Форма № Н-6.01

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіотехнічних систем літальних апаратів

Кафедра комп'ютерних систем та мереж

**КУРСОВИЙ ПРОЕКТ**

**(РОБОТА)**

з *Системного програмного забезпечення*

(назва дисципліни)

на тему: «*Поштовий клієнт (отримання пошти)»*

Студента *3* курсу *535а* групи

напряму підготовки *комп’ютерні системи та мережі*

спеціальності  *комп’ютерна інженерія*

*Бершадський Я.В.*

(прізвище та ініціали)

Керівник *ст. викладач кафедри 503*

*Бабешко Є.В.*

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала

Кількість балів: Оцінка: ECTS

Члени комісії

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали)

(підпис) (прізвище та ініціали

м. Харків – 2017 рік

РЕФЕРАТ

Курсовой проект: 76 с., 43 ил., 11 табл., 4 приложения, 7 источников.

Цель работы – разработка программного продукта, предназначенного для получения писем и распределения их по каталогам в зависимости от файла настроек с возможностью разархивирования прикреплённых архивов.

Задачами данной работы являются:

1. периодически, в фоновом режиме, получение почты согласно настройкам, указанным в конфигурационном файле;
2. распределение писем по каталогам в зависимости от настроек пользователя;
3. распаковка приложенных архивов в каталог в зависимости от отправителя;
4. настройка конфигурационного файла через графический интерфейс;
5. ведение лога работы в текстовом файле.

В результате выполнения данного проекта было разработано программное обеспечение, которое периодически обращается к почтовому серверу, получает новые сообщения и в зависимости от настроек пользователя распределяет их по каталогам. Также была реализована возможность автоматического разархивирования прикреплённых архивов.

ПОЧТОВЫЙ КЛИЕНТ, СЛУЖБА, ПОЧТОВЫЙ СЕРВЕР, IMAP, SMTP, POP3, SSL, TSL, ЛОГ-ФАЙЛ, КОНФИГУРАТОР.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

ПО – программное обеспечение

IMAP ­– протокол прикладного уровня для доступа к электронной почте

TLS (Transport Layer Security) – криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет.

SSL (Secure Sockets Layer) – криптографический протокол, обеспечивающие защищённую передачу данных между узлами в сети Интернет.

POP3 – протокол прикладного уровня для доступа к электронной почте

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 5](#_Toc483327535)

[ВВЕДЕНИЕ 6](#_Toc483327536)

[1 ПОСТАНОВКА ЗАДАЧИ 7](#_Toc483327537)

[1.1 Анализ предметной области 7](#_Toc483327538)

[1.2 Анализ существующих решений 7](#_Toc483327539)

[1.3 Постановка задачи 9](#_Toc483327540)

[2 ПРОЕКТИРОВАНИЕ 11](#_Toc483327541)

[2.1 Выбор архитектуры 11](#_Toc483327542)

[2.2 Диаграмма вариантов использования: 13](#_Toc483327543)

[2.3 Диаграмма последовательности 18](#_Toc483327544)

[2.4 Проектирование графического интерфейса 19](#_Toc483327545)

[3 РАЗРАБОТКА 24](#_Toc483327546)

[3.1 Реализация выбранной архитектуры в виде диаграммы пакетов 24](#_Toc483327547)

[3.2 Разработка диаграммы классов 25](#_Toc483327548)

[3.3 Общая диаграмма классов с отображением на архитектуру 26](#_Toc483327549)

[3.4 Разработка алгоритма работы программы 28](#_Toc483327550)

[3.5 Разработка методов классов 31](#_Toc483327551)

[4 ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ 34](#_Toc483327552)

[4.1 Верификация 34](#_Toc483327553)

[4.2 Тестирование 37](#_Toc483327554)

[ВЫВОДЫ 39](#_Toc483327555)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 40](#_Toc483327556)

[ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 41](#_Toc483327557)

[ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММ 45](#_Toc483327558)

[ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ОПЕРАТОРА 56](#_Toc483327559)

[ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ 67](#_Toc483327560)

# ВВЕДЕНИЕ

Данная работа направлена на реализацию возможности получения входной информации по e-mail для систем, которые по умолчанию получают эту информацию из файлов, находящихся в каталогах файловой системы (старые системы). Это может использоваться, например, для автоматического общения каких-либо 2-х систем. Где в одной системе будет стоять некоторого рода передатчик, а в другой, приёмник. Передатчик будет формировать в зависимости от каких-то условий данные, например, информацию о работе системы, информацию об ошибках, какие-то отчёты и т.д. А приемник (им и будет являться разрабатываемое программное обеспечение) принимает в “режиме реального времени” эти данные, которые в дальнейшем могут быть проанализированы и обработаны в соответствии с логикой работы данной системы.

