МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра прикладной информатики

Институт информационных систем и геотехнологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(бакалаврская работа)
по направлению подготовки 09.03.03 Прикладная информатика
(квалификация – бакалавр)

На тему «Проектирование приложения-месседжера для рассылки информационных сообщений»

Исполнитель Носов Сергей Александрович

Руководитель к.т.н., доцент, доцент кафедры ПИ Попов Николай Николаевич

«К защите допускаю»

заведующий кафедрой _____

доктор технических наук, профессор

Истомин Евгений Петрович

«____» _____ 2020 г.

Санкт-Петербург
2020

ОГЛАВЛЕНИЕ

Введение

Актуальность выполнения данной работы обусловлена тем, что сегодня уже невозможно обойтись без средств современных информационно-коммуникационных технологий. Необходимость использования средств связи и коммуникации в глобальной сети Интернет стало необходимым условием, как ведения бизнеса, так и общения в сети. Все это накладывается определенные условия на наличие специализированных средств обработки информации для взаимодействия в современном информационном мире и условиях глобальной сети Интернет.

Компании получают колоссальные конкурентные преимущества, если уровень развития информационных систем соответствует уровню развития предприятия и сфер его деятельности. Улучшается инвестиционная привлекательность компании, основные бизнес-процессы становятся прозрачными и понятными для контроля и управления, исключаются ошибки, брак и связанные с ними потери и времени, и средств, в конечном счете, увеличивается прибыль компании. Повышения уровня информированности, как сотрудников, так и клиентов по средствам рассылки информационных сообщений позволяет добиться высокого уровня информационного взаимодействия и экономического эффекта.

Электронная почта представляет собой технологию и службу по пересылке и получению электронных сообщений между пользователями компьютерной сети (в том числе глобальной сети Интернет).

Недостатки электронной почты: наличие такого явления, как спам (массовые рекламные и вирусные рассылки); возможные задержки доставки сообщения (до нескольких суток); ограничения на размер одного сообщения и на общий размер сообщений в почтовом ящике (персональные для пользователей). Требуется постоянная серьезная работа не только IT-специалистов, но и топ-менеджеров по согласованию или точнее – синхронизации всех усилии в области стратегического развития компании и

используемых в ней информационных систем.

Поэтому процесс проектирования информационной системы в настоящее время становиться обязательным условием для эффективного функционирования компании. В данном случае, если этот процесс не впервые выполняется компанией, то термин проектирование может быть приравнен к понятию развитие информационной системы.

Этим объясняется дос
проектирования информационных систем в последние годы. Прежде всего, создание CASE-технологии, которые во много позволят значительно сократить общие сроки проектирования информационной системы, позволяют организовать одновременную коллективную работу группы разработчиков и заказчика, оперативно вносить необходимые изменения и достаточно быстро реагировать на изменение обстановки на предприятии и дополнительные требования заказчика.

Все это актуализирует необходимость использования в работе предприятия приложений для рассылки информационных сообщений, что позволит повысить уровень информационного взаимодействия между пользователями.

Объект исследования – предприятие.

Предмет исследования – проектирование приложения.

Целью данной работы является проектирование приложения для рассылки информационных сообщений.

В соответствии с целью была определена необходимость постановки и решения следующих задач:

– изучить теоретические основы проектирования приложений;

– выполнить анализ существующих разработок;

– выбрать и описать средства разработки;

– разработать приложение для рассылки информационных сообщений;

– описать порядок работы с приложением.

1 Теоретические основы проектирования приложений

1.1 Введение в предметную область

Информационная система – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые и т.д.), которые обеспечивают и распространяют информацию (ISO/IEC 2382:2015).

Информационная система предназначена для своевременного обеспечения надлежащих людей надлежащей информацией, то есть для удовлетворения конкретных информационных потребностей в рамках определённой предметной области, при этом результатом функционирования информационных систем является информационная продукция – документы, информационные массивы, базы данных и информационные услуги [6, с.130].

Информационная система реализуется через совокупность информации, которая содержится в базе данных, и технологий, которые обеспечивают обработку информации. В данном случае, к технологиям можно отнести и методы обнаружения, сбора, обработки, хранения, распространения информации, и различные способы, позволяющие эти методы реализовать на практике. Информационное управление при этом сводится к непосредственному использованию данных методов для контроля за процессами планирования, дизайна, эксплуатации и анализа программной системы.

Разработка информационных систем связана с большим количеством стадий, который взаимодействуют между собой и дополняют друг друга. Это обуславливает необходимость изучения подходов и методологий разработки программных систем.

Проектированием информационных систем называется многоступенчатый процесс их создания и/или последующей модернизации по средствам непосредственного применения упорядоченной совокупности специализированных методологий и соответствующего инструментария.

Проектирование информационной системы (в отличие от моделирования) предполагает реализацию комплекса работ с пока несуществующими объектами и направлено на непосредственное создание информационной системы в области:

– обработки структурных объектов будущей базы данных;

– написания прикладных программ (в том числе – отчётных и экранных пользовательских форм), которые предназначены обеспечивать взаимодействие пользователя с системой в процессе выполнения структурированных запросов к данным;

– выполнения оперативного учёта функционирования конкретной среды функционирования информационной системы (используемой технологии) [9, с.76].

Если выделять стадию проектирования информационной системы в качестве самодостаточного этапа, то его можно разместить между этапами анализа и разработки программной системы. Однако на практике чёткое разделение на отдельные этапы, как правило, является затруднительным или невозможным, в связи с тем, что проектирование, формально начинаясь с непосредственного определения цели проекта, часто продолжается на последующих стадиях тестирования и реализации программной системы.

В общем виде целевые установки проектов по созданию программных систем сводятся к непосредственному обеспечению комплекса условий, которые позволяют эту информацию получать, обрабатывать и в последствии использовать по средствам создания функциональной безотказной системы с достаточными условиями, в частности:

– высоким уровнем программной адаптивности к изменяемым условиям функционирования программной системы;

– пропускной способностью реализуемой программной системы в рамках проекта автоматизации;

– временем осуществления системной реакции на структурированный запрос к программной системе;

– уровнем безопасности программной системы;

– степенью общей простоты в практической эксплуатации программной системы.

Организацию операций проектирования программной системы можно разделять на несколько типов:

– каноническое проектирование программной системы отражает характерные особенности технологии оригинального (индивидуального) процесса;

– типовое проектирование, для которого является характерным типовое проектное решение, которое может быть тиражировано и пригодно к многократному использованию в рамках автоматизации предметной задачи.

Каноническое проектирование программных систем применяется, главным образом, для относительно небольших и локальных прикладных программных систем с минимальным использованием типовых подходов к разработке. Адаптация реализуемых проектных решений выполняется только посредством перепрограммирования прикладных программных модулей определенной программной системы.

Каноническое проектирование организовывается с практическим использованием каскадной модели жизненного цикла определенной прикладной программной системы. Это предполагает непосредственное разделение процесса разработки программной системы на следующие стадии и этапы:

– предпроектная стадия реализуется через анализ и в процессе нее составляется техническое задание на программную системы. То есть, формируются набор требований к программной системе, разрабатывается её общая концепция, составляется технико-экономическое обоснование и пишется техническое задание на программную систему;

– проектная стадия предусматривает непосредственное составление эскизного и технического проектов, разработку рабочей документации на программную систему;

– послепроектная стадия даёт старт комплексу мероприятий по непосредственному внедрению программной системы, обучению персонала, анализу результатов опытных испытаний системы. Частью этой стадии становится сопровождение программной системы и устранение выявленных недостатков в ее работе [2, с.230].

Перечисленные этапы проектирования определенной программной системы, в случае возникновения необходимости, можно выполнять укрупнение или сильнее детализировать – объединять последовательные этапы, можно исключать «лишние», начинать выполнение очередной стадии до завершения предыдущей.

Метод типового проектирования отличается наличием возможностей декомпозиции проектируемой программной системы с разделением на структурные компоненты, в число которых могут входить прикладные программные модули, специализированные программные подсистемы, комплексы задач и др. Для непосредственной реализации программных компонентов можно воспользоваться типовыми решениями, существующими на рынке, и настроить их под нужны конкретной организации и автоматизируемых бизнес-процессов. При этом типовое проектирование программной системы предполагает обязательное наличие документации, которая позволяет описать в деталях технологию проектирования программной системы и процедуры настройки.

Декомпозиция автоматизируемой задачи может иметь несколько уровней, что позволяет выделить соответствующие классы проектирования программной системы:

– элементные – по отдельной задаче автоматизации (отдельному элементу);

– подсистемные – по отдельным подсистемам разрабатываемой программной системы;

– объектные представляют собой отраслевые типовые проектные решения, которые в себе содержат полный комплекс специализированных прикладных подсистем программной системы [1, с.57].

Возможность практической реализации модульного подхода считается достоинством элементных типовых проектных решений. Однако, необходимо отметить, что в случае установления определенной несовместимости различных элементов процесс их непосредственного объединение приводит к существенному увеличению затрат на разработку прикладной программной системы.

Подсистемные типовые проектные решения, помимо практической реализации модульного подхода, дают определенные возможности по проведению параметрической настройки на информационные объекты различных уровней управления. Проблемы с непосредственным объединением в таких системах возникают в случае использования продуктов от нескольких различных производителей прикладного программного обеспечения.

В ходе реализации типового проектирования применяются параметрически-ориентированный и модельно-ориентированный подходы.

В основе технологии проектирования лежит выбранная для конкретной задачи методология как совокупность принципов, выраженная в единой определённой концепции программной системы.

1.2 Методологические аспекты разработки информационной системы

Методология создания информационных систем заключается в организации процесса разработки информационной системы и обеспечении непосредственного управления этим процессом для того, чтобы в полной мере гарантировать выполнение всех установленных требований, как к самой информационной системе, так и к характеристикам всех стадий процесса разработки.

Среди основных задач, решение которых должна обеспечить используемая методология создания информационных систем, являются следующие:

– обеспечение непосредственного создания информационной системы, которая отвечает всем установленным целям и задачам конкретного предприятия и соответствующих предъявляемым к ним функциональным требованиям;

– гарантия создания информационной системы с четко установленными параметрами в течение определенного времени в рамках оговоренного заранее бюджета проекта автоматизации предметной области;

– простота последующего сопровождения, модификации и расширения информационной системы с целью непосредственного обеспечения ее соответствия постоянно изменяющимся условиям работы предприятия и взаимодействия с клиентами;

– обеспечение создания информационной системы, которая отвечает установленному комплексу требований открытости, переносимости и последующей масштабируемости;

– возможность практического использования в создаваемой информационной системе разработанных ранее средств информационных технологий (прикладного программного обеспечения, баз данных, средств информационно-коммуникационных технологий) [4, с.304].

Сегодня существует не так много методологий, особенно полных, т.е. тех, которые позволяют учесть все значимые стадии жизненного цикла разработки прикладной программной системы. Именно методология обеспечивает определение, какие именно языки программирования и системы будут использоваться для непосредственной разработки информационной системы и, во многом, рекомендует, какой именно технологический подход будет при этом использован командой разработчиков системы.

Рассмотрим основные методологии, которые могут быть использованы для разработки информационной системы.

RAD (от англ. rapid application development – быстрая разработка приложений) представляет собой концепцию непосредственной организации технологических процессов разработки информационной системы, ориентирована на максимально быстрое получение конечного результата в условиях множества ограничений в части сроков и бюджета и частично очерченных требований к продукту автоматизации. Эффект ускорения разработки информационной системы достигается по средствам непосредственного использования определенных технических средств и непрерывного, параллельного с ходом разработки, уточнения комплекса требований и оценки текущих результатов с привлечением заказчика информационной системы [11, с.69].

Принципы быстрой разработки RAD технологии в полной мере направлены на непосредственное обеспечение основных преимуществ, к которым можно отнести: высокую скорость разработки программной системы, низкую стоимость и высокое качества конечного программного продукта, (рисунок 1).

Рисунок 1 – Преимущества RAD методологии

Достигнуть высокого качества разрабатываемой программной системы достаточно сложно и одна из основных причин появляющихся трудностей в проекте разработки лежит в том, что разработчики и заказчик могут видеть

предмет разработки программной системы по-разному в следующих аспектах разработки:

– используемый инструментарий должен быть направлен на минимизацию используемого времени разработки программной системы;

– создание прототипа программной системы для возможности уточнения специфических требований заказчика;

– цикличность разработки реализуется через то, что каждая последующая версия программной системы базируется на оценке результатов работы предыдущих версий системы;

– минимизация времени разработки версий программной системы, за счёт непосредственного переноса уже готовых прикладных программных модулей и добавления специфической функциональности в новую версию программной системы;

– команда разработчиков программной системы должна тесно сотрудничать, каждый отдельный участник в полной мере должен быть готов выполнять набор обязанностей в рамках проекта автоматизации предметной области;

– управление проектом должно позволить значительно минимизировать длительность циклов разработки программной системы [14, с.149].

Принципы методологии RAD используют не только для практической реализации, но и данная методология распространяется на все этапы жизненного цикла разрабатываемых программных систем, в частности на этапе обследования определенной организации, построения комплекса требований, выполнение анализа и реализации дизайна разрабатываемой программной системы.

Немаловажным является и наличие в RAD методологии фаз проекта автоматизации: планирование; пользовательское проектирование; конструирование; переключение, (рисунок 2).

Планирование представляет собой совокупность требований, которые были получены в процессе проведения системного планирования и анализа процедур разработки жизненного цикла (SDLC) программной системы.

Данная фаза завершается непосредственным согласованием ключевых моментов с RAD-группой и получением от руководителя проекта необходимых разрешений на продолжение работа по проекту автоматизации предметной области.

Рисунок 2 – Фазы RAD методологии

Пользовательское проектирование – на протяжении данного этапа пользователи, в процессе взаимодействия с системными аналитиками, выполняют разработку моделей и прототипов, включающие в себя все необходимые системные функции будущей программной системы. Для перевода прототипов в рабочие модели RAD-группой используется техника объединенной разработки приложений (JAD) и специализированные CASE-инструменты в виде прикладного программного обеспечения, например, ERWin.

Пользовательское проектирование оказывается длительным интерактивным процессом, позволяющий пользователям получить поние о том,

что нужно ли изменять и в конечном счёте выбрать рабочую модель, которая будет отвечать их требованиям к программной системе.

Переключение – включает в себя множество операций по конверсии оперативных данных, непосредственное тестирование программной системы, переход на новую систему и тренировку конечных пользователей программной системы. По своим задачам данная стадия напоминает финальную стадию SDLC методологии [17, с.37].

Сравнивая с традиционными методами разработки программных систем, весь процесс представляется сжатым по времени реализации. Как результат, новую программную систему можно быстрее построить, доставить до непосредственного заказчика и установить на рабочих местах конечных пользователей.

Адаптивная методология (Adaptive Software Development или ASD) представляет собой одну из достаточно новых методологий, которые разработали в виде определенной альтернативы традиционным методологиям, которые в полной мере ориентированы на непосредственный процесс, методам управления практической разработкой прикладных программных систем. Основную роль в методологии ASD играет человеческий фактор, конечные результаты работы и минимизация непосредственного процесса при максимальном увеличении соответствующего взаимодействия между всеми участниками проекта автоматизации поставленного комплекса задач в рамках предметной области.

Методология ASD базируется на основополагающих принципах реализации непрерывной адаптации, за счет которой возникает управляемый жизненный цикл проекта автоматизации. Постоянно появляющиеся изменения в проекте автоматизации становятся нормой и с ними можно эффективно работать.

В адаптивной методологии ASD обычный статический жизненный цикл разработки программной системы включает следующие этапы «Планирование – Проектирование – Конструирование» заменен на динамический, который

включает такие этапы разработки программной системы: «Обдумывание – Взаимодействие – Обучение».

Этот цикл обеспечивает непрерывное обучение пользователей разрабатываемой программной системы. Данный цикл в полной мере связан с возникающими изменениями, получением повторных оценок попытками в какой то мере предугадать неизвестное раннее на данный момент будущее проекта автоматизации и требует тесного взаимодействий между разработчиками программной системы, тестировщиками и первичным заказчиками разрабатываемой системы.

Адаптивная методология ASD построена на базе концептуальной теории сложных адаптивных систем. Данная методологии в полной мере рассчитана на непосредственное использование в сложных проектах, в которых преобладает достаточно быстрый темп разработки, практически полная непредсказуемость и частые изменения проекта автоматизации предметной области. Есть уникальные проекты, которые не являются экстремальными, однако для всех остальных ASD подходит намного лучше, чем любой подход, который сейчас используется для разработки программного обеспечения.

Feature Driven Development или FDD представляет собой функционально-ориентированную разработку, в которой основную роль играет понятие функций или свойств (feature) разрабатываемой программной системы. Каждая отдельная функция программной системы должна быть реализована в течении двух недель и не более. То есть, если реализуемый сценарий непосредственного использования будет мал, его можно считать программной функцией. Если сценарий будет больше, то его нужно будет разбить на несколько, относительно небольших функций.

Методология FDD включает несколько процессов, последние два из которых будут повторяться для каждой отдельной функции программной системы:

– разработка общей информационной модели программной системы и ее подсистем;

– составление полного списка необходимого функционала программной системы;

– планирование соответствующих работ над каждой отдельной функцией разрабатываемой программной системы;

– проектирование вспомогательных функций программной системы;

– конструирование программных функций [3, с.48].

Разработчики программной системы в нотации методологии FDD делятся на «главных программистов» и «хозяев классов». Главные программисты могут привлекать хозяев используемых классов к непосредственной работе над очередным свойством разрабатываемой программной системой. Для сравнения можно привести такой пример, в ОС Windows XP нет персонально ответственных за программные классы или функциональные методы. К работе над любым отдельном классом может привлекаться любой из вспомогательных участников разработки программной системы или прикладных программных компонентов [5, с.87].

Работы над проектом автоматизации предметной области предполагает реализацию частых сборок и деления работа на частные итерации, каждая из которых будет предполагать непосредственную реализацию некоторого набора функций конечной программной системы.

Таким образом, анализ методологий разработки программных систем показал наличии большого количества подходов, среди которых были выделены такие методологии: RAD; ASD; FDD. Каждая из перечисленных методологий имеется свои особенности. Наиболее подходящей для проекта автоматизации поставленного комплекса задач является первая методология RAD.

Практическое использование методологии RAD разработки программных систем позволяет обеспечить следующие преимущества: быстроту непосредственного продвижения разработанной программной системы на рынок информационных технологий; пользовательский интерфейс, в полной мере будет устраивать конечного пользователя; предполагает лёгкую

адаптацию проекта автоматизации к изменяющимся требованиям конечного заказчика и окружения предметной области; простоту последующего развития функциональности реализованной программной системы.

1.3 Анализ существующих разработок и методик

Современный рынок информационных технологий предлагает массу программных решений рассылки сообщений. Среди предложений выделяются следующие программы: ePochta Mailer; ePochta Studio; AMS Enterprise.

Программная система ePochta Mailer является одной из лидеров среди программных продуктов для e-mail маркетинга используется для массовых e-mail рассылок, поиска e-mail адресов, проверки и управления списками рассылок сообщений, (рисунок 3).

Преимущества массовой рассылки e-mail сообщений при помощи программной системы ePochta Mailer: реализован полный набор функций для выполнения профессиональной массовой рассылки писем; представлен набор встроенных инструментов для управления списками e-mail адресов; реализована возможность непосредственной проверки электронных писем на спамность при помощи использования специального фильтра SpamAssasin; представлены инструменты мониторинга результатов проведенной e-mail кампании; не требует установки дополнительных программных утилит для полноценной работы конечным пользователем [7, с.210].

Программа разработана с учетом поддержки трех видов SMTP серверов: встроенного, внешнего и партнерского.

По умолчанию в программу встроен собственный SMTP сервер, который позволяет отправлять e-mail сообщения напрямую, обходя посреднические SMTP сервера и увеличивая общую скорость отправки электронных писем. В тоже время, практическое использование такого способа может быть возможным лишь в случае, если прямая рассылка будет разрешена интернет-провайдером организации [30, с.205].

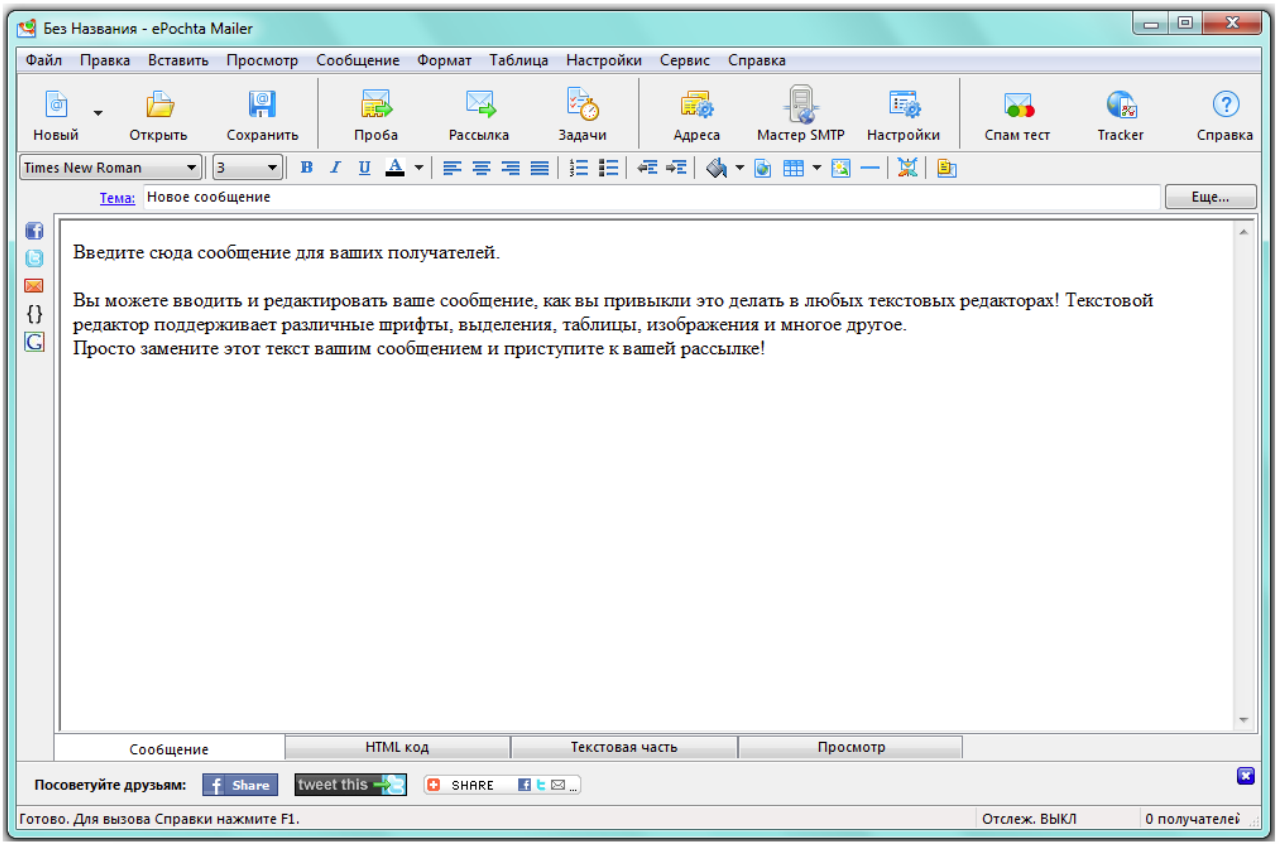

Рисунок 3 – Интерфейс ePochta Mailer

В противном случае рассылка сообщений будет в полной мере заблокирована. Для того, чтобы избежать случаев блокировки можно рекомендовать использовать вспомогательные прикладные программные компоненты SMTP сервера.

Благодаря наличию возможностей для многопоточной рассылки и неограниченному числу подключаемых серверов процесс непосредственной отправки e-mail кампаний существенно может быть ускорен и защищен соответствующими средствами протокола SMTP [10, с.52].

Преимущество практического использования партнерского SMTP сервера в том, что пользователю нет необходимости настраивать сервер вручную. Программа ePochta Mailer самостоятельно может подключиться к серверу SMTP, требуя от пользователя только параметры аутентификации, и затем отправляет рассылки электронных сообщений по средствам использования интегрированного SMTP.

Следующая программа ePochta Studio представляет собой уникальное прикладное программное решение, включающее многофункциональный пакет прикладных продуктов для электронного маркетинга в условиях глобальной сети Интернет. Данная программная системы выступает своеобразной оболочкой к приложениям и сервисам ePochta.

В состав продукта ePochta Studio входят специализированные прикладные программные модули для: рассылки e-mail сообщений; управления списками рассылки; автоматизации процесса подписки и удаления адресов в электронной рассылке [8, с.170].

Каждый отдельный прикладной программный модуль представлен отдельной программой ePochta, которую можно запустить при помощи непосредственного выбора определенной функции программной системы. В ePochta в полной мере реализованы возможности создания и возможности последующего управления проектом e-mail кампании в глобальной сети Internet.

Так как программная система ePochta Studio работает с отдельными проектами информационной e-mail кампании, то конечный пользователь может выполнить следующий набор действий в отношении к каждому отдельному проекту: добавить необходимый набор e-mail адресов; выполнять необходимые правки списка контактов; проверять электронные адреса; создать и разослать информационные письма; оценить общую эффективность проведенной e-mail кампании организацией [12, с.360].

AMS Enterprise представляет собой высокоэффективную программу для выполнения массовой e-mail рассылки. Весь необходимый функционал для подготовки и непосредственного проведения рассылок в программе AMS Enterprise реализован следующими возможностями, среди которых можно выделить: импорт списков имеющихся подписчиков из любых источников, обработка имеющихся списков (удаление дубликатов в информационном массиве, объединение, разъединение, сравнение списков, многокритериальный поиск и удаление необходимых контактов по любым установленным

критериям, реализована возможность выполнения своих структурированных SQL запросов), представлен достаточной мощный Html редактор для выполнения непосредственной подготовки электронных писем, реализована автоматизация процесса подписки/отписки на e-mail рассылки информационных сообщений и реализован анализ входящей почты, запуск рассылок по установленному расписанию, (рисунок 4).

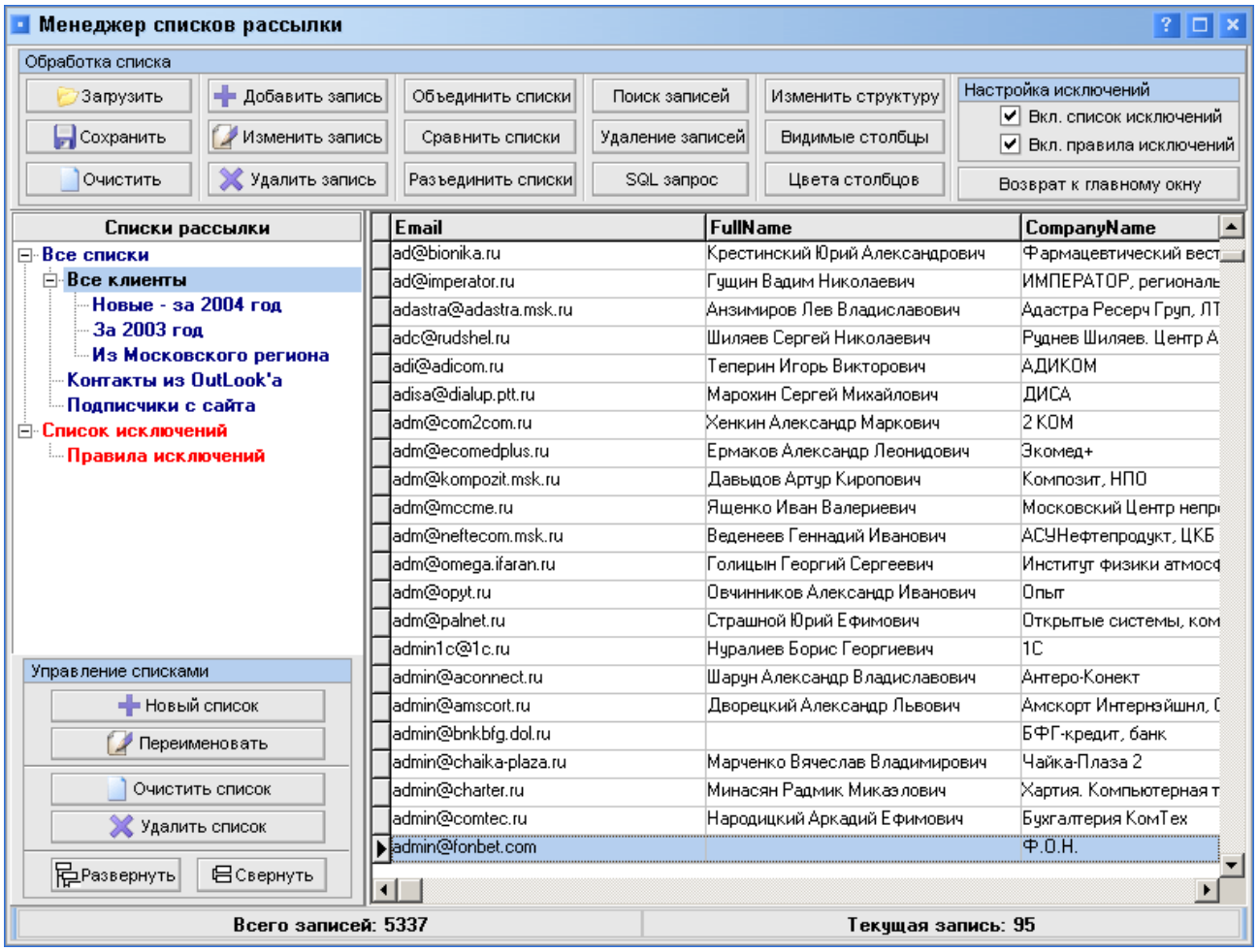

Рисунок 4 – Интерфейс AMS Enterprise

Многофункциональный MailMerge функционал, который реализован в программной системе AMS Enterprise позволяет обеспечить конечного пользователя комплексом возможностей рассылать информационные письма. Гибкое ядро для отправки информационных сообщений в программной системе AMS Enterprise обеспечивает: возможность выполнения рассылки по средствам использования Smtp релеев и/или через специализированный Smtp сервер

программной системы, реализованы возможности гибкого распределения потоков писем между ролями и доменами адресов конечных получателей сообщений, представлены настройки различных ограничений скорости отправки информационных сообщений, реализованы возможности непосредственного запуска большого количества рассылок электронных сообщений одновременно [16, с.249].

Таким образом, анализ существующих программных систем показал наличие в них большого количества функциональных возможностей, воспользовавшись которыми можно решить часть поставленного комплекса задач. В тоже время, для решения задачи рассылки информационных сообщений в условиях предприятия недостаточно использовать готовый продукт, а необходимо использовать дополнительный модуль, который позволит использовать корпоративные данные для своей работы.

1.4 Выбор и описание средств разработки

Реализация приложения для рассылки информационных сообщений с учетом требований предприятия может быть выполнена на базе использования средств языка программирования Python. Данный язык позволяет реализовать необходимый спектр функциональных возможностей разрабатываемой программной системы для рассылки информационных сообщений.

Python представляет собой высокоуровневый язык программирования общего назначения, который в полной мере ориентирован на непосредственное повышение общей производительности разработчика программной системы и читаемости конечного программного кода. Синтаксис ядра языка программирования Python является минималистичным и хорошо воспринимается командой разработчиков. В то же, время стандартная библиотека языка программирования Python включает достаточно большой объём полезных дополнительных функций, при помощи которых можно решить множество прикладных задач.

Python поддерживает объектно-ориентированное, функциональное, структурное, аспектно-ориентированное и императивное программирование. Основные черты архитектуры языка программирования Python включают динамическую типизацию данных, возможность автоматического управления памятью, реализована полная интроспекция, представлены механизмы обработки множества исключений, имеется поддержка многопоточных вычислений, реализована технология работы с высокоуровневыми структурами данных.

В языке программирования Python реализована возможность разбиения прикладных программных систем на вспомогательные программные модули, которые, в свою очередь, могут объединяться в специальные программные пакеты.

Язык Python обладает достаточно последовательным и чётким синтаксисом, в нем на высоком уровне продумана масштабируемость и модульность, за счет чего конечных программный код программ легко читаем, как разработчиком, так и заказчиком программной системы.

При осуществлении передачи аргументов в используемые программные функции на языке программирования Python используется уникальный механизм вызова, который называется соиспользованием (call-by-sharing). Иногда это может быть похожим на стратегию по определенному значению (например, при разработке программ, в которых есть скалярные значения int, float и str), а иногда и как эталонная стратегия разработки (например, при непосредственной работе со значениями структурированного типа, такими как list, dict, set и object).

Набор встроенных операторов в языке Python практически схож с другими языками программирования и включает:

– условный оператор if с соответствующим альтернативным блоком else. Если разработчиком будет использовано условий и альтернатив несколько, можно воспользоваться оператором elif (сокр. от else if);

– операторы цикла while и for. Внутри программного цикла можно применить continue и break для осуществления прерывания определенного цикла и выполнения перехода сразу к последующей итерации алгоритма;

– оператор определения класса class;

– оператор определения пользовательских функций, генератора или метода def. Внутри можно применить возврат return для осуществления возврата из выполняемой функции или некоторого метода, а в случае реализации генератора – yield;

– оператор обработки исключений try – except – else или try – finally (работа с исключениями позволяет осуществлять обработки и параллельное тестирование разрабатываемой программной системы);

– оператор pass ничего не выполняется и используется для реализации пустых блоков программного кода разрабатываемой программной системы [20, с.340].

Одной из достаточно интересных синтаксических особенностей языка Python является возможность выделения блоков программного кода при помощи использования отступов (табуляций или пробелов), в свиязи с чем, в языке Python нет операторных скобок begin/end, например, как в языке программирования Pascal, или фигурные скобки, как в языке программирования Си.

Такой подход к разработке программных систем позволяет значительно сократить общее количество строк программного кода и используемых символов в разрабатываемой программной системе и способствует непосредственному приучению к «хорошему» стилю программирования разработчика программной системы.

В языке программирования Python поддерживается динамическая типизация, то есть тип переменной будет определяться только во время непосредственного исполнения переменной в программном коде, этот механизм принято называть связыванием значения с определенным именем [29, с.140].

Дизайн языка программирования Python построен вокруг объектно-ориентированной модели программирования. Реализация объектно-ориентированного программирования в Python является элегантной, достаточно мощной и в полной мере продуманной, но вместе с тем достаточно специфической в сравнении с другими объектно-ориентированными языками программирования [24, с.43].

Все объекты в языке программирования Python делятся на ссылочные и атомарные.

К атомарным объектам относятся int, long (в версии Python 3 любое число int, так как в версии Python 3 нет ограничения на размер), complex и некоторые другие [13, с.350].

Богатая стандартная библиотека является одной из привлекательных сторон языка программирования Python. В нем имеются специализированные средства для работы с большим количеством сетевых протоколов и форматами глобальной сети Интернет, например, программные модули для написания HTTP-серверов и программных клиентов, для разбора и непосредственного создания почтовых сообщений, для работы с XML и т.п.

Набор модулей для работы с операционной системой позволяет писать кросс-платформенные прикладные программные приложения.

Таким образом, анализируя возможности языка программирования Python было установлено, что данный язык позволит реализовать необходимую программу рассылки информационных сообщений.

2 Проектирование системы рассылки информационных сообщений

2.1 Постановка задачи на разработку

В современном мире люди получают множество писем, а у некоторых есть несколько почтовых ящиков. Все это усложняет процесс их непосредственного администрирования, что вынуждает искать пути решения проблемы информационного взаимодействия в условиях глобальной сети Интернет.

Электронные письма – это набор текстовых протоколов для передачи информации между сетями; как и весь остальной Интернет, в общем. Всё это лишь несколько уровней протоколов, а конечный пользователь работает с интерфейсом.

Корпоративная почта - система электронных почтовых ящиков предприятия, объединенная одним доменным именем. В качестве доменного имени чаще всего используется название предприятия. Кроме того корпоративная почта как система включает в себя ряд дополнительных сервисов, позволяющих оптимизировать работу сотрудников и подразделений предприятия.

Корпоративная почта – незаменимый помощник для бизнеса и лицо организации: она помогает привлекать клиентов, вести переписку с контрагентами и партнерами, моментально обмениваться важными документами, макетами и прочими файлами, гарантирует своевременное получение важной информации, выполняет функцию автоответчика. Корпоративная почта позволяет отделить личную переписку сотрудников от служебной, что позволяет минимизировать отвлекающие от работы моменты. Кроме того, корпоративная почта позволяет руководству компании контролировать переписку сотрудников и управлять потоками информации, снизить зависимость от человеческого фактора.

Создание корпоративной почты реализуется по средствам актуализации следующих положений.

Во-первых, для создания корпоративной почты необходимо зарегистрировать домен. Если домен уже зарегистрирован и у компании есть собственный web-сайт, можно переходить к следующему пункту. Если собственного домена ещё нет, его можно приобрести с необходимым набором параметров.

Во-вторых, необходимо подключиться к соответствующей услуге корпоративного почтового хостинга. Услуга обеспечивает возможность расположить сервер корпоративной почты на защищённом от вирусных атак и спама почтовом хостинге и позволяет достаточно сильно сэкономить на приобретении серверного оборудования, специального прикладного программного обеспечения почтового сервера, его последующей настройки, поддержании высокоскоростного отказоустойчивого подключения к глобальной сети Internet.

После выполнения подключения к соответствующей услуге администраторам высылается полная инструкция по параметрам настройки корпоративной почты. Настройка подключения может выполняться по всем возможным почтовым протоколам – POP3/IMAP4/SMTP/HTTP.

Во всех возможных случаях для обеспечения надежной безопасности соединения используются специализированное шифрованные версии протоколов: SSL или TLS. Также в инструкции приведены примеры настроек аккаунтов корпоративной почты для наиболее широко распространённых почтовых клиентов: Microsoft Outlook, Mozilla Thunderbird и Apple Mail.

Кроме имиджевой составляющей, которая, безусловно, важна для любой компании, корпоративная почта имеет и существенные технологические преимущества. Среди них стоит выделить:

– практически полное отсутствие спама благодаря высококачественным спам-фильтрам;

– практически неограниченный объем почтового пространства;

– централизованное управление почтовыми ящиками предприятия, что особенно актуально при смене персонала.

Администрирование почты представляет собой процесс анализа и обработки электронных писем (входящий и исходящий контент). Если пользователю нужно написать или ответить на 5 – 10 сообщений, особых затруднений возникнуть не должно. А вот, если нужно сделать массовую рассылку 200 клиентам или более могут возникнуть серьезные проблемы учета этих писем. Для решения данной проблемы нужно сделать первые шаги автоматизации процесса рассылки e-mail сообщений в условиях конкретного предприятия и его информационного пространства.

Рассылка e-mail сообщений может быть организована при реализации следующих положений:

– подключение к серверу электронной почты и авторизация;

– формирование письма e-mail средствами определенного текстового редактора, наполнение его необходимыми данными;

– формирование списка e-mail адресов рассылки, для чего целесообразно использовать отдельный файл с адресами рассылки;

– отправка сообщения при помощи использования специального скрипта на языке программирования Python.

Преимущества использования собственной программной системы для отправки сообщений:

– высокая степень защиты информации;

– использование новейшего оборудования обеспечивающего стабильную работу почтового сервиса;

– полное отсутствие спама;

– синхронизация почты, контактов, календаря и других полезных программ, а также возможность работы через почтовых клиентов.

2.2 Проектирование программной системы

В терминологии электронной почты выделяются следующие программные компоненты: MTA; MDA; MUA; MRA.

MTA (Mail Transfer Agent – агент пересылки почты) – отвечает за непосредственную пересылку электронной почты между используемыми почтовыми серверами определенного пользователя; как правило, первый агент пересылки электронной почты MTA в определенной пересылочной цепочке получает информационное сообщение от почтового агента пользователя MUA, последний выполняет непосредственную передачу информационного сообщения к агенту доставки почты MDA; при этом, становится возможным практическая реализация отправки электронной почты по средствам практического использования smart host.

MDA (англ. Mail Delivery Agent – агент доставки почты) – отвечает за непосредственную доставку электронной почты конечному пользователю программной системы, которая будет работать электронными сообщениями.

MUA (Mail user agent – почтовый агент пользователя) – программа, которая в полной мере позволяет обеспечить пользовательский интерфейс прикладной программной системы, которая может отображать полученные электронные письма и предоставляющая соответствующие возможности отвечать, создавать, перенаправлять электронные письма в соответствующем информационном пространестве.

MRA (англ. Mail retrieve agent) – почтовый сервер, который забирает электронную почту с другого сервера по средствам использования соответствующих протоколов, которые предназначены для агента доставки почты MDA [15, с.206].

В случае практического использования выделенных серверов для хранения электронной почты пользователей всё информационное взаимодействие пользователя с сервером может происходить по средствам использования протоколов, которые не укладываются в описанную схему отправки сообщений.

Почтовые сервера обычно выполняют роль соответствующего агента пересылки почты MTA и агента доставки электронной почты MDA. Некоторые почтовые сервера (прикладные программы) выполняют роль как MTA, так и

MDA, некоторые подразумевают непосредственное разделение на два независимых сервера: сервер-MTA и сервер-MDA (при этом если для осуществления доступа к почтовому ящику используются различные программные протоколы.

Например, POP3 и IMAP, то MDA в свою очередь может быть реализован либо как единое прикладное программное приложение, либо как специальный набор программных приложений, каждое из которых будет отвечать за отдельный протокол передачи данных.

Взаимоотношения между MTA, MDA и MUA при передаче электронной почты может быть представлено схемой, (рисунок 5).

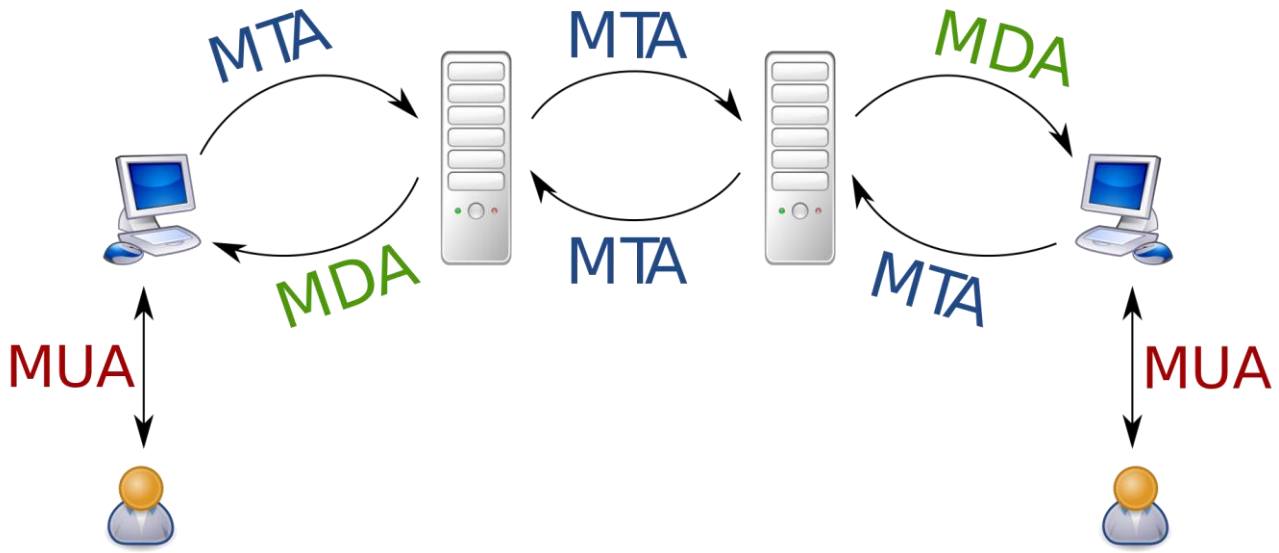

Рисунок 5 – Схема взаимоотношения между MTA, MDA и MUA при передаче электронной почты

Существуют несколько ключевых протоколов для осуществления работы с электронной почтой. Наиболее популярным является POP (Post Office Protocol). Его суть заключается в том, что прикладное программное обеспечение для осуществления работы с электронной почтой подключается к соответствующему серверу (удаленному серверу), скачивает электронными письма на персональный компьютер конечного пользователя, и они сразу же

становятся доступными без осуществления подключения к глобальной сети Internet.

Текущим стандартом, который обеспечивает доступ к электронным ящикам является протокол IMAP (Internet Message Access Protocol), который является намного быстрее и в большей степени соответствует тому, как сегодня используется глобальная сеть Internet. IMAP позволяет множеству пользователей подключаться к одному и тому же ящику и поддерживает их непосредственное соединение в течение всей сессии работы с электронным ящиком.

Web-браузеры получают необходимый доступ к электронной почте с помощью дополнительного протокола – HTTP, но в основе лежат всё те же протоколы передачи данных POP и IMAP.

В основном для непосредственного получения писем с сервера используются POP (актуальная версия POP3) и IMAP. Но для того чтобы отправить письмо, нужен другой протокол – SMTP (Simple Mail Transfer Protocol). Всё потому, что нельзя просто отправить письмо получателю. Его нужно отправить на сервер, с которого получатель это письмо скачает, используя IMAP и POP3.

Обычно протокол SMTP используют только для непосредственного осуществления непосредственной отправки электронных сообщений на почтовый сервер. Для получения электронных сообщений используются другие протоколы, такие как POP (Post Office Protocol) или IMAP (Internet Message Access Protocol) [18, с.309].

Принцип отправки и доставка электронной почты достаточно похож на принцип отправки и доставки классической (бумажной) почты только при помощи средств глобальной сети Интернет.

При таком сравнении SMTP-сервер выступает доставщиком электронной почты, который забирает письмо. А после с почты электронное письмо будет отправлено конечному получателю. И уже другой доставщик, в данном случае

это протоколы POP/IMAP, доставляет электронное письмо конечному получателю в соответствующий ящик.

Рассмотрим этот процесс подробнее, со стороны отправки электронных писем с web-сайта (однако основополагающий принцип будет схожим).

Так, пользователь программной системы создает электронное письмо и вызываете специальную функцию отправки сообщения.

Функция в свою очередь подключается к SMTP-серверу согласно указанным данным, к которым относятся следующие: порт, хост, логин, пароль почты.

SMTP-сервер получает сообщение электронной почты по средствам использования порта 25 (в основном используется именно данный порт) и выполняет пересылку его конечному получателю. Для этого сервер считывает и проверяет данные отправителя и конечного получателя, а после чего выполняет отправку электронного письма.

В тоже время, протокол SMTP не будет выполнять проверку или чтение содержимого электронного сообщения, он только выполняет непосредственную передачу электронной почты.

E-mail будет доставлен немедленно, если домен конечного получателя напрямую будет подключен к серверу. Иначе, электронная почта будет передана другому ближайшему серверу входящей почты при помощи использования протокола SMTP.

После чего web-сервер выполняет соединение с сервером конечного получателя (он будет определен по MX записи соответствующего домена конечного получателя электронного письма), который получит электронную почту и будет хранить ее.

В том случае, если сервер занят, то электронная почта будет перенаправлена на резервный сервер по протоколу SMTP. Если такого сервера не окажется, то сообщение будет перемещено в очередь, из которой периодически будет выполнятся задание на повторную доставку электронного письма.

В случае неудачи, после определенного количества попыток, электронная почта будет возвращена с соответствующей информацией об провале в доставке данного сообщения.

Последним шагом будет обрабатываться POP/IMAP, который заберет электронную почту от принимающего сервера и положит во «Входящие» получателя. Детализированная схема отправки почты по SMTP представлена на схеме, (рисунок 6)

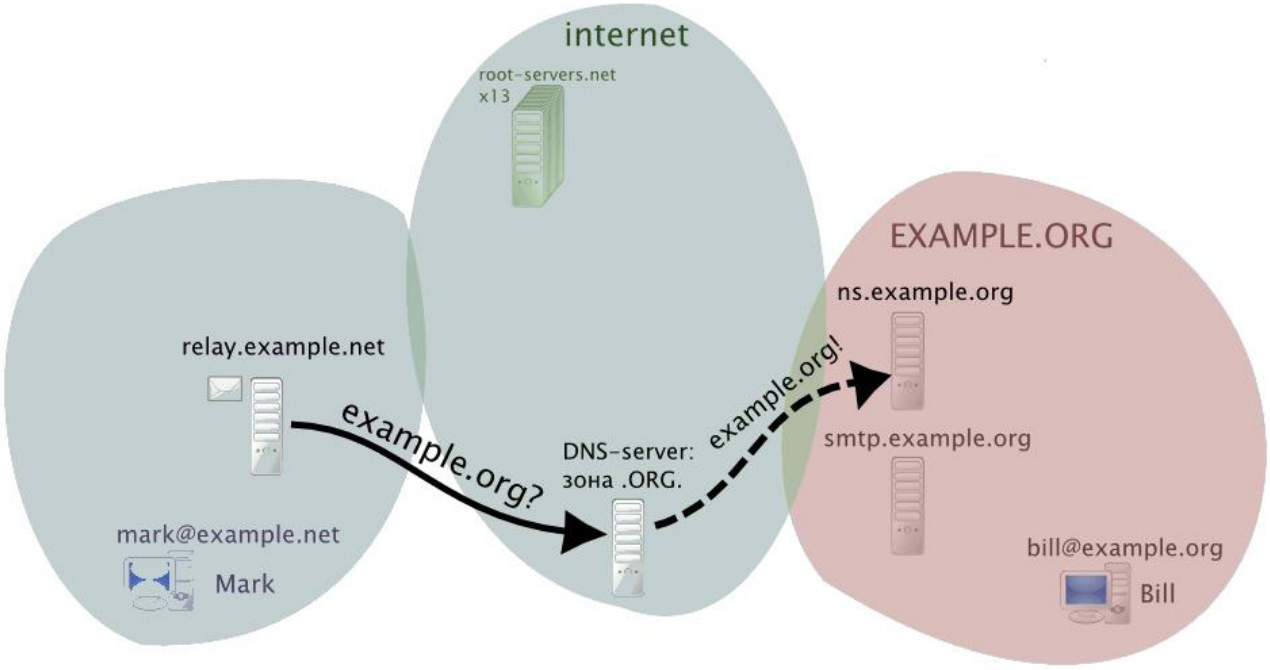

Рисунок 6 – Детализированная схема отправки почты по SMTP

Внутри определенной почтовой системы (обычно она находится в определенной организации или предприятии) может быть большое количество почтовых серверов, выполняющие как операции непосредственной пересылки почты внутри данной организации, так и другие, связанные с электронной почтой задачи, как, например: фильтрация спама, проверка вложений специализированным антивирусным программным обеспечением, обеспечение автоответа, выполнение архивации входящей/исходящей почты, осуществление операций доступа активным пользователям по средствам непосредственного

использования различных методов, среди которых можно выделить POP3 или ActiveSync.

ActiveSync представляет собой программу, которая позволяет установить синхронизированную связь между мобильным устройством и персональным компьютером, а также сервером, который работает под управлением Microsoft Exchange Server. С Exchange Server можно синхронизировать только PIM-данные (Почту/Календарь/Контакты/Задачи) посредством практического использования Microsoft Outlook.

Следует различать программу ActiveSync и Exchange ActiveSync (EAS), протокол синхронизации мобильного устройства с сервером электронной почты Microsoft Exchange поверх сетевого протокола HTTP/HTTPS. Технология Exchange ActiveSync используется такими почтовыми серверами и специализированными средствами для коллективной работы, включая Novell GroupWise и Lotus Domino [21, с.570].

Информационное взаимодействие между почтовыми серверами в рамках единой почтовой системы может быть, как подчиненным каким-то определенным правилам (например, в рамках такого взаимодействия может быть установлена необходимость использования DNS и набора правил маршрутизации электронной почты при помощи использования протокола SMTP), так и может быть описана необходимость следовать собственным правилам определенного предприятия или компании (которая использует прикладное программное обеспечение).

Рассмотрим более подробно, что именно выполняется в процессе отправки электронной почты почтовым клиентом. В примере пользователь Иванов, находящийся в домене example.org (ivanov@example.org), пишет определенное электронное письмо другому участнику информационного взаимодействия, который находится в домен example.com (kozlov@example.com). Для Иванова процесс непосредственной отправки электронной почты состоит из операции создания информационного сообщения

и нажатия на соответствующую кнопку «Отправить» в используемом почтовом клиенте.

Почтовый клиент выполнит соединение с агентом пересылки почты MTA при помощи практического использования протокола SMTP и сначала сообщит свои учетные данные. Авторизовав необходимого пользователя, агентом пересылки почты MTA примет электронное сообщение и будет пытаться доставить его дальше по цепочке передачи данных. Схема работы почтовых серверов без осуществления непосредственной привязки представлена на схеме, (рисунок 7).

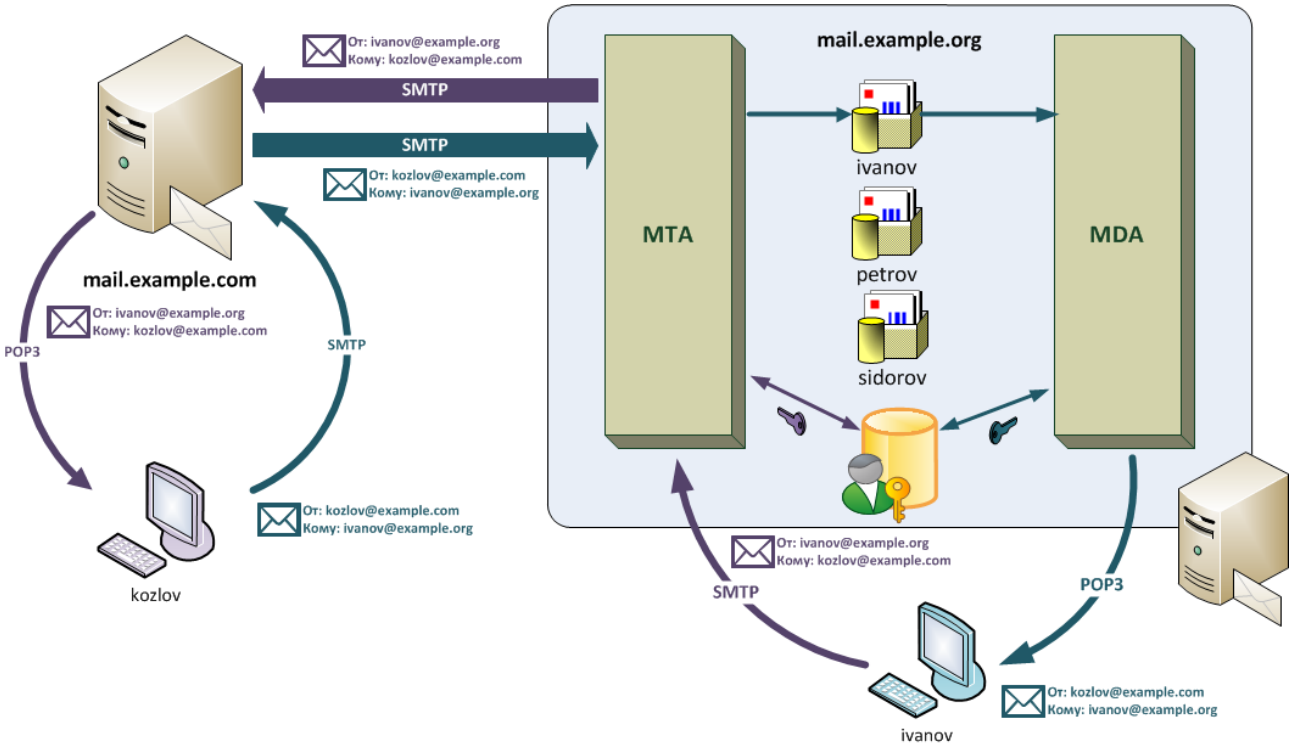

Рисунок 7 – Схема работы почтовых серверов без привязки

В тоже время, необходимо отметить, что операции авторизации не является обязательной процедурой для агента пересылки почты MTA, но без выполнения авторизации нельзя получить роли, т.е. без осуществления авторизации любой пользователь может воспользоваться почтовым сервером для пересылки электронной почты на свое усмотрение.

Сегодня, открытые ролей может быть из-за наличия различных ошибок в настройках почтового сервера. Однако вполне может быть допустима такая ситуация, когда агент пересылки почты MTA без авторизации будет принимать почту от своих доверенных пользователей, например из локальной вычислительной сети определенного предприятия.

Для авторизации агента пересылки почты MTA может воспользоваться собственным списком конкретных пользователей, использовать системный список, имеющиеся списки пользователей LDAP или другие дополнительные списки из AD.

Также существует еще один способ, который предполагает, что авторизация POP будет выполнена прежде чем SMTP, когда пользователь перед непосредственной отправкой почты будет авторизована на MDA, который, в свою очередь выполнить подтверждение аутентификации конкретного пользователя для агента пересылки почты MTA.

Следующим шагом агента пересылки почты MTA выполняется анализ служебной информацию электронного письма, определяя имеющийся домен конечного получателя, если он будет относиться к доменам, которые обслуживаются данным агентом пересылки почты MTA, производится многокритериальный поиск конкретного получателя и электронное письмо помещается в его почтовый ящик.

Если домен конечного получателя не будет обнаружен агентом пересылки почты MTA, тогда будет сформирован специализированный DNS-запрос, который запросит необходимые MX-записи для данного домена. MX-запись это вид DNS-записи, которая включает в себя соответствующие имена почтовых серверов, которые обрабатывают входящую электронную почту для конкретного домена. MX-записей может быть больше одной, в этом случае агент пересылки почты MTA будет последовательно устанавливать соединение, начиная с сервера у которого наибольший приоритет [28, с.760].

При отсутствии необходимой MX-записи будет выполнен структурированный запрос A-записи (запись адреса, которая будет в полной

мере совпадать определенное доменное имя с IP-адресом получателя) и будет выполнена соответствующая попытка выполнить доставку почту на имеющийся там хост.

Затем, сервер, обслуживающий домен example.com, аналогичные действия и выполняет необходимые попытки по передаче электронной почты первичному почтовому серверу. Получив входящее электронное сообщение агент пересылки почты MTA, как и в случае с обработкой локального отправителя, выполняет соответствующую проверку домена конечного получателя, если он входит в число обслуживаемых агентом пересылки почты MTA, обработка электронного сообщения будет продолжена, иначе сервер откажется принимать электронную почту данного отправителя.

После выполнения соответствующей проверки домена будет проверен конечный получатель, если он будет находиться в списке активных пользователей, информационное сообщение будет доставлено в его почтовый ящик, в противном случае будут возможны следующие варианты: отказ от непосредственного приема сообщения или прием электронного сообщения в общий почтовый ящик (например, ящик администратора). С одной стороны такая настройка позволяет увеличить число получаемого спама, с другой позволяет не потерять важные письма, к которых есть ошибки в написании конечного адреса.

2.3 Разработка алгоритмов функционирования программной системы

Разработка алгоритмов функционирования программной системы включает процесс алгоритмизации предметной области и описание используемых функций выбранного языка программирования.

Для начала нужно подключить подходящий модуль Python – Smtplib, который поставляется вместе с Python, для этого необходимо ввести следующую команду:

```
import smtplib
```

Модуль smtplib определяет объект сеанса клиента SMTP, который можно использовать для отправки почты на любой компьютер подключенный к глобальной сети Интернет с демоном SMTP или ESMTP.

class smtplib.SMTP([host[, port[, local_hostname[, timeout]]]])

Экземпляр SMTP инкапсулирует соединение SMTP. У него есть методы, которые поддерживают полный набор операций SMTP и ESMTP. Если указаны необязательные параметры хоста и порта, во время инициализации вызывается метод SMTP connect () с этими параметрами. Если указано, local_hostname используется как полное доменное имя локального хоста в команде HELO / EHLO. В противном случае локальное имя хоста можно найти с помощью socket.getfqdn ().

Для обычного использования необходимо использовать только методы инициализации / подключения, sendmail () и SMTP.quit ().

class smtplib.SMTP_SSL([host[, port[, local_hostname[, keyfile[, certfile[, timeout]]]]]])

Экземпляр SMTP_SSL ведет себя точно так же, как экземпляры SMTP. SMTP_SSL следует использовать в ситуациях, когда требуется SSL с начала подключения, а использование starttls () не подходит [23, с.34].

keyfile и certfile также являются необязательными и могут содержать закрытый ключ в формате PEM и файл цепочки сертификатов для соединения SSL.

Необязательный параметр timeout указывает время ожидания в секундах для операций блокировки, таких как попытка подключения (если не указано, будет использоваться глобальная настройка времени ожидания по умолчанию). Если время ожидания истекло, то значение socket.timeout повышается.

class smtplib.LMTP([host[, port[, local_hostname]]])

Протокол LMTP, который очень похож на ESMTP, в значительной степени основан на стандартном SMTP-клиенте. Обычно для LMTP используются сокеты Unix, поэтому метод connect () должен поддерживать это, а также обычный сервер host: port. local_hostname имеет то же значение, что и для класса SMTP. Чтобы указать сокет Unix, необходимо использовать абсолютный путь к хосту, начиная с ‘/’.

Для получения справки по модулю можно воспользоваться специальной функцией help:

help(smtplib)

Прежде всего, нужно создать соответствующий объект smtplib, который можно рассматривать как определенный портал, который позволяет дать соответствующий доступ к подключению и доступ к различным инструментам для работы с ним из прикладного программного модуля smtplib.

В данном случае функция, возвращающая функция необходимый информационный объект, принимает два аргумента или параметра. Первый параметр включает соответствующее доменное имя, то есть необходимый адрес электронной почты, который начинается с «smtp», как будет представлено далее. Второй параметр – это номер используемого порта, к которому будет выполнено непосредственного подключения на сервере электронной почты.

Модуль smtplib использует специальный протокол RFC 821 для SMTP. Кроме этого, множество почтовых провайдеров используют те же порты подключения. Он почти всегда принимает значение 587 в соответствии со стандартом шифрования TLS:

smtpObj = smtplib.SMTP('smtp.gmail.com', 587)

Переменная smtpObj является объектом типа SMTP, которую можно назвать в любом удобном виде.

Следующее, что нужно сделать для установления соединения, это «сказать» SMTP-объекту, что сообщение должно быть зашифровано. Для этого следует воспользоваться следующим оператором:

SMTP.starttls([keyfile[, certfile]])

Конечный вид установления соединения будет иметь следующее представление в виде такого оператора

smtpObj.starttls()

Это позволит сообщить Gmail, что отправляемое соединение шифровалось при помощи практического использования протокола TLS (Transport Layer Security), который на сегодняшний день является общепринятым стандартом для осуществления эффективной интернет-коммуникаций.

Сам по себе протокол TLS не относится к криптографическим алгоритмам, он скорее может сообщить, что именно необходимо использовать шифрование или как именно должно быть выполнено соединение.

Перед тем, как начать информационный обмен оперативными данными по средствам использования протокола TLS, клиент и почтовый сервер должны согласовать соответствующие параметры соединения, а именно: версия используемого протокола, необходимый способ шифрования передаваемых данных, а также проверить имеющиеся сертификаты, если это будет необходимо.

Разберём подробнее каждый отдельный шаг данной программной процедуры в области информационного обмена при отправке электронного сообщения.

Так как протокол TLS работает над протоколом TCP, для начала между клиентом и почтовым сервером должно быть установлено соответствующее TCP-соединение.

После необходимой установки TCP, клиент выполняет отправку на почтовый сервер спецификации в виде текста (а именно версию используемого протокола, которую он хочет использовать, поддерживаемые методы шифрования оперативных данных, etc).

Сервер выполняет утверждение версии используемого протокола передачи данных, выбирает необходимый способ шифрования данных из некоторого списка, прикрепляет свой уникальный сертификат и выполняет отправку ответа конечному клиенту.

Способ шифрования и версия протокола на данном этапе будет считаться утверждёнными, клиент выполняет проверку присланного сертификата и сможет инициировать либо RSA, либо выполнить обмен ключами по Диффи-Хеллману, в зависимости от установленных параметров обработки данных почтовым сервером.

Сервер выполняет обработку присланных клиентом информационных сообщений, сверяет MAC, и выполняет проверку клиента и формирует заключительное (‘Finished’) сообщение в зашифрованном виде.

Клиент выполняет расшифровку полученного сообщения, сверяет MAC, и если всё будет хорошо, то соединение будет считаться установленным и начнется обмен данными почтовыми приложениями.

Следующий шаг алгоритмизации программной системы заключается в авторизации пользователя.

Для того чтобы авторизоваться пользователю, нужно написать следующий оператор:

```
SMTP.login(user, password)
```

При помощи использования данной команд можно зайти на SMTP-сервер, который требует соответствующей аутентификации. Аргументами данной функции являются следующие: имя конкретного пользователя и его пароль для выполнения аутентификации на почтовом сервере. В случае, если в этом сеансе не было никаких других команд инициализации EHLO или HELO, данный метод сначала выполняет попытку запустить ESMTP EHLO. Этот метод вернется нормально, если аутентификация будет выполнена успешно, или может вызвать некоторые исключения, на основании которых можно принимать соответствующие решения.

Рассмотрим основные исключения для SMTP.login языка программирования Python.

SMTPHeloError

Сервер не принял поданную ему комбинацию имени пользователя и пароля.

SMTPException

Не найдено подходящего метода аутентификации запроса программной системы.

Конечный вид аутентификации будет иметь следующее представление в виде следующего оператора:

smtpObj.login('tt0065746@gmail.com','12aAs123')

Для непосредственной отправки сообщения можно воспользоваться следующим оператором:

SMTP.sendmail(from_addr, to_addrs, msg[, mail_options, rcpt_options])

Рассмотрим, каким образом происходит работа с SMTP.sendmail. Обязательными аргументами являются строка адреса RFC 822, список строк адреса RFC 822 (пустая строка будет рассматриваться как список с 1 адресом) и строка сообщения.

Если в этом сеансе не было никаких предыдущих команд EHLO или HELO, этот метод сначала пытается выполнить ESMTP EHLO. В случае сбоя EHLO будет пробоваться HELO, а параметры ESMTP будут подавлены [26, с.430].

Данный метод будет возвращаться в нормальном виде, если информационное письмо будет принято хотя бы для одного конкретного получателя. В другом случае это вызовет соответствующее исключение. То есть, если данный метод не будет вызывать исключение, то кто-то должен будет получить передаваемую почту.

Каждая отдельная запись включает кортеж с программным кодом соответствующей ошибки протокола SMTP и сопровождающее информационное сообщение об ошибке, которое отправляется почтовым сервером.

Данный метод может вызвать исключения, среди которых можно отметить следующие:

SMTPRecipientsRefused

Всем получателям было отказано сообщение. Никто не получил передаваемую электронную почту. Атрибут конечного получатели объекта исключения - это словарь с оперативной информацией об отклоненных получателях (например, тот, который был возвращен, когда был принят хотя бы один получатель).

SMTPHeloError

Сервер не ответил должным образом на соответствующее приветствие HELO.

SMTPSenderRefused

Если не указано иное, соединение будет открыто даже после возникновения исключения.

Конечный вид оператора отправки сообщения будет иметь следующий вид, представленный далее

smtpObj.sendmail("tt0065746@gmail.com","michaels.byrnel@vice.com","The end my freand!")

Для завершения соединения достаточно использовать следующую команду:

SMTP.quit()

Завершить сеанс SMTP и закрыть соединение подключения можно при помощи команды quit(). Вернуть результат команды SMTP QUIT.

smtpObj.quit()

Полный программный код программы представлен в Приложении. Таким образом, были продемонстрированы основные программные решения, использовав которые можно получить желаемую программу рассылки e-mail.

3 Методические указания по использованию системы

3.1 Системные требования к аппаратной части

Инфраструктура организации включает сетевое оборудование, рабочие места пользователей, периферийное оборудование. На данный момент в организации оборудовано достаточное количество рабочих мест для сотрудников, так же сотрудники имеют возможность работать удаленно с корпоративными данными.

Техническое обеспечение типового рабочего места: процессор Intel(R) corei3, 3.00 ГГц, оперативная память 4 GB, HDD 240 GB, сетевая карта 100 Мб/с.

Техническое обеспечение рабочего места сотрудника отдела ИТ: используются компьютеры с процессором Intel(R) corei5, с частотой 4.20 ГГц, ОЗУ 8 GB, SSD 500 GB, сетевая карта 250 Мб/с.

Среди вспомогательных компонентов вычислительной сети можно выделить:

– сервер публичного каталога LDAP протокол прикладного уровня для доступа к службе каталогов X.500, разработанный IETF как облегчённый вариант разработанного ITU-T протокола DAP. LDAP – относительно простой протокол, использующий TCP/IP и позволяющий производить операции аутентификации (bind), поиска (search) и сравнения (compare), а также операции добавления, изменения или удаления записей [19, с.230]. Обычно LDAP-сервер принимает входящие соединения на специальный порт 389 по сетевым протоколам TCP или UDP. Для LDAP-сеансов, инкапсулированных в SSL, обычно используется порт 636.

– сервер управления ISS Sete Protector - проприетарный набор серверов для нескольких служб Интернета от компании Microsoft. Основным компонентом IIS является веб-сервер, который позволяет размещать в Интернете сайты. IIS поддерживает протоколы HTTP, HTTPS, FTP, POP3, SMTP, NNTP;

– сервер управления средствами аутентификации - программно-аппаратный комплекс, реализующий различные методы аутентификации для различных приложений. Чтобы достичь максимального уровня защищенности этих процессов, необходима поддержка аппаратных модулей безопасности (Hardware Security Module; HSM).

– сервер сегмента - приложение, предназначенное для ответов на DNS-запросы по соответствующему протоколу. Также DNS-сервером могут называть хост, на котором запущено соответствующее приложение. Сервер HP Proliant DL560 G8 - 4xIntel XEON Eight Core E5-4650 2.70 GHz, 768GB DDR3 (48x16GB), 2xPS, Без HDD, 5x2,5' Tray for HDD (5 корзин в комплекте), HP Smart Array P420i.

– коммутаторы ядра ЛВС - осуществляют маршрутизацию пакетов с данными между отдельными сегментами сетевой инфраструктуры. Коммутатор SNR-S2995G-12FX входит в линейку управляемых L3 коммутаторов SNR, предназначен для использования на уровне агрегации в сетях операторов связи и корпоративных клиентов. Полностью аппаратные коммутация, маршрутизация и политики ACL гарантируют отсутствие задержек и потерь трафика. Современные ASIC обеспечивают работу на полной скорости всех портов устройства [22, с.730]. Аппаратная поддержка IPv6, прошедшая сертификацию IPv6 Phase II, позволяет строить сети IPv4/IPv6 DualStack. SNR-S2995G-12FX является идеальным решением в качестве коммутатора уровня агрегации для предоставления сервисов Triple Play в сетях операторского класса, а также для построения распределенного ядра корпоративных сетей;

– коммутатор сети - устройство, предназначенное для соединения нескольких узлов компьютерной сети в пределах одного или нескольких сегментов сети. Zyxel GS1900-48 - коммутатор со Smart-управлением, который обеспечивает гигабитную скорость и основные функции контроля [25, с.243]. Применение этого коммутатора существенно улучшает гибкость сетей бизнеса и возможность подключения к ним. Изделие оснащено функциями сокращения

энергопотребления, не имеет вентилятора и полностью поддерживает IPv6, что гарантирует готовность вашей сети к расширению в будущем, (рисунок 8).

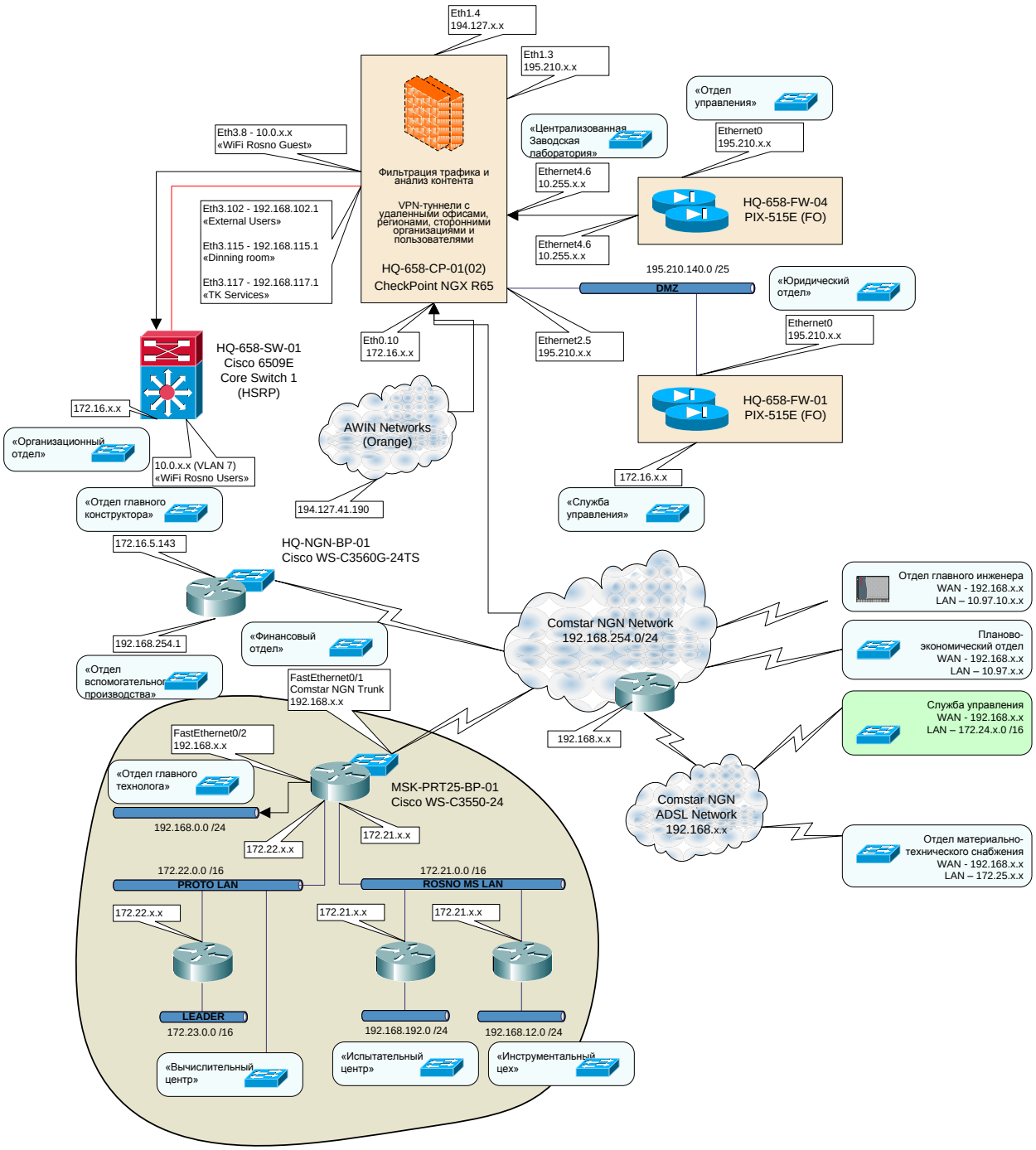

Рисунок 8 – Схематическое представление ИТ-инфраструктуры

Также, немаловажным компонентом вычислительной сети предприятия является программное обеспечение.

Microsoft SQL Server представляет собой систему управления реляционными базами данных от корпорации Microsoft. Microsoft SQL Server использует язык Transact-SQL, который был создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с дополнительными расширениями. Используется для работы с базами данных, размер которых от персональных до крупных баз данных масштаба предприятия; конкурирует с другими системами управления базами данных в этом сегменте рынка информационных технологий. Symantec EndPoint Protection представляет собой комплексный антивирус и высокоэффективный фаервол с несколькими уровнями безопасности. Антивирусная и превентивная защита, защита от сетевых угроз и эксплойтов нулевого дня, система предотвращения вторжений. Благодаря технологии Insight обеспечивает более раннее и точное обнаружение новых угроз по сравнению с решениями, реализующими проверку сигнатур.

Основные преимущества Symantec Endpoint Protection:

– благодаря технологии Insight обнаруживает новые и быстро видоизменяющиеся вредоносные программы;

– предотвращает эпидемии, связанные с информационной безопасностью, и снижает административную нагрузку;

– снижает совокупную стоимость обслуживания конечных систем [27, с.415].

Microsoft Exchange представляет собой программный продукт для обмена сообщениями и совместной работы.

Основные функции Microsoft Exchange: обработка и пересылка почтовых сообщений, совместный доступ к календарям и задачам, поддержка мобильных устройств и веб-доступ, интеграция с системами голосовых сообщений, поддержка систем обмена мгновенными сообщениями.

Таким образом, представленный комплекс программно-технических компонентов позволит раскрыть в полной мере функционал разработанной программной системы и обеспечит работу всего предприятия в целом.

3.2 Установка и настройка системы рассылки информационных сообщений

Установка и настройка системы рассылки информационных сообщений включает определенный перечень работ с почтой.

Для работы с почтой Gmail в сторонних почтовых клиентах нужно включить IMAP-доступ и настроить соответствующие параметры SMTP. С IMAP-доступом получать письма Gmail можно на нескольких устройствах сразу. При этом сообщения будут синхронизироваться в реальном времени. Для работы с почтой также можно использовать протокол POP.

Чтобы не потерять доступ к собственному аккаунту, нужно убедиться, что не превышается лимит по трафику для протокола IMAP: не более 2500 МБ в день на скачивание и не более 500 МБ в день на загрузку сообщений. Если нужно настроить IMAP-доступ к одному конкретному аккаунту на нескольких компьютерах, нужно делать перерывы после настройки на каждом отдельном устройстве.

Для настройки IMAP необходимо выполнить следующие шаги. Сначала нужно включить IMAP-доступ, затем открыть Gmail на персональном компьютере.

В правом верхнем углу нажать на значок Настройки и открыть вкладку Пересылка и POP/IMAP.

В разделе «Доступ по протоколу IMAP» нужно выбрать Включить IMAP и сохранить сделанные изменения.

Следующий шаг предполагает непосредственное изменение SMTP и других параметров в почтовом клиенте. Далее нужно указать в почтовом клиенте соответствующие данные из табл. 1. Дополнительную справочную информацию о параметрах IMAP для почтового клиента можно подобрать в глобальной сети Интернет.

Таблица 1 – Настройка клиента почтового ящика

№ п/п	Наименование параметра	Значение
1	Сервер входящей почты (IMAP)	imap.gmail.com Требуется SSL: да Порт: 993
2	Сервер исходящей почты (SMTP)	smtp.gmail.com
3	Полное или отображаемое имя	Имя
4	Учетная запись, имя пользователя или адрес электронной почты	Адрес электронной почты полностью
5	Пароль	пароль Gmail

Для использования почтового POP-клиента для работы с электронными письмами Gmail необходимо следующее. Сообщения Gmail можно получать через сторонние почтовые клиенты, которые поддерживают протокол POP, например, Microsoft Outlook. Для этого необходимо выполнить следующие настройки. Microsoft Outlook представляет собой персональный информационный менеджер с дополнительными функциями почтового клиента и Groupware компании Microsoft.

Кроме того, Microsoft Outlook позволяет отслеживать работу с оперативными документами офисного пакета Microsoft Office для автоматического составления дневника работы над заданиями и корпоративными назначениями.

Outlook может использоваться как отдельное прикладное программное приложение, так и выступать в роли специализированного клиента для почтового сервера Microsoft Exchange Server, что предоставляет дополнительные функциональные возможности для осуществления совместной работы нескольких пользователей в рамках одной конкретной организации: возможность использования общих почтовых ящиков, папок задач, календарей,

конференций, планирования и резервирования времени общих встреч, согласование документов.

Первое, нужно убедиться, что нужен именно POP-клиент. Для работы с почтой Gmail в сторонних клиентах можно использовать протокол IMAP или POP. По протоколу IMAP можно работать с почтой Gmail на нескольких устройствах сразу. Письма в этом случае синхронизируются в реальном времени. Доступ по протоколу POP можно включить только на одном компьютере. При этом письма не будут синхронизироваться в реальном времени – они будут скачиваться в клиент с той частотой, которую вы укажете.

Далее, необходимо настроить доступ по протоколу POP. Для этого сначала нужно настроить POP в Gmail. Для этого нужно открыть клиент (например, Microsoft Outlook) и задать следующие параметры, представленные в табл. 2.

Таблица 2 – Настройка клиента почтового ящика

№ п/п	Наименование параметра	Значение
1	Сервер входящей почты (POP)	pop.gmail.com Требуется SSL: да Порт: 995
2	Сервер исходящей почты (SMTP)	smtp.gmail.com
3	Время ожидания сервера	Более 1 минуты (рекомендуется 5 минут)
4	Полное или отображаемое имя	Имя
5	Учетная запись, имя пользователя или адрес электронной почты	Адрес электронной почты
6	Пароль	пароль Gmail

Таким образом, были представлены основные настройки, которые необходимо выполнить для работы с программной системой.

3.3 Порядок работы с программной системой

Для начала работы с программой необходимо сформировать список рассылки e-mail адресов и записать их в текстовый файл, содержание которого представлено следующим образом, (рисунок 9). Файлы e-mail адресов и письма рассылки должны быть в папке, в которой находится запускаемая программа.

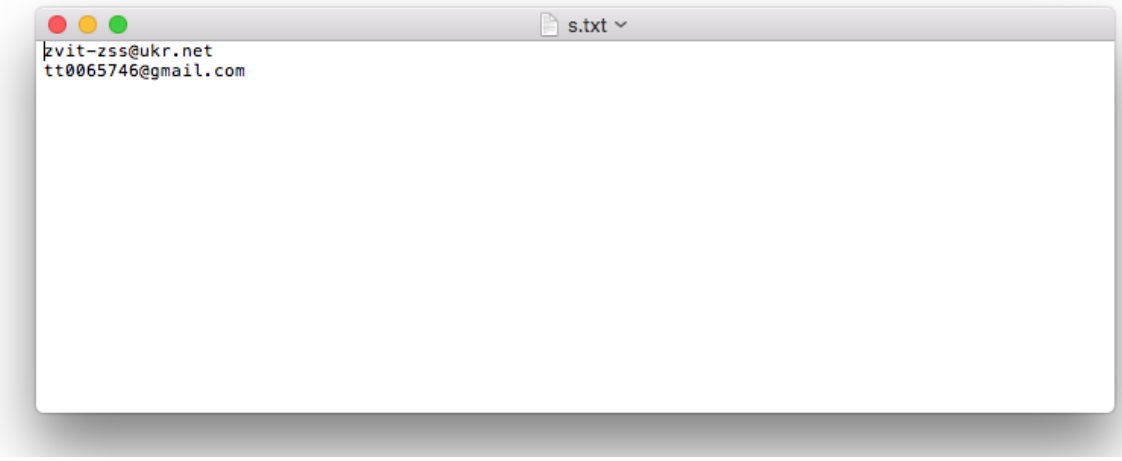

Рисунок 9 – Содержание файла рассылки e-mail адресов

Далее, необходимо составить письмо рассылки в виде тактового файла, содержание которого представлено далее, (рисунок 10).

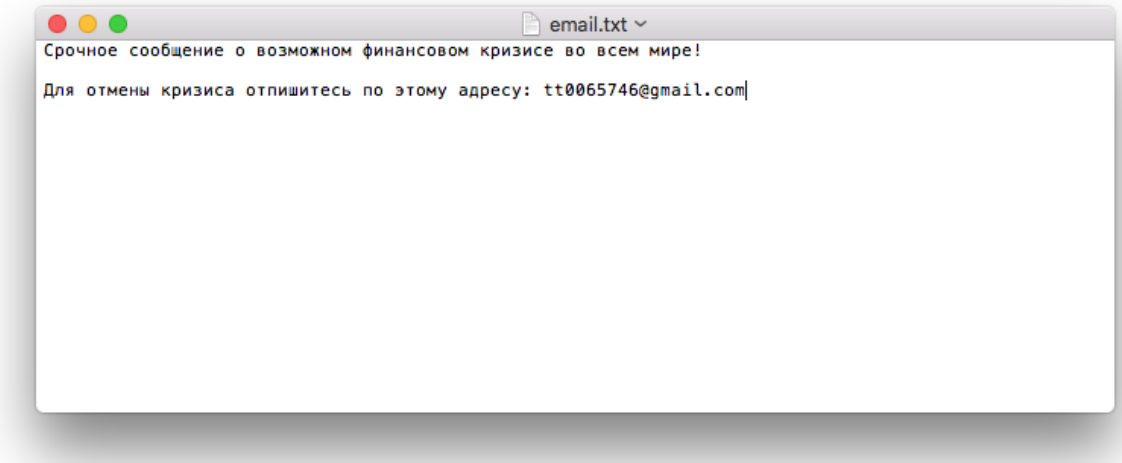

Рисунок 10 – Содержание файла письма рассылки e-mail

Запуск программы можно выполнить при помощи программы TextMate или среды Python. Интерфейс программной среды разработки представлен на рисунке, (рисунок 11).

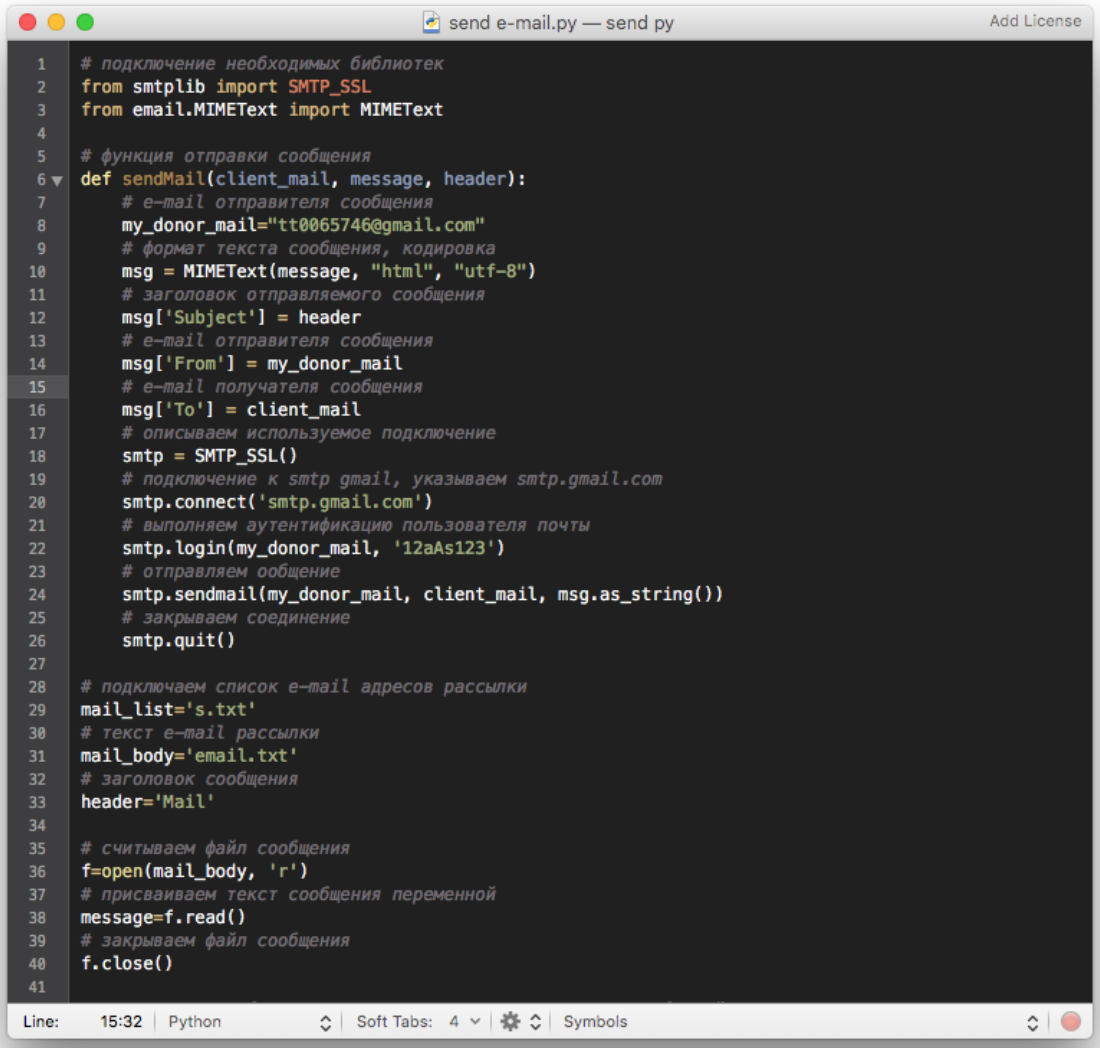

Рисунок 11 – Интерфейс среды разработки

После запуска программы на выполнение пользователю будут выданы соответствующие сообщения, (рисунок 12).

В сообщениях программы представлен список e-mail адресов на которые были отправлены сообщения. Список сообщений можно изменить на свое усмотрение, количество неограниченно, при большом количестве e-mail

адресов время работы программы будет увеличено в соответствующее число раз.

Рисунок 12 – Результат выполнения программы

Проверить результат выполнения разработанной программы, можно посмотрев соответствующий почтовый ящик, указанный в файле e-mail рассылки. Так, сообщение было отправлено на адрес tt0065746@gmail.com, (рисунок 13).

В разработанной программе присутствуют полезные функции для ведения email-маркетинга:

– многоканальность;

– готовые шаблоны писем;

– автоматические рассылки;

– формы подписки;

– отдел поддержки;

– тестирование рассылок.

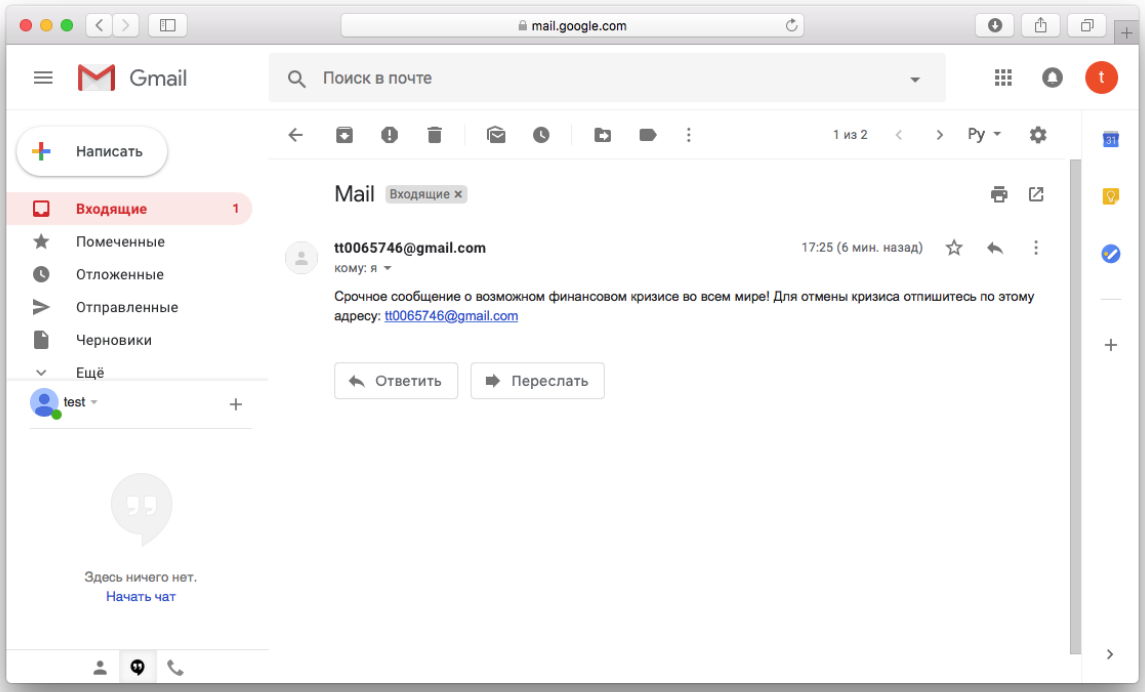

Рисунок 13 – Результат приема сообщения

Таким образом, разработка программы e-mail рассылки позволила ознакомиться с основными возможностями языка программирования Python, реализовать актуальную задачу информационного взаимодействия и наметить приоритеты для дальнейшей работы в области программирования на языке программирования Python.

Заключение

В процессе выполнения дипломной работы были получены следующие результаты. Установлено, что информационная система представляет собой систему, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы, которые обеспечивают и распространяют информацию. Проектирование информационной системы предполагает реализацию комплекса работ с пока несуществующими объектами и направлено на непосредственное создание информационной системы в области: обработки структурных объектов будущей базы данных; написания прикладных программ, которые предназначены обеспечивать взаимодействие пользователя с системой в процессе выполнения структурированных запросов к данным; выполнения оперативного учёта функционирования конкретной среды функционирования информационной системы.

Методология создания информационных систем заключается в организации процесса разработки информационной системы и обеспечении непосредственного управления этим процессом для того, чтобы в полной мере гарантировать выполнение всех установленных требований, как к самой информационной системе, так и к характеристикам всех стадий процесса разработки.

Анализ методологий разработки программных систем показал наличии большого количества подходов, среди которых были выделены такие методологии: RAD; ASD; FDD. Каждая из перечисленных методологий имеется свои особенности. Наиболее подходящей для проекта автоматизации поставленного комплекса задач является первая методология RAD.

Современный рынок информационных технологий предлагает массу программных решений рассылки сообщений. Среди предложений выделяются следующие программы: ePochta Mailer; ePochta Studio; AMS Enterprise. Анализ существующих программных систем показал наличие в них большого

количества функциональных возможностей, воспользовавшись которыми можно решить часть поставленного комплекса задач. В тоже время, для решения задачи рассылки информационных сообщений в условиях предприятия недостаточно использовать готовый продукт, а необходимо использовать дополнительный модуль, который позволит использовать корпоративные данные для своей работы.

Реализация приложения для рассылки информационных сообщений с учетом требований предприятия была выполнена на базе использования средств языка программирования Python. Данный язык позволяет реализовать необходимый спектр функциональных возможностей разрабатываемой программной системы для рассылки информационных сообщений.

В процессе реализации программной системы были изучены и реализованы в необходимой степени следующие программные компоненты: MTA; MDA; MUA; MRA.

MTA – отвечает за пересылку электронной почты между почтовыми серверами. MDA – отвечает за непосредственную доставку электронной почты конечному пользователю программной системы. MRA – почтовый сервер, который забирает электронную почту с другого сервера по соответствующим протоколам, предназначенным для агента доставки почты MDA.

В качестве практической части дипломной работы была разработана программа e-mail рассылки средствами языка программирования Python.

Рассылку e-mail сообщений была организована на следующих положениях: подключение к серверу электронной почты и авторизация пользователя; формирование письма e-mail средствами текстового редактора, наполнение его необходимыми данными; формирование списка e-mail адресов рассылки, для чего целесообразно использовать отдельный файл с адресами; отправка сообщения при помощи специального скрипта на языке программирования Python.

Дальнейшим развитием программы может быть создание графического интерфейса для удобства работы с программой.

Список использованной литературы

1. Архитектура и проектирование программных систем : монография / С.В. Назаров. – 2-е изд., перераб. и доп. – М. : ИНФРА-М, 2018. – 374 с.
2. Архитектура ЭВМ : учеб. пособие / В.Д. Колдаев, С.А. Лупин. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 383 с.
3. Архитектура ЭВМ и вычислительные системы : учеб. / В.В. Степина. – М.: КУРС: ИНФРА-М, 2018. – 384 с.
4. Баранова, Е.К. Информационная безопасность и защита информации : учеб. пособие / Е. К. Баранова, А. В. Бабаш. - 3-е изд., перераб. и доп. – М. : РИОР ; М. : ИНФРА-М, 2017. – 322 с.
5. Башлы, П.Н., Боярчук А.Э. Архитектура ЭВМ и вычислительных сетей: практикум / П.Н. Башлы,А.Э. Боярчук. Ростов н/Д.: Российская таможенная академия, Ростовский филиал, 2017. – 114 с.
6. Безопасность и управление доступом в информационных системах : учеб. пособие / А.В. Васильков, И.А. Васильков. – М. : ФОРУМ : ИНФРА-М, 2017. – 368 с.
7. Бенкен, Е. PHP, MySQL, XML. Программирование для Интернета; БХВ-Петербург - М., 2017. – 336 с.
8. Борзунов, С.В. Практикум по параллельному программированию: учеб. пособие / С.В. Борзунов, С.Д. Кургалин, А.В. Флегель. – СПб.: БХВ, 2017. – 236 с.
9. Васильков, А.В. Безопасность и управление доступом в информационных системах [Текст]: учебное пособие / А.В. Васильков, И.А. Васильков. - Москва: ФОРУМ: ИНФРА-М, 2017. – 368 с.
10. Гагарина, Л.Г. Технология разработки программного обеспечения: учеб. пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Виснадул ; под ред. Л.Г. Гагариной. – М. : ИД «Форум» : ИНФРА-М, 2017. – 400 с.

11. Гвоздева, В.А. Основы построения автоматизированных информационных систем [Текст]: учебник. - Москва: ИД «ФОРУМ»: ИНФРА-М, 2017. – 320 с.
12. Глушаков, С. В., Коваль А.В., Черепнин С.А., Программирование на Visual C++, ФОЛИО, Москва, 2017. – 580 с.
13. Дунаев, В.В. HTML, скрипты и стили; БХВ-Петербург - М., 2017. – 527 с.
14. Информатика: программные средства персонального компьютера : учеб. пособие / В.Н. Яшин. – М. : ИНФРА-М, 2018. – 236 с.
15. Информатика : учебник / И.И. Сергеева, А.А. Музалевская, Н.В. Тарасова. – 2-е изд., перераб. и доп. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 384 с.
16. Информатика (курс лекций) : учеб. пособие / В.Т. Безручко. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 432 с.
17. Информатика и информационно-коммуникационные технологии (ИКТ) : учеб. пособие / Н.Г. Плотникова. – М. : РИОР : ИНФРА-М, 2018. – 124 с.
18. Информационные технологии в профессиональной деятельности : учеб. пособие / Е.Л. Федотова. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 367 с.
19. Информационные технологии управления : учебник / Б.В. Черников. – 2-е изд., перераб. и доп. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 368 с.
20. Колисниченко, Д. PHP и MySQL. Разработка Web-приложений; БХВ-Петербург - М., 2017. – 560 с.
21. Кузнецов, М., Симдянов И., Голышев С. PHP 5. Практика создания Web-сайтов; БХВ-Петербург - М., 2017. – 960 с.
22. Ляпин, Д.А. PHP – это просто. Начинаем с видеоуроков (+ CD-ROM); БХВ-Петербург - М., 2017. – 881 с.
23. Мартишин, С.А. Проектирование и реализация баз данных в СУБД MySQL с использованием MySQL Workbench: Методы и средства проектирования информационных систем и техноло / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - М.: Форум, 2018. - 61 с.

24. Никсон, Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS; Питер - М., 2017. – 204 с.
25. Программирование в алгоритмах / Окулов С.М., – 6-е изд., (эл.) - М.:Лаборатория знаний, 2017. – 386 с.
26. Программирование графики на С++. Теория и примеры : учеб. пособие / В.И. Корнеев, Л.Г. Гагарина, М.В. Корнеева. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 517 с.
27. Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с.
28. Прохоренок, Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера (+ CD-ROM); БХВ-Петербург - М., 2017. – 912 с.
29. Рудаков, А.В. Технология разработки программных продуктов: учеб. пособие для студ. учреждений сред. проф. образования – 11-е изд., стер. – М.: Издательский центр «Академия», 2017. – 208 с.
30. Скабцов, Н. Аудит безопасности информационных систем. – СПб.: Питер, 2018. – 272 с.

Приложение

Исходный код программы

```
# подключение необходимых библиотек языка программирования Python
from smtplib import SMTP_SSL
from email.MIMEText import MIMEText

# функция отправки информационного сообщения
def sendMail(client_mail, message, header):
    # e-mail отправителя информационного сообщения
    my_donor_mail="tt0065746@gmail.com"
    # формат текста информационного сообщения, его кодировка
    msg = MIMEText(message, "html", "utf-8")
    # заголовок отправляемого информационного сообщения
    msg['Subject'] = header
    # e-mail отправителя информационного сообщения
    msg['From'] = my_donor_mail
    # e-mail получателя информационного сообщения
    msg['To'] = client_mail
    # описываем используемое подключение для отправки сообщения
    smtp = SMTP_SSL()
    # подключение к smtp gmail, указываем smtp.gmail.com для отправки
    smtp.connect('smtp.gmail.com')
    # выполняем аутентификацию пользователя почты
    smtp.login(my_donor_mail, '12aAs123')
    # отправляем информационное сообщение
    smtp.sendmail(my_donor_mail, client_mail, msg.as_string())
    # закрываем соединение, используемое для отправки сообщения
    smtp.quit()
```

```
# подключаем список e-mail адресов рассылки информационных сообщений
mail_list='s.txt'
# текст e-mail рассылки информационных сообщений
mail_body='email.txt'
# заголовок информационного сообщения
header='Mail'

# считываем файл информационного сообщения
f=open(mail_body, 'r')
# присваиваем текст информационного сообщения переменной
message=f.read()
# закрываем файл информационного сообщения
f.close()

# отправляем сообщение, цикл для отправки нескольких сообщений
for mail in open(mail_list, 'r'):
    try:
        # воспользовавшись функцией, отправляем информационное сообщение,
        # передаем mail, message, header
        sendMail(mail, message, header)
        # если успешная отправка, выводим сообщение с адресом отправки
        print 'Send to: '+mail[:-1]
    except:
        # если не успешная отправка, выводим сообщение с адресом отправки
        print 'NOT send to: '+mail[:-1]
f.close()
```