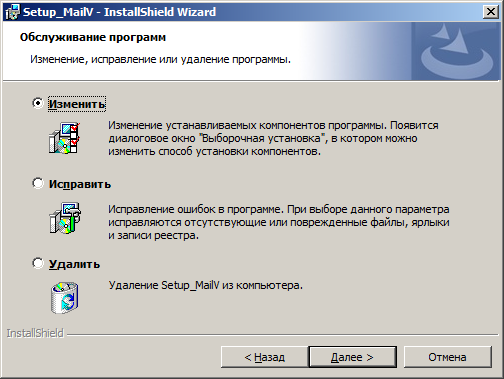


Рисунок 1.8. Обслуживание программы

В той директории, которую вы выбрали для установки, создастся каталог «Mail\_Client» в нем папка «MailV», а в папке два файла: сама программа - «MailClient.exe» и библиотека «OpenPop.dll», в ней же будут создаваться папки для хранения вложений от получателей и второстепенные файлы: «Change.txt» и «Contacts.txt». В «Change.txt» хранятся данные для авторизации пользователя, а в «Contacts.txt» - контакты пользовательского аккаунта.

Необходимо хранить все файлы в одном каталоге. В начале об этом уже позаботился установщик, просто надо придерживаться такой организации файлов и каталогов.

Программа установлена, все компоненты, которых не хватало для корректного выполнения - тоже установлены. Теперь приступим в работе с самой программой.

*Первый запуск почтового клиента.*

Первый запуск отличается от последующих тем, что почтовый клиент еще с не знаком с пользователем, поэтому выводится окно приветствия, оно показано на рисунке 1.9., затем приглашает на регистрацию (рисунок 1.10.).

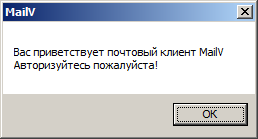


Рисунок 1.9. Привет пользователь!

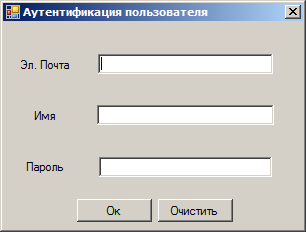


Рисунок 1.10. Авторизация

В поле «Эл. Почта» вводится полный адрес электронной почты. Например, ivanov@corp.ru

В поле «Имя» вводится имя пользователя, которое будут видеть получатели в графе «От кого»

В поле «Пароль» вводится пароль от почтового ящика, которое указывалось при регистрации на почтовом сервисе. Примечание: Некоторые почтовые сервисы просят генерировать отдельные пароли для сторонних приложений. Сделать это можно в настройках того сервиса, на котором зарегистрирован почтовый ящик.

Так же нужно учесть еще один нюанс при авторизации. Заключается он в том, что не все почтовые сервисы разрешают скачивать письма по протоколам imap и pop. Обычно по умолчанию эта функция запрещает скачивать входящие письма. В настройках требуется разрешить доступ к почте сторонним клиентам.

После заполнения всех полей, нажимаем на кнопку «Ок». В дальнейшем авторизация будет проходить автоматически. Если ящик зарегистрирован на mail.ru, rambler.ru, gmail.com или yandex.ru с любым доменом, больше действий при регистрации от пользователя не требуется. В противном случае программа попросит ввести самостоятельно названия серверов smtp и pop3. Названия можно узнать в справке почтового сервиса, на котором зарегистрирован адрес. Окно заполнения адресов показано на рисунке 1.11.

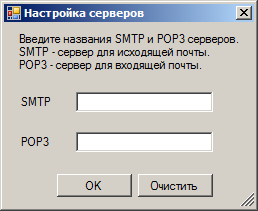


Рисунок 1.11. Заполнение серверов smtp и pop3

После успешной регистрации откроется главное окно рабочей области почтового клиента. Рассмотрим его на рисунке 1.12.

Вверху окна находится панель инструментов . Она состоит из: Написать письмо, Контакты, Настройка. Ниже расположена область с именем и адресом пользователя. В данном случае имя – [Insomnia], адрес – unlucky\_v@mail.ru. В этой же области ниже есть кнопка «Входящие» и надпись «Всего:» - здесь будет указываться общее количество входящих сообщений. Еще ниже расположена таблица, поля которой следующие: «От кого», «Тема», «Дата». В правой части окна расположена кнопка «Трей» и пустая область, для показа открытого сообщения. В системном меню расположены кнопки «справка» и «закрыть». Остальные кнопки, надписи и так далее могут быть не видны или не доступны в связи с ненадобностью или ограничением интернета. Сделано это для того, чтобы пользователь знал наверняка какие функции ему доступны в данный момент.

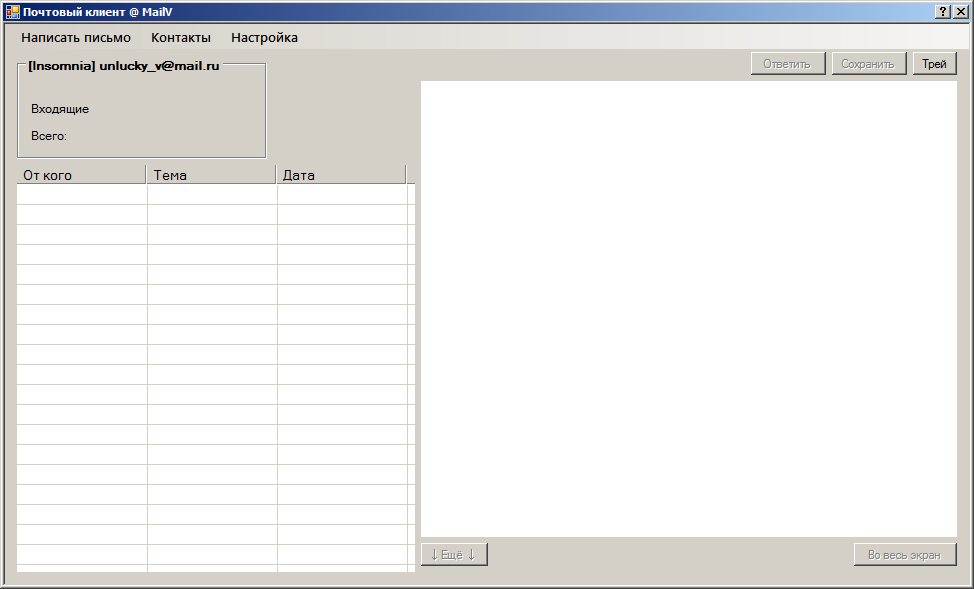


Рисунок 1.12. Главное окно, авторизированного пользователя

*Написание и отправка сообщений.*

Чтобы написать сообщение сначала нужно выбрать пункт «Написать сообщение» на панели инструментов или открыть его сочетанием клавиш Ctrl + N. Окно показано на рисунке 1.13. Рассмотрим его содержание.

В поле «Кому:» указывается адрес получателя. Адресов может быть сколь угодно много. При многочисленной отправке адреса записываются в треугольных скобках (<>). Так же может указываться имя получателя. Оно записывается в двойных кавычках (“”). При наличии зафиксированных контактов в адресной книге, адреса автоматически переносятся в раскрывающийся список для выбора получателя.

В поле «Сс:» указывается кому из получателей отошлется копия письма. Технически поле «Кому» и поле «Сс:» не отличаются. Разве что, при просмотре письма получатели принявшие копию почты будут находиться в графе «Копии» в почтовом сервисе. Различия только психологические.

В поле «Всс:» указывается кому из получателей отправится скрытая копия письма. Это поле в отличие от обычной копии, другие получатели не увидят.

В поле «Тема:» указывается тема письма. Она может быть пустой.

Приоритет письма. Поставив галочку напротив пункта «Важное» отправитель назначает письму наивысший приоритет. Это значит, что письмо в интерфейсе будет помечено восклицательным знаком. И получатель обратит на это внимание.

В поле «Сообщение:» заносится текст письма. Текст может содержать html вставки.

К письму можно прикрепить файл, нажав на кнопку со скрепкой. Файл должен быть один. Если требуется прикрепить группу файлов, предварительно необходимо соединить в единый архив. После прикрепления выведется информация о том, что файл прикреплен, его имя с расширением и размер.

Отправляем письмо нажатием на кнопку отправки со стрелкой вверх. При успешной отправке выведется соответствующее сообщение.

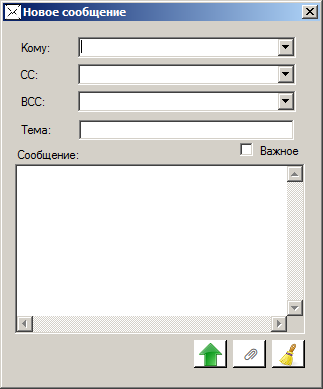


Рисунок 1.13. Составление нового письма на отправку

*Контакты.*

Чтобы перейти в адресную книгу выберите пункт «Контакты» на панели инструментов или откройте ее сочетанием клавиш Ctrl + K. Рассмотрим окно на рисунке 1.14. Контакты будут доступны пользователю всегда даже при отсутствии Интернет-соединения. Адреса хранятся в файле, поэтому при выходе из программы они все загрузятся вновь, и будут доступны все действия с ними.

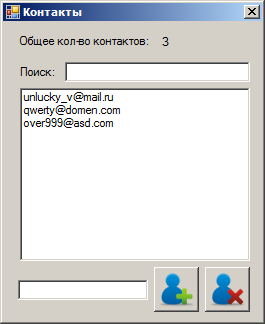


Рисунок 1.14. Контакты

Сверху распложено общее кол-во контактов, которое обновляется в реальном времени. Ниже присутствует поле поиск. Под ним список контактов. Далее поле для добавления нового контакта. Справа в нижнем углу две кнопки: добавить контакт и удалить его.

Чтобы воспользоваться поиском, введите первые буквы или полный адрес. В случае совпадения предполагаемый адрес выделится. В противном случае программа проигнорирует ввод.

Чтобы добавить контакт введите адрес в нижнем поле напротив кнопок, и нажмите «добавить».

Чтобы удалить контакт или контакты следует выделить один или несколько контактов и нажать на кнопку «удалить». Есть возможность выделять один или несколько контактов. Чтобы выделить несколько контактов, зажмите клавишу ctrl и далее ЛКМ по адресу.

Когда все операции с контактами завершены, закройте окно на крестик. Контакты автоматически перенесутся в список получателей, когда вы будете отправлять новое сообщение.

*Настройка.*

Чтобы перейти в настройки программы нажмите на пункт «Настройка» на панели инструментов или откройте настройки сочетанием клавиш ctrl + J.

В настройках существует только одна функция – смена пользователя. Смена пользователя осуществляется, абсолютно так же как и при регистрации.

*Работа с входящими письмами.*

Чтобы загрузить входящие письма, требуется нажать на кнопку «Входящие», расположенной на главном окне программы. Письма формируются в список, показанный на рисунке 1.15.

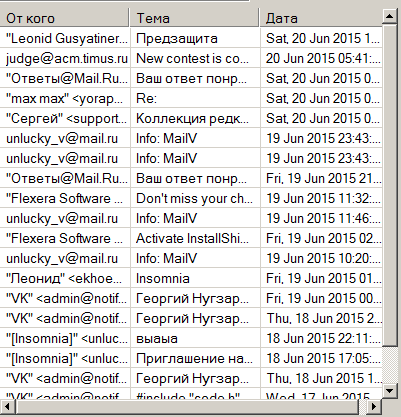


Рисунок 1.15. Список входящих

Список состоит не больше чем из 20 последних сообщений. Соответственно самое свежее письмо находится вверху списка. Если не видно, что содержится в одном из столбцов можно потянуть за ползунок столбца влево или вправо.

Чтобы удалить письмо, вызовите контекстное меню путем нажатия на письмо правой кнопкой мыши. Далее нажмите удалить. Строка исчезнет, однако письмо удалиться с сервера только после выхода из программы. Это особенности протокола pop3. Он только пометить письмо, как удаленное, а удалит его только после успешного высвобождения ресурсов.

Чтобы просмотреть письма, следующие дальше по списку, нажмите на кнопку «Еще», расположенную справа от списка входящих писем. Программа загрузит еще 20 писем, если это возможно, выделит первое письмо из этого нового списка и перенесет фокус в новую область. Если писем больше нет, программа оповестит об этом пользователя.

Чтобы открыть содержание письма, кликните левой кнопкой мыши по интересующему письму. Поле «От кого» выделится и в правой области окна начнется отображаться текст письма и т.п.

В зависимости от содержания письма, могут появляться соответствующие оповещения о том, что хранится в конверте. Например, кому посланы копии данного письма, есть ли вложения и какие они. Рассмотрим пример письма с html вставкой, в котором есть копии и прикрепление в виде картинки с расширением jpg на рисунке 1.16

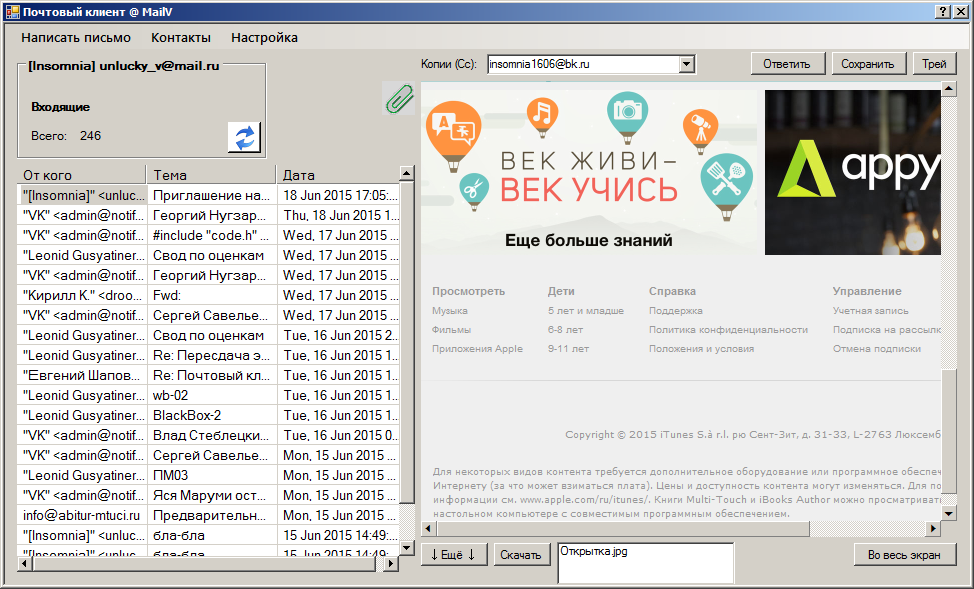


Рисунок 1.16. Полноценное входящее письмо

Чтобы ответить отправителю письма нажмите на кнопку «Ответить». Откроется окно отправки сообщений и в поле «Кому» автоматически вставится адрес отправителя.

Чтобы сохранить письмо, нажмите на кнопку «Сохранить». Откроется диалоговое окно, предлагающее выбрать место сохранения файла. Файл может быть назван любым именем. Согласно правилам имя не должно содержать запрещенные символы. Так же и тип сохраняемого письма может быть любым. Рекомендуется текстовые сообщения сохранять как .txt, письма с html кодом сохранять как .html страницу, а xml письма сохранять как .xml.

Для удобного просмотра сообщений имеется кнопка «Во весь экран», нажимая на нее сообщение, открывается в новом окне во весь пользовательский экран.

Письмо открывается в компоненте, который находится под управлением браузера Internet Explorer. Поэтому доступно контекстное меню этого браузера. В нем есть много полезных функций, например: печать письма на принтере, экспорт в excel и др.

Чтобы проверить наличие новых сообщений, следует нажать на кнопку обновления. При наличии новых писем, программа оповестит о количестве. В противном случае программа проигнорирует обновление.

# ПРИЛОЖЕНИЕ 2. Листинги программы

Листинг 1.1. MainForm.cs – Главное окно

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.Net.Mail;  using System.Net;  using System.Net.NetworkInformation;  using System.IO;  using OpenPop;  using OpenPop.Common;  using OpenPop.Common.Logging;  using OpenPop.Mime;  using OpenPop.Mime.Decode;  using OpenPop.Mime.Header;  using OpenPop.Mime.Traverse;  using OpenPop.Pop3;  using OpenPop.Pop3.Exceptions;  using Message = OpenPop.Mime.Message;  //Главная форма  namespace MailClient  {  public partial class MainForm : Form  {  private Pop3Client client;  public MainForm()  {  InitializeComponent();  client = new Pop3Client();  }  string fpath = "Change.txt";  string From;  Message message;  //Открыть окно "Новое сообщение"  private void новоеСообщение\_Click(object sender, EventArgs e)  {  NewMessage fr2 = new NewMessage();  fr2.Show();  }  //Открыть окно "Контакты"  private void контакты\_Click(object sender, EventArgs e)  {  Contacts fr3 = new Contacts();  fr3.Show();  }  //Открыть окно "Настройки"  private void настройка\_Click(object sender, EventArgs e)  {  Settings fr4 = new Settings();  fr4.Show();  }  //Справка  private void MainForm\_HelpButtonClicked(object sender, CancelEventArgs e)  {  MessageBox.Show("Разработчик: Валиев Иван\nСвязаться с автором: unlucky\_v@mail.ru", "Справка", 0, MessageBoxIcon.Information);  }  //Проверка интернет соединения  private bool internetconnection()  {  IPStatus status = IPStatus.Unknown;  try  {  status = new Ping().Send("smtp.mail.ru").Status;  }  catch { }  if (status == IPStatus.Success)  {  return true;  }  else  {  return false;  }  }  //Соединиться с сервером  public void userpopconnect()  {  string username, password, pop3host;  bool ssl = true;  int port;  string[] change = File.ReadAllLines(fpath);  if (change.Length != 0)  {  username = change[0];  password = change[1];  pop3host = change[3];  port = 995;  try  {  if (client.Connected)  client.Disconnect();  client.Connect(pop3host, port, ssl);  client.Authenticate(username, password,AuthenticationMethod.Auto);  }  catch (Exception ex)  {  MessageBox.Show("Ошибка в подключении пользователя\nТекст ошибки" + ex.Message);  }  }  else  {  MessageBox.Show("Авторизуйтесь пожалуйста");  }  }  //Функция получения входящих писем с сервера  private void GetAllMess()  {  try  {  int count = client.GetMessageCount();  label1.Text = count.ToString();  MessageHeader headers;  RfcMailAddress from;  string subject, date;  for (int i = count; i > 0; i--)  {  headers = client.GetMessageHeaders(i);  from = headers.From;  subject = headers.Subject;  date = headers.Date;  ListViewItem lv = new ListViewItem(from.ToString());  lv.SubItems.Add(subject);  lv.SubItems.Add(date);  listView1.Items.Add(lv);  progressBar1.Value++;  if (listView1.Items.Count >= 20)  break;  }  progressBar1.Value = 0;  progressBar1.Visible = false;  }  catch (Exception ex)  {  MessageBox.Show("Не удалось загрузить входящие сообщения\n" + ex.Message);  }  }  //Загрузка формы главного окна  private void MainForm\_Load(object sender, EventArgs e)  {  DownloadB.Visible = false;  AttachBox.Visible = false;  pictureBox1.Visible = false;  SaveB.Enabled = false;  ShowB.Enabled = false;  label4.Visible = false;  Answer.Enabled = false;  comboBox1.Visible = false;  progressBar1.Visible = false;  UpdateB.Visible = false;  if (listView1.Items.Count == 0)  More.Enabled = false;  if (internetconnection() == false)  {  MessageBox.Show("Нет соединения c интернетом!\nФункционал программы ограничен.", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Warning);  новоеСообщениеToolStripMenuItem.Enabled = false;  настройкаToolStripMenuItem.Enabled = false;  InBox.Enabled = false;  label1.Visible = false;  More.Enabled = false;  }  else  {  if (!File.Exists(fpath))  {  File.CreateText(fpath);  MessageBox.Show("Вас приветствует почтовый клиент MailV\nАвторизуйтесь пожалуйста!", "MailV");  ChangeUser fr5 = new ChangeUser();  fr5.ShowDialog();  }  else  {  string[] mailadr = File.ReadAllLines(fpath);  if (mailadr.Length != 0)  {  this.groupBox1.Text = mailadr[4] + " " + mailadr[0];  userpopconnect();  }  else  {  MessageBox.Show("Вас приветствует почтовый клиент MailV\nАвторизуйтесь пожалуйста!", "MailV");  ChangeUser fr5 = new ChangeUser();  fr5.ShowDialog();  }  }  }  }  //Нажатие на кнопку "Входящие"  private void InBox\_Click(object sender, EventArgs e)  {  if (listView1.Items.Count == 0)  {  InBox.Font = new Font("Microsoft Sans Serif", 8F, FontStyle.Bold);  progressBar1.Visible = true;  progressBar1.Value = 0;  GetAllMess();  UpdateB.Visible = true;  More.Enabled = true;  }  }  //Наведение курсора  private void InBox\_MouseEnter(object sender, EventArgs e)  {  this.InBox.Font = new Font("Microsoft Sans Serif", 10);  }  //Отдаление курсора  private void InBox\_MouseLeave(object sender, EventArgs e)  {  if (!(InBox.Font.Bold))  this.InBox.Font = new Font("Microsoft Sans Serif", 9);  }  //Нажатие по письмо ПКМ  private void listView1\_MouseClick(object sender, MouseEventArgs e)  {  if (e.Button == MouseButtons.Right)  {  if (listView1.FocusedItem.Bounds.Contains(e.Location) == true)  {  contextMenuStrip1.Show(Cursor.Position);  }  }  }  //Открытие выбранного письма  private void listView1\_ItemActivate(object sender, EventArgs e)  {  this.SaveB.Enabled = true;  this.ShowB.Enabled = true;  this.Answer.Enabled = true;  AttachBox.Items.Clear();  if (comboBox1.Items.Count != 0)  this.comboBox1.Items.Clear();  try  {  //Количество сообщений  int count = client.GetMessageCount();  //Индекс выбранного письма  int index = listView1.SelectedIndices[0];  int copy\_count;  message = client.GetMessage(count - index);  MessageHeader headers = client.GetMessageHeaders(count - index);  RfcMailAddress from;  from = headers.From;  From = from.ToString();  copy\_count = headers.Cc.Count;  int countattach = message.FindAllAttachments().Count;  if (countattach != 0)  {  pictureBox1.Visible = true;  DownloadB.Visible = true;  AttachBox.Visible = true;  foreach (MessagePart attachment in message.FindAllAttachments())  {  AttachBox.Items.Add(attachment.FileName);  }  }  else  {  DownloadB.Visible = false;  AttachBox.Visible = false;  pictureBox1.Visible = false;  }  if (copy\_count != 0)  {  label4.Visible = true;  comboBox1.Visible = true;  comboBox1.Enabled = true;  for (int j = 0; j < copy\_count; j++)  comboBox1.Items.Add(headers.Cc[j].ToString());  }  else  {  label4.Visible = false;  comboBox1.Visible = false;  }  //Просто текст  MessagePart plainTextPart = message.FindFirstPlainTextVersion();  if (plainTextPart != null)  {  webBrowser1.DocumentText = plainTextPart.GetBodyAsText();  }  //HTML Вставка  MessagePart html = message.FindFirstHtmlVersion();  if (html != null)  {  //без создания файла  //webBrowser1.DocumentText = html.GetBodyAsText(); //работает вроде бы медленнее и может не открыть с первого раза  //с файлом  //1-ый вариант  html.Save(new FileInfo("htmlmessage.html"));  string curDir = Directory.GetCurrentDirectory();  this.webBrowser1.Url = new Uri(String.Format("file:///{0}/htmlmessage.html", curDir));  //2-ой вариант  //webBrowser1.Navigate("file:///C:/Проекты/MailClient/MailClient/bin/Debug/htmlmessage.html");  }  //XML Вставка  MessagePart xml = message.FindFirstMessagePartWithMediaType("text/xml");  if (xml != null)  {  string xmlString = xml.GetBodyAsText();  System.Xml.XmlDocument doc = new System.Xml.XmlDocument();  doc.LoadXml(xmlString);  doc.Save("Xml.xml");  }  }  catch (Exception ex)  {  MessageBox.Show("Попробуйте еще раз!\n" + ex.Message);  }  }  //Удалить письмо  private void удалить\_Click(object sender, EventArgs e)  {  //при удалении количество сообщений меняется, но само сообщение удалится только после дисконекта или dispose  try  {  int count = client.GetMessageCount();  int index = listView1.SelectedIndices[0];  client.DeleteMessage(count - index);  listView1.Items.RemoveAt(index);  label1.Text = client.GetMessageCount().ToString();  }  catch (Exception ex)  {  MessageBox.Show("Попробуйте еще раз!\n" + ex.Message);  }  }  //Сохранить письмо  private void Save\_Click(object sender, EventArgs e)  {  Stream s;  string mess = webBrowser1.DocumentText;  SaveMessageDialog.FileName = String.Empty;  if (SaveMessageDialog.ShowDialog() == DialogResult.OK)  {  if ((s = SaveMessageDialog.OpenFile()) != null)  {  using (StreamWriter sw = new StreamWriter(s))  {  sw.Write(mess);  sw.Close();  }  s.Close();  }  }  }  //Ответить отправителю  private void Answer\_Click(object sender, EventArgs e)  {  NewMessage fr2 = new NewMessage();  fr2.Show();  fr2.comboBox1.Text = From;  }  //нажатие кнопки Трей  private void Tray\_Click(object sender, EventArgs e)  {  //скрыть окно для пользователя  this.Hide();  notifyIcon1.Visible = true;  notifyIcon1.BalloonTipText = "MailV скрыт в трей";  //показать на 1 секунду всплывающую подсказку  notifyIcon1.ShowBalloonTip(1000);  }  //нажатие ярлыка в области уведомлений  private void notifyIcon1\_MouseClick(object sender, MouseEventArgs e)  {  notifyIcon1.Visible = false;  this.Show();  InBox.Focus();  }  //Показать письмо целиком  private void ShowFullMess\_Click(object sender, EventArgs e)  {  FullMessage fr6 = new FullMessage();  fr6.Show();  fr6.webBrowser1.DocumentText = webBrowser1.DocumentText;  }  //Закрытие формы  private void MainForm\_FormClosing(object sender, FormClosingEventArgs e)  {  try  {  client.Dispose();  }  catch (Exception ex)  {  MessageBox.Show("Ошибка в освобождении ресурсов:\nТекст ошибки: " + ex.Message);  }  }  //Обновить входящие письма  private void UpdateB\_Click(object sender, EventArgs e)  {  try  {  int countold = client.GetMessageCount();  if (listView1.Items.Count != 0)  listView1.Items.Clear();  progressBar1.Visible = true;  client.Disconnect();  userpopconnect();  GetAllMess();  int countnew = client.GetMessageCount();  if (countold < countnew)  MessageBox.Show("У вас есть новое(ые) сообщение(ия)!\nКоличество: " + (countnew - countold));  }  catch (Exception ex)  {  MessageBox.Show("Ошибка. Возможно потребуется переавторизация\n" + ex.Message);  }  }  //Показать больше писем  private void More\_Click(object sender, EventArgs e)  {  try  {  int countitems = listView1.Items.Count;  int count = client.GetMessageCount();  MessageHeader headers;  RfcMailAddress from;  string subject;  if (count == countitems)  {  MessageBox.Show("Больше писем нет!");  }  for (int i = count - countitems; i > 0; i--)  {  headers = client.GetMessageHeaders(i);  subject = headers.Subject;  from = headers.From;  string date = headers.Date;  ListViewItem lv = new ListViewItem(from.ToString());  lv.SubItems.Add(subject);  lv.SubItems.Add(date);  listView1.Items.Add(lv);  progressBar1.Value++;  if (progressBar1.Value == 100 || listView1.Items.Count > 25 + countitems)  break;  }  listView1.EnsureVisible(countitems);  listView1.Items[countitems].Selected = true;  progressBar1.Value = 0;  progressBar1.Visible = false;  }  catch (Exception ex)  {  MessageBox.Show("Ошибка обновления\n" + ex.Message);  }  }  //Скачать прикрепление к письму  private void DownloadB\_Click(object sender, EventArgs e)  {  foreach (MessagePart attachment in message.FindAllAttachments())  {  pictureBox1.Visible = true;  string dirname = From;  char[] invaliddirname = Path.GetInvalidFileNameChars();  foreach (char charinvalid in invaliddirname)  {  dirname = dirname.Replace(charinvalid, ' ');  }  try  {  Directory.CreateDirectory(dirname);  File.WriteAllBytes(Path.Combine(dirname, attachment.FileName), attachment.Body);  }  catch (Exception ex)  {  MessageBox.Show("Ошибка при скачивании файла" + ex.Message);  }  }  }  }  } |

Листинг 1.2. NewMessage.cs – Окно отправки нового сообщения

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.Net;  using System.Net.Mail;  using System.IO;  //Отправить новое сообщение  namespace MailClient  {  public partial class NewMessage : Form  {  string attachfpath = "";  public NewMessage()  {  InitializeComponent();  }  //Загрузка формы  private void NewMessage\_Load(object sender, EventArgs e)  {  MessageSent.Visible = false;  InfoAttach.Visible = false;  string filename = "Contacts.txt";  if (!File.Exists(filename))  {  FileStream fs = File.Create(filename);  fs.Close();  }  try  {  StreamReader sr = new StreamReader(filename);  string line = sr.ReadLine();  while (line != null)  {  comboBox1.Items.Add(line);  line = sr.ReadLine();  }  sr.Close();  }  catch (Exception ex)  {  MessageBox.Show("Ошибка : " + ex.Message);  }  object[] copyobj = new object[comboBox1.Items.Count];  comboBox1.Items.CopyTo(copyobj, 0);  comboBox2.Items.AddRange(copyobj);  comboBox3.Items.AddRange(copyobj);  }  //Функция отправки письма  public void SendMail()  {  string fpath = "Change.txt";  if (MessageSent.Visible == true)  MessageSent.Visible = false;  try  {  string[] change = File.ReadAllLines(fpath);  MailAddress to = new MailAddress(comboBox1.Text);  MailAddress from = new MailAddress(change[0], change[4]);  MailMessage mail = new MailMessage(from, to);  mail.Subject = SubjectField.Text;  mail.Body = BodyField.Text;  if (PriorityBox1.Checked == true)  {  mail.Priority = MailPriority.High;  }  if (comboBox2.Text != "")  {  MailAddress cc\_copy = new MailAddress(comboBox2.Text);  mail.CC.Add(cc\_copy);  }  if (comboBox3.Text != "")  {  MailAddress bcc\_copy = new MailAddress(comboBox3.Text);  mail.Bcc.Add(bcc\_copy);  }  SmtpClient smtp = new SmtpClient();  smtp.Host = change[2];  smtp.Port = 25;  smtp.Credentials = new NetworkCredential(change[0], change[1]);  smtp.EnableSsl = true;  if (!String.IsNullOrEmpty(attachfpath))  mail.Attachments.Add(new Attachment(attachfpath));  smtp.Send(mail);  MessageSent.Visible = true;  mail.Dispose();  }  catch (Exception e)  {  MessageBox.Show("Ошибка: " + e.Message);  }  }  //Отправить сообщение  private void SendMessage\_Click(object sender, EventArgs e)  {  SendMail();  }  //Прикрепить файл  private void AddFile\_Click(object sender, EventArgs e)  {  string name = "";  openFileDialog1.FileName = String.Empty;  if (openFileDialog1.ShowDialog() == DialogResult.OK)  {  attachfpath = openFileDialog1.FileName;  FileInfo f = new FileInfo(attachfpath);  if (f.Name.Length > 15)  {  name = f.Name.Remove(15);  name += "...";  }  else  { name = f.Name; }  InfoAttach.Visible = true;  InfoAttach.Text = "Файл прикреплен!\nИмя: " + name + "\nРазмер: " + f.Length / 1024 + "КБ";  }  }  //Очистить поля ввода  private void Clear\_Click(object sender, EventArgs e)  {  this.SubjectField.Clear();  this.BodyField.Clear();  }  //Справка, при наведении. Копия письма  private void label6\_MouseEnter(object sender, EventArgs e)  {  toolTip1.SetToolTip(label6, "CC:-(Carbon Copy). Копия письма");  }  //Справка, при наведении. Скрытая копия письма  private void label7\_MouseEnter(object sender, EventArgs e)  {  toolTip2.SetToolTip(label7, "BCC:-(Blind Carbon Copy). Скрытая копия письма");  }  }  } |

Листинг 1.3. Contacts.cs – Окно контакты

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.IO;  //Контакты  namespace MailClient  {  public partial class Contacts : Form  {  public Contacts()  {  InitializeComponent();  }  string filename = "Contacts.txt";  //Количество контактов  public void ContactsCount()  {  this.label3.Text = listBox1.Items.Count.ToString();  }  //Возникает при загрузке формы. Формирования списка контактов  private void Contacts\_Load(object sender, EventArgs e)  {  //если файл не существует, то создадим его  if (!File.Exists(filename))  {  FileStream fs = File.Create(filename);  fs.Close();  }  else  {  try  {  using (StreamReader sr = new StreamReader(filename))  {  //читаем построчно из файла до конца и добавляем строки в listbox1  string line = sr.ReadLine();  while (line != null)  {  listBox1.Items.Add(line);  line = sr.ReadLine();  }  sr.Close();  }  ContactsCount(); //общее кол-во контактов  }  catch (Exception ex)  {  MessageBox.Show("Ошибка: " + ex.Message);  }  }  }  //Поиск контактов  private void Search\_KeyUp(object sender, KeyEventArgs e)  {  if ((e.KeyCode == Keys.Enter) && (Search.Text != string.Empty))  {  int index = listBox1.FindString(Search.Text);  if (index != -1)  listBox1.SetSelected(index, true);  }  }  //Добавить контакт  private void AddUser\_Click(object sender, EventArgs e)  {  if (textBox2.Text != string.Empty)  {  listBox1.Items.Add(textBox2.Text);  ContactsCount();  }  textBox2.Clear();  }  //Удалить контакт  private void DeleteUser\_Click(object sender, EventArgs e)  {  if ((listBox1.SelectedItems.Count == 0) && (textBox2.Text == string.Empty))  {  MessageBox.Show("Выделите хотя бы один элемент из списка!", "Внимание!");  }  else  {  while (listBox1.SelectedItems.Count > 0)  {  listBox1.Items.Remove(listBox1.SelectedItems[0]);  }  }  if (textBox2.Text != string.Empty)  listBox1.Items.Remove(textBox2.Text);  textBox2.Clear();  ContactsCount();  }  //Возникает при закрытии формы  private void Contacts\_FormClosing(object sender, FormClosingEventArgs e)  {  int count = listBox1.Items.Count;  using (StreamWriter sw = new StreamWriter(filename))  {  for (int j = 1; j <= count; j++)  {  sw.WriteLine(listBox1.Items[j - 1]);  }  sw.Close();  }  }  }  } |

Листинг 1.4. Authorization.cs – Окно авторизации пользователя

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  using System.Net;  using System.Net.Mail;  using System.IO;  using System.Net.Mime;  using System.Text.RegularExpressions;  //Окно авторизации пользователя  namespace MailClient  {  public partial class ChangeUser : Form  {  string smtphost = "", pop3host = "";  public ChangeUser()  {  InitializeComponent();  }  //Очистить поля ввода  private void Clear\_Click(object sender, EventArgs e)  {  this.Login\_Field.Clear();  this.Password\_Field.Clear();  this.Name\_Field.Clear();  }  //Ок. Авторизоваться  private void Ok\_Click(object sender, EventArgs e)  {  string login, password, part = "";  int port = 25, index;  login = Login\_Field.Text;  password = Password\_Field.Text;  string[] domens = { "mail.ru", "bk.ru", "inbox.ru", "list.ru", "yandex.ru", "ya.ru", "gmail.com",  "rambler.ru", "lenta.ru", "autorambler.ru", "myrambler.ru", "ro.ru", "r0.ru" };  index = login.IndexOf("@");  part = login.Substring(index + 1);  for (int i = 0; i < 13; i++)  {  if (part == domens[i])  {  if (i >= 0 && i <= 3)  {  smtphost = "smtp.mail.ru";  pop3host = "pop.mail.ru";  break;  }  else  if (i == 4 || i == 5)  {  smtphost = "smtp.yandex.ru";  pop3host = "pop.yandex.ru";  break;  }  else  if (i == 6)  {  smtphost = "smtp.gmail.com";  pop3host = "pop.gmail.com";  break;  }  else  if (i >= 7 && i <= 12)  {  smtphost = "smtp.rambler.ru";  pop3host = "pop.rambler.ru";  break;  }  }  }  if (smtphost == "" && pop3host == "")  {  ServerSettings ss = new ServerSettings();  DialogResult dr = ss.ShowDialog();  if (ss.DialogResult == DialogResult.OK)  {  smtphost = ss.ReturnSMTP();  pop3host = ss.ReturnPOP3();  }  this.Close();  }  try  {  SmtpClient client = new SmtpClient(smtphost, port);  client.Credentials = new NetworkCredential(login, password);  client.EnableSsl = true;  MailMessage mail = new MailMessage(login, login);  mail.Subject = "Info: MailV";  mail.IsBodyHtml = true;  var sHtml = "<HTML>\n" +  "<HEAD>\n" +  "<TITLE>рассылка при успешной авторизации пользователя</TITLE>\n" +  "</HEAD>\n" +  "<BODY><P>\n" +  "<table style=\"border: 1px solid darksalmon\"><tbody><tr><td>Configuration Successful!</td></tr><tr><td>Thank you!</td></tr><tr><td>Creator: [Insomnia]. (unlucky\_v@mail.ru)</td></tr></tbody></table>" +  "</BODY>\n" +  "</HTML>";  var htmlType = new ContentType("text/html");  var htmlView = AlternateView.CreateAlternateViewFromString(sHtml, htmlType);  mail.AlternateViews.Add(htmlView);  mail.Body = sHtml;  client.Send(mail);  MessageBox.Show("Авторизация прошла успешно!", "Готово!");  this.Close();  Program.mainForm.groupBox1.Text = login;  using (StreamWriter file = new StreamWriter("Change.txt"))  {  file.WriteLine(login);  file.WriteLine(password);  file.WriteLine(smtphost);  file.WriteLine(pop3host);  file.WriteLine(Name\_Field.Text);  file.Close();  }  Program.mainForm.userpopconnect();  }  catch (Exception ex)  {  MessageBox.Show("Попробуйте снова!\nТекст ошибки:\n" + ex.Message);  }  }  //Справка при заполнении поля "Пароль"  private void textBox2\_Enter(object sender, EventArgs e)  {  toolTip1.SetToolTip(Password\_Field, "Некоторые сервисы используют отдельный пароль для сторонних приложений!");  }  //Справка при заполнении поля "Имя"  private void textBox3\_Enter(object sender, EventArgs e)  {  toolTip2.SetToolTip(Name\_Field, "Это имя, которое видят получатели ваших писем в поле «От кого».");  }  }  } |

Листинг 1.5. FullMessage.cs – Окно просмотра письма во весь экран

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  //Письмо во весь экран  namespace MailClient  {  public partial class FullMessage : Form  {  public FullMessage()  {  InitializeComponent();  }  //Закрыть окно  private void Close\_Click(object sender, EventArgs e)  {  this.Close();  }  }  } |

Листинг 1.6. Settings.cs – Окно настроек

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  //Настройки  namespace MailClient  {  public partial class Settings : Form  {  public Settings()  {  InitializeComponent();  }  //Сменить, авторизовать пользователя  private void ChangeUser\_Click(object sender, EventArgs e)  {  ChangeUser dialog = new ChangeUser();  DialogResult dr = dialog.ShowDialog();  this.Close();  }  }  } |

Листинг 1.7. ServerSettings.cs – Настройка серверов

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Windows.Forms;  //Окно настройки smtp и pop3 серверов  namespace MailClient  {  public partial class ServerSettings : Form  {  public ServerSettings()  {  InitializeComponent();  }  private void Clear\_Click(object sender, EventArgs e)  {  this.Smtp\_Field.Clear();  this.Pop3\_Field.Clear();  }  private void OK\_Click(object sender, EventArgs e)  {  this.DialogResult = DialogResult.OK;  }  public string ReturnSMTP()  {  return (Smtp\_Field.Text);  }  public string ReturnPOP3()  {  return (Pop3\_Field.Text);  }  }  } |