Целью данной работы является разработка программного продукта, предназначенного для получения писем и распределения их по каталогам в зависимости от файла настроек с возможностью разархивирования прикреплённых архивов.

В результате выполнения данного проекта было разработано ПО, которое периодически обращается к почтовому серверу, получает новые сообщения и в зависимости от настроек пользователя распределяет их по каталогам. Также была реализована возможность автоматического разархивирования прикреплённых архивов.

# 1 ПОСТАНОВКА ЗАДАЧИ

## 1.1 Анализ предметной области

Задача данной курсовой работы заключается в создании службы, которая будет автоматически проверять почтовый ящик на наличие новых писем и хранить их в определённом месте. Разрабатываемая служба может быть использована для добавления возможности получения входной информации по e-mail для систем, которые по умолчанию получают эту информацию из файлов, находящихся в каталогах файловой системы (старые системы).

В данном проекте каналом связи будет почтовый сервер. Работа с которым может осуществляться с помощью протоколов IMAP, SMTP, POP3 и некоторых других. На рисунке 1.1 изображена схема работы почтового сервера при отправке и получении почты.

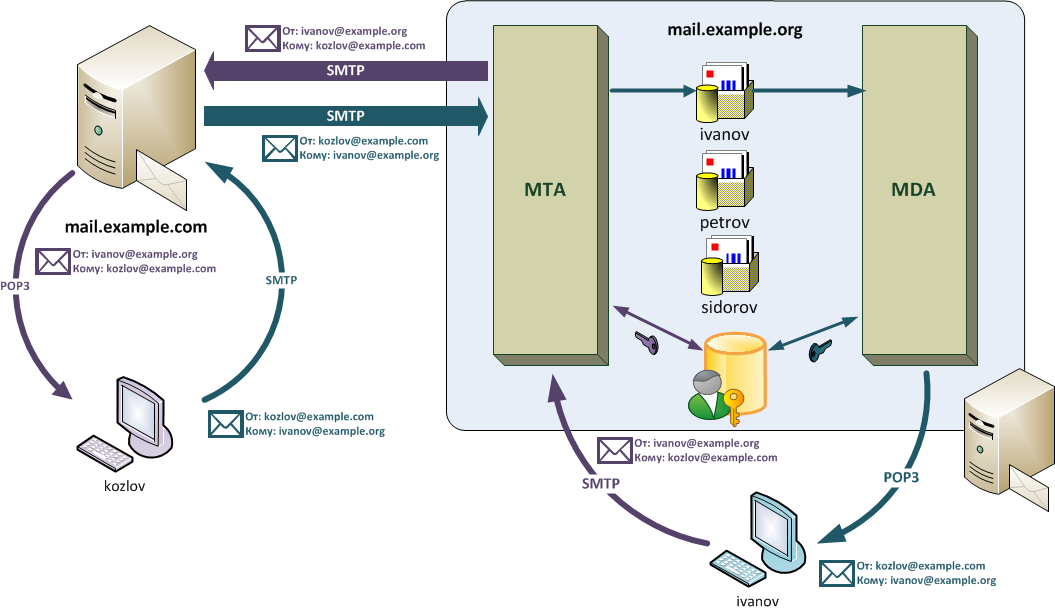


Рисунок 1.1 – Принцип работы почтового сервера [1].

Для решения поставленной задачи было решено выбрать IMAP и POP3 протоколы. Данный выбор обосновывается тем, что для реализации поставленной задачи необходимо только получение почты.

## 1.2 Анализ существующих решений

На сегодняшний день существуют огромное количество качественных и удобных почтовых клиентов, в том числе и от известных производителей программного обеспечения, такого как Microsoft. Наиболее близкими по функционалу к разрабатываемой службе являются современные полноценные почтовые клиенты, которые также периодически проверяют почтовый ящик пользователя. Полных аналогов ПО для данного проекта найти не удалось, поэтому было решено сравнить обычные почтовые клиенты.

Для сравнения возьмем такие почтовые клиенты, как Opera Mail и Mozilla Thunderbird.

Таблица 1.1 – Сравнительная характеристика почтовых клиентов Opera Mail и Mozilla Thunderbird

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Opera Mail** | **Mozilla Thunderbird** |
| Стоимость | бесплатная | бесплатная |
| Русский язык | ✔ | ✔ |
| HTML письма | ✔ | ✔ |
| Анти-спам | ✔ | ✔ |
| Вид диалога | - | ✔ |
| POP3 | ✔ | ✔ |
| IMAP4 | ✔ | ✔ |
| SMTP | ✔ | ✔ |

Оба почтовых клиента имеют удобный интерфейс, просты в работе и хорошо функционируют, однако можно отметить некоторые особенности каждого из них.

Основные особенности программы Opera Mail:

* почтовый клиент поддерживает протоколы smtp, imap, esmtp и pop3;
* программа работает с функцией фильтрации спама;
* удобный интерфейс, с удобным размещением окна сообщений;
* комфорт эксплуатации для пользователей операционной системы windows 7 сразу нескольких ящиков почты;
* программа windows оснащена тематической сортировкой электронных писем. Правила распределения сообщений по темам программирует сам пользователь.

Функциональные особенности данного клиента windows:

* mozilla thunderbird поддерживает протоколы smtp, imap и pops, что позволяет совершать эффективную передачу корреспонденции;
* возможность производить удаленный доступ к содержимому почтового сервера;
* простота настройки программного обеспечения;
* одновременное управление большим количеством почтовых ящиков и получение сообщений в одну папку из нескольких аккаунтов;
* программа работает с открытым доступом к передвижению данных из других почтовых программ посредством миграционного помощника;
* удобная система вкладок, структурированная адресная книга;
* мощная система фильтрации корреспонденции и проверки электронной почты;
* возможность работы по проверке орфографии.

Как видно из сравнительных характеристик эти почтовые клиенты, как и большинство современных не сильно отличаются по своему функционалу [2].

Данный курсовой проект не будет представлять из себя конкурентоспособную разработку, так как область применения такого рода ПО является очень узкой и используется для решения малого количества задач. Несмотря на это уникальными способностями будет возможность разархивирования прикреплённым к письму архивов, а также распределение писем по каталогам в зависимости от отправителя.

## 1.3 Постановка задачи

Разработать службу почтовый клиент, которая будет работать на получение писем и распределение их по каталогам в зависимости от файла настроек.

Данное ПО в фоновом режиме будет периодически обращаться к почтовому серверу, и проверять почтовый ящик на наличие новых писем. Если таковые присутствуют, то они будут сохраняться в отдельно созданный каталог в зависимости от отправителя. Т.е. служба будет создавать каталог для нового отправителя, и добавлять сообщение в уже существующий каталог для известных отправителей.

Также будет реализована возможность разархивирования прикреплённых к сообщению архивов. Все файлы из архива, будут автоматически выгружены в каталог с именем отправителя, в подкаталог с названием архива.

Пользователь должен будет указать в конфигурационном файле настройки программы такие как:

– корневой каталог, где будут храниться все остальные подкаталоги;

– путь к архиватору;

– сервер, порт, использование SSL/TLS, логин, пароль от почтового ящика пользователя;

– разархивировать ли автоматически прикреплённые архивы к сообщению.

Для изменения настроек конфигурационного файла будет реализован графический интерфейс, в то время как, у самой службы он будет отсутствовать.

Программа будет брать информацию о настройках из конфигурационного файла.

Поскольку ПО постоянно будет обращаться к удалённому серверу, то пользователю необходимо активное подключение к интернету.

# 2 ПРОЕКТИРОВАНИЕ

## 2.1 Выбор архитектуры

Архитектура – это высокоуровневая часть проекта приложения, каркас, состоящий из деталей проекта. Архитектуру также называют «архитектурой системы», «высокоуровневым проектом» и «проектом высокого уровня». Как правило, архитектуру описывают в единственном документе, называемом «спецификацией архитектуры» или «высокоуровневым проектом». Некоторые разработчики проводят различие между архитектурой и высокоуровневым проектом: архитектурой называют характеристики всей системы, тогда как высокоуровневым проектом – характеристики, описывающие подсистемы или наборы классов, но не обязательно в масштабе всей системы.

Продуманная архитектура предоставляет структуру, нужную для поддержания концептуальной целостности в масштабе системы. Она предоставляет программистам руководство, уровень детальности которого соответствует их навыкам и выполняемой работе. Она позволяет разделить работу на части, над которыми отдельные разработчики и группы могут трудиться независимо. Хорошая архитектура облегчает конструирование. Плохая архитектура делает его почти невозможным.

В первую очередь архитектура должна включать общее описание системы. Без такого описания вам будет трудно составить согласованную картину из тысячи деталей или хотя бы десятка отдельных классов [3].

Удачная архитектура позволяет:

– сделать возможным/Ускорить/Удешевить достижение бизнес-цели;

– достичь приемлемой производительности/увеличить производительность;

– сделать приложение легко расширяемым на случай, если это понадобится. Надо помнить, что, как правило, расширяемое приложение это простое приложение;

– уменьшить количество ошибок.

Процесс разработки структуры включает в себя создание общей инфраструктуры организации системы и управления ею, выбор протоколов и методов синхронизации и доступа к данным, распределение функций системы между компонентами, физическое распределение, объединение элементов проекта, масштабирование, оптимизацию производительности и выбор оптимальных вариантов среди доступных альтернатив.

Существует три типа системной архитектуры:

– одноуровневая;

– двухуровневая;

– многоуровневая.

В одноуровневой системной архитектуре все действия реализации графического интерфейса и решения задачи выполняются на одном уровне системы.

В двухуровневой архитектуре компоненты, отвечающие за графический интерфейс и прикладную логику, размещаются на клиентской машине и обращаются к общему источнику данных по сети.

В многоуровневой архитектуре презентационный сервис (первый уровень), прикладная логика (второй уровень) и сервис данных (третий уровень) отделены друг от друга.

Для данной работы было решено выбрать трёхуровневую архитектуру проекта (рисунок 2.1).

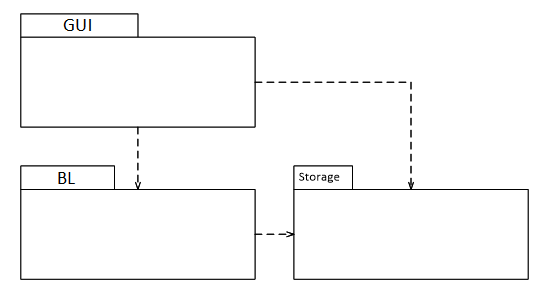


Рисунок 2.1 – Архитектура данного проекта

На рисунке 2.1 представлена архитектура системы, которая включает GUI – графический интерфейс пользователя; BL-бизнес логика; Storage –хранилище на котором хранятся данные.

Это решение обосновано тем, что данная архитектура наиболее подходит для решения поставленной задачи, поскольку мы имеем графический интерфейс, с которым пользователь взаимодействует напрямую (для настройки конфигурационного файла); логику проекта, т.е. служба, которая будет выполнять основные функции разрабатываемого программного продукта; и, конечно же, хранилище данных, в роле которого выступает конфигурационный файл, расположенный в файловой системе ПК на котором будет запущенна программа.

## 2.2 Диаграмма вариантов использования:

**UML –** унифицированный язык моделирования – это язык графического описания для объектного моделирования в области разработки программного обеспечения. UML является языком широкого профиля, это открытый стандарт, использующий графические обозначения для создания абстрактной модели системы, называемой UML моделью. UML был создан для определения, визуализации, проектирования и документирования в основном программных систем [4].

Главные цели в разработке UML:

– предоставить пользователям готовый к использованию выразительный язык визуального моделирования, позволяющий им разрабатывать осмысленные модели и обмениваться ими;

– предусмотреть механизмы расширяемости и специализации для расширения базовых концепций;

– обеспечить независимость от конкретных языков программирования и процессов разработки;

– обеспечить формальную основу для понимания этого языка моделирования (язык должен быть одновременно точным и доступным для понимания, без лишнего формализма);

– стимулировать рост рынка объектно-ориентированных инструментальных средств.

Диаграммы вариантов использования (Use Case) показывают взаимодействия между вариантами использования и действующими лицами, отражая функциональные требования к системе с точки зрения пользователя.

Цель построения – документирование функциональных требований в общем виде (требование – простота). Вариант использования - последовательность действий (транзакций), выполняемых системой в ответ на событие, инициируемое некоторым внешним объектом (действующим лицом). Вариант использования описывает типичное взаимодействие между пользователем и системой и отражает представление о поведении системы с точки зрения пользователя [5].

В процессе разработки были выделены следующие роли:

1) «User» – Пользователь;

2) «FileSysem» – Конфигурационный файл в файловой системе;

3) «MailServer» –Почтовый сервер.

«User» – актор который представляет из себя пользователя программы. «FileSystem» – актор – конфигурационный файл в файловой системе ПК. «MailPostServer» – актор – почтовый сервер, к которому будет обращаться данное ПО.

Исходя из вышеперечисленных суждений, диаграмма вариантов использования имеет вид представленный на рисунке 2.2.

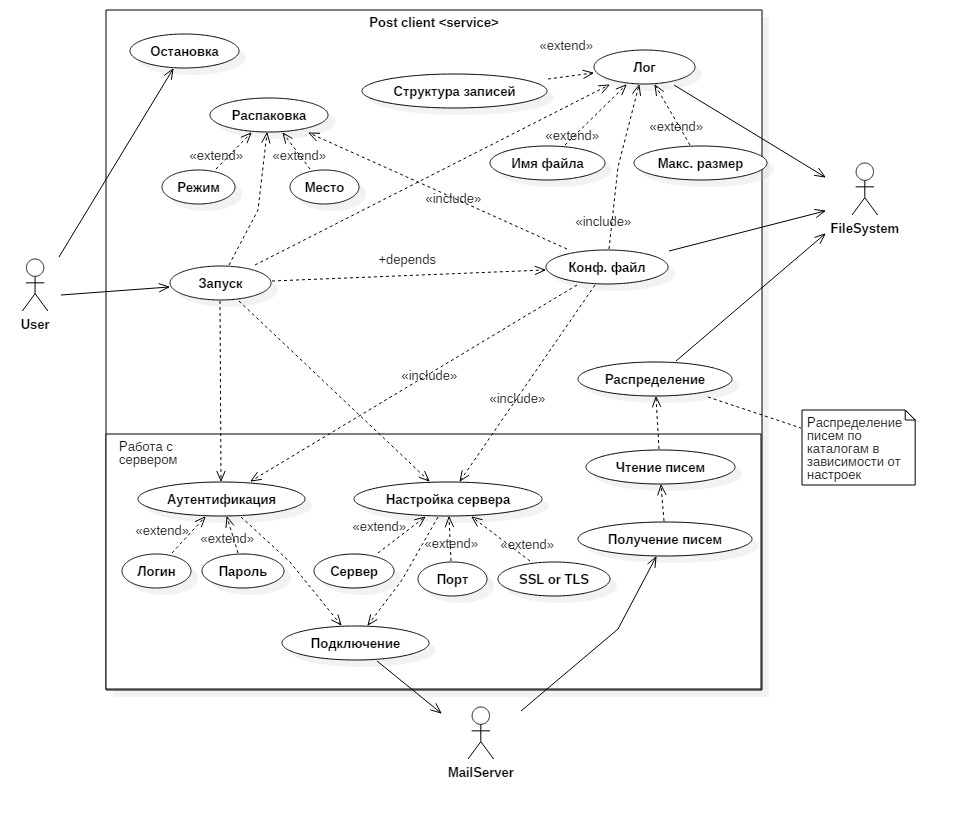


Рисунок 2.2 – Диаграмма вариантов использования

Исходя из анализа построенной диаграммы использования, а также функций, которые должна выполнять программа, были выделены акторы и варианты использования, основные из которых, представлены в таблице 2.1 и таблице 2.2, соответственно.

Таблица 2.1 – Описание ролей

|  |  |
| --- | --- |
| **Название роли** | **Описание** |
| «User» | Пользователь |
| «FileSysem» | Файловая система |
| «MailServer» | Почтовый сервер |

Таблица 2.2 – Варианты использования

|  |  |
| --- | --- |
| **Номер** | ВИ – 01 |
| **Название** | Запуск |
| **Участники** | Пользователь |
| **Описание** | Пользователь запускает программу |
| **Предварительные условия** | Нет |
| **Выходные условия** | Программа запущена |
| **Порядок действий** | Пользователь запускает программу |
| **Номер** | ВИ – 02 |
| **Название** | Основные настройки |
| **Участники** | Пользователь |
| **Описание** | Пользователь указывает основные настройки работы службы |
| **Предварительные условия** | Программа запущена |
| **Выходные условия** | Выбраны корневой каталог, частота обращения к серверу и критерии формирования подкаталога относительно отправителя |
| **Порядок действий** | Пользователь запускает программу, переходит по вкладке «Основное» и указывает там необходимые данные |
| **Номер** | ВИ – 03 |
| **Название** | Распаковка |
| **Участники** | Пользователь |
| **Описание** | Пользователь указывает настройки работы службы с архивами |
| **Предварительные условия** | Программа запущена |
| **Выходные условия** | Выбран режим распаковки архива (автоматический/ручной), выбрана опция создания дополнительного подкаталога для содержимого архива |
| **Порядок действий** | Пользователь запускает программу, переходит по вкладке «Работа с архивами» и указывает там необходимые данные |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| **Номер** | ВИ – 04 |
| **Название** | Лог |
| **Участники** | Пользователь, файловая система |
| **Описание** | Пользователь указывает настройки ведения лог-файлов |
| **Предварительные условия** | Программа запущена |
| **Выходные условия** | Выбрано имя создаваемых лог-файлов, максимальный размер одного файла, максимальная длительность хранения лог-файлов (до очистки) |
| **Порядок действий** | Пользователь запускает программу, переходит по вкладке «Работа с лог-файлами» и указывает там необходимые данные |
| **Номер** | ВИ – 05 |
| **Название** | Настройка сервера |
| **Участники** | Пользователь, почтовый сервер |
| **Описание** | Пользователь указывает настройки подключения к почтовому серверу |
| **Предварительные условия** | Программа запущена |
| **Выходные условия** | Выбраны имя сервера, порт и указан сертификат безопасности |
| **Порядок действий** | Пользователь запускает программу, переходит по вкладке «Аутентификация» и указывает там необходимые данные |
| **Номер** | ВИ – 06 |
| **Название** | Аутентификация |
| **Участники** | Пользователь, почтовый сервер |
| **Описание** | Пользователь указывает настройки авторизации почтового ящика |
| **Предварительные условия** | Программа запущена |
| **Выходные условия** | Указаны логин и пароль для авторизации |
| **Порядок действий** | Пользователь запускает программу, переходит по вкладке «Аутентификация» и указывает там необходимые данные |
| **Номер** | ВИ – 07 |
| **Название** | Конфигурационный файл |
| **Участники** | Пользователь, файловая система |
| **Описание** | Все настройки сохраняются в конфигурационном файле в файловой системе компьютера |
| **Предварительные условия** | Программа запущена, указаны все настройки (необязательно) |
| **Выходные условия** | Все настройки сохранены в конфигурационном файле |
| **Порядок действий** | Пользователь запускает программу, выбирает указывает все нужные данные и нажимает кнопку «Принять» |
| **Номер** | ВИ – 08 |
| **Название** | Подключение |
| **Участники** | Почтовый сервер |
| **Описание** | Происходит подключение к почтовому серверу по указанным в конф. файле настройкам (сервер, порт, безопасность) |

Продолжение таблицы 2.2

|  |  |
| --- | --- |
| **Предварительные условия** | Программа запущена, была нажата кнопка «Принять» |
| **Выходные условия** | Пользователь авторизован (или не авторизован) |
| **Порядок действий** | Пользователь запускает программу, выбирает указывает все нужные данные и нажимает кнопку «Принять» |
| **Номер** | ВИ – 09 |
| **Название** | Распределение писем |
| **Участники** | Файловая система |
| **Описание** | Письма распределяются по каталогам в зависимости от отправителя по критериям указанным в конф. файле |
| **Предварительные условия** | Программа запущена, нажата кнопка «Принять», пользователь успешно авторизован, имеются новые письма |
| **Выходные условия** | В соответственных подкаталогах, созданы файлы, которые представляют из себя прикреплённые файлы к письму, либо, если таковых не имеется, то текст сообщения в текстовом файле с названием –даты полученного письма |
| **Порядок действий** | Пользователь запускает программу, выбирает указывает все нужные данные и нажимает кнопку «Принять» |
| **Номер** | ВИ – 10 |
| **Название** | Остановка |
| **Участники** | Пользователь |
| **Описание** | Пользователь завершают работу программы |
| **Предварительные условия** | Программа запущена |
| **Выходные условия** | Работа программы завершена |
| **Порядок действий** | Пользователь запускает программу, нажимает кнопку «Завершить» |

## 2.3 Диаграмма последовательности

Диаграмма последовательности – это диаграмма, которая служит для представления взаимодействия элементов модели в форме последовательности сообщений и соответствующих событий на линиях жизни объектов в динамике. Только время на данной диаграмме имеет шкалу порядка, а не шкалу отношений и это важно знать [6].

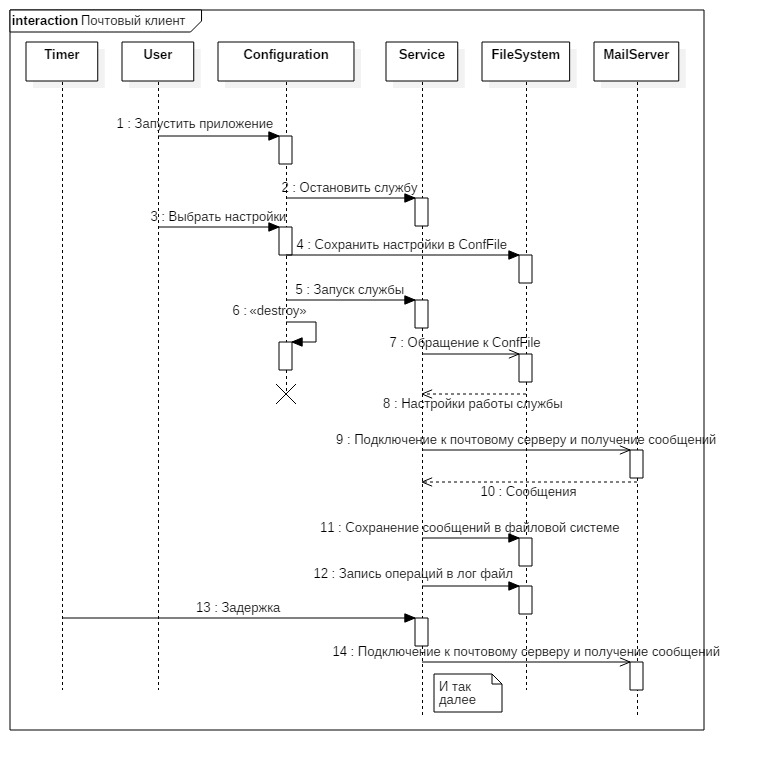
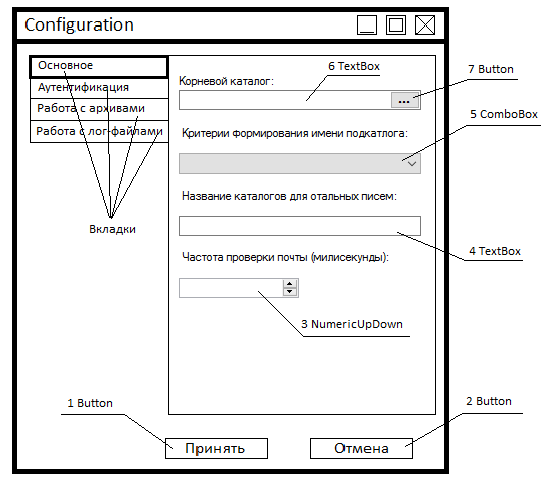


Рисунок 2.3 – Диаграмма последовательности процесса работы всех компонентов разрабатываемого ПО

Данная диаграмма показывает порядок взаимодействия элементов модели и соответствующих событий в конкретно в данном проекте. Главной линией жизни является «Timer».

## 2.4 Проектирование графического интерфейса

На основании поставленных задач было решено выбрать графический интерфейс, который изображен на рисунках 2.4 – 2.7.

Рисунок 2.4 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Основное»)

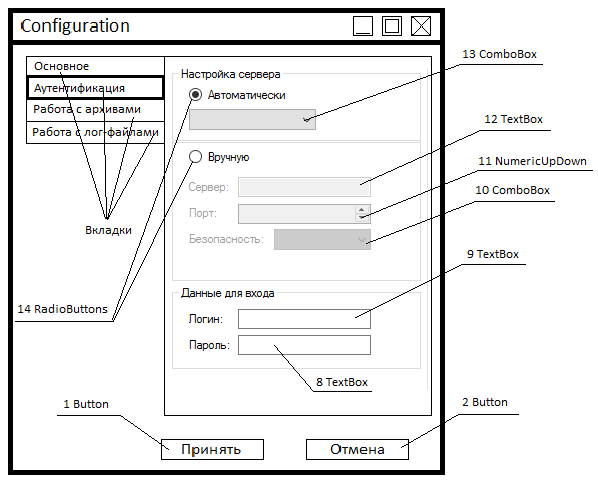


Рисунок 2.5 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Аутентификация»)

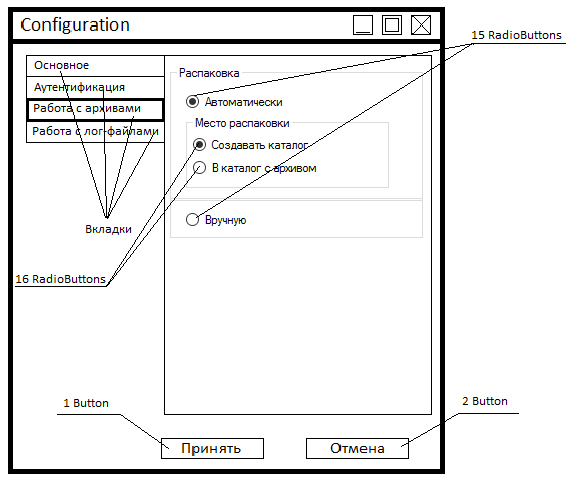


Рисунок 2.6 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Работа с архивами»)

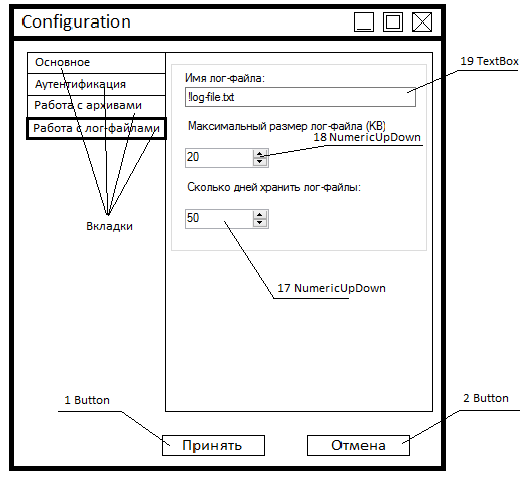


Рисунок 2.7 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Работа с лог-файлами»)

В таблице 2.3 приведено описание элементов управления.

Таблица 2.3– Описание элементов управления формы

|  |  |
| --- | --- |
| **Элемент управления** | **Описание** |
| **1 Button** | По нажатию на данную кнопку приложение «Configuration» сохраняет все выбранные настройки в конфигурационный файл и закрывает себя. |
| **2 Button** | Закрывает приложение «Configuration» ничего не сохраняя. |
| **3 NumericUpDown** | Поле для указания частоты работы службы. |
| **4 TextBox** | Поле для указания названия каталога для писем на подходящих под критерии. |
| **5 ComboBox** | Поле для выбора критериев формирования имени подкаталогов. |
| **6 TextBox** | Поле для указания корневого каталога. |

Продолжение таблицы 2.3

|  |  |
| --- | --- |
| **7 Button** | Кнопка открывает folderBrowserDialog и позволяет выбрать нужную папку. |
| **8 TextBox** | Поле для указания пароля от почтового ящика. |
| **9 TextBox** | Поле для указания логина от почтового ящика. |
| **10 ComboBox** | Поле для выбора сертификата безопасности. |
| **11 NumericUpDown** | Поле для указания порта используемого сервером. |
| **12 TextBox** | Поле для указания сервера. |
| **13 ComboBox** | Поле для выбора наиболее популярных серверов с их портами и сертификатами безопасности. Например: Yandex, Google и т.д. |
| **14 RadioButtons** | Переключатели для выбора способа ввода информации о сервере (ручной, автоматический). |
| **15 RadioButtons** | Переключатели для выбора способа распаковки архивов (ручной автоматический). |
| **16 RadioButtons** | Переключатели для выбора места извлечения файлов из архива (создавать дополнительный подкаталог или нет). |
| **17 NumericUpDown** | Поле для выбора времени хранения лог-файлов. |
| **18 NumericUpDown** | Поле для выбора максимального размера лог-файлов. |
| **19 TextBox** | Поле для указания имени для лог-файлов |

В дальнейшем программа может улучшаться, и реализации новых функций, возможно, понадобятся новые элементы управления, но на данный момент этого достаточно, для выполнения базового функционала.

# 3 РАЗРАБОТКА

## 3.1 Реализация выбранной архитектуры в виде диаграммы пакетов

На основании выбранной архитектуры можно разработать диаграмму пакетов, которая представлена ниже на рисунке 3.1.

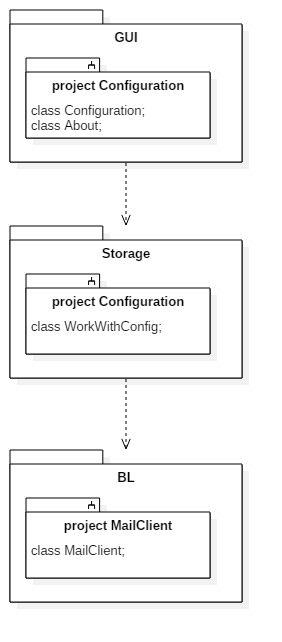


Рисунок 3.1 – Диаграмма пакетов проекта

GUI, в роли которого выступает проект Configuration, обращается к Storage и с помощью методов класса WorkWithConfig сохраняет настройки для работы службы. Как следует из рисунка, GUI напрямую не взаимодействует с BL, это происходит через прослойку Storage, таким образом, пользователь может изменять настройки работы службы в удобном интерфейсе, и для этого ему не нужно знать язык XML.

## 3.2 Разработка диаграммы классов

На рисунке 3.2 приведена общая диаграмма классов, без проекции на архитектуру проекта, в следующем разделе будет приведена она же, разбита на пакеты, которая даст полное представление о структуре проекта.

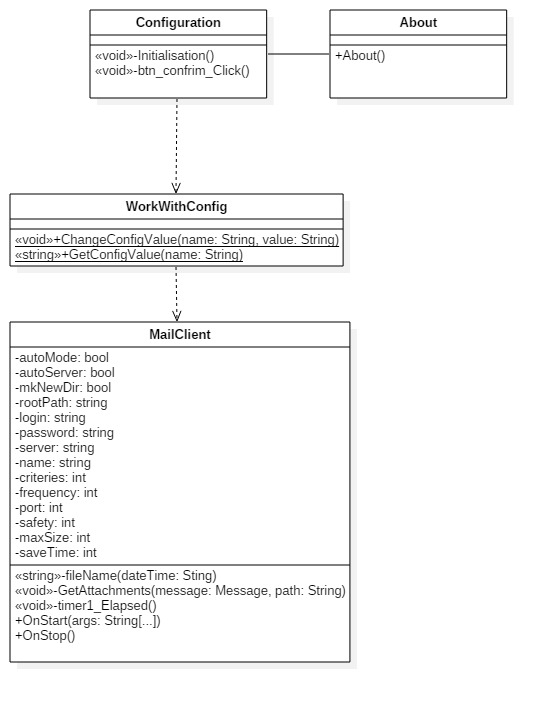


Рисунок 3.2 – Диаграмма классов проекта

Исходя из задания проекта, сложно выделить определённую сущность от которой можно было бы создать множество объектов, поэтому сущности были выделены из функций, за которые они отвечают.

В итоге были выделены классы, описание которых представлено в таблице 3.1.

Таблица 3.1 – Описание классов данного проекта

|  |  |
| --- | --- |
| **Название класса** | Описание класса |
| Configuration | Функцией данного класса является формирование GUI и связь с пакетом Storage. Класс имеет множество обработчиков событий для формирования GUI. |
| WorkWithConfig | Данный класс реализует функции работы с хранилищем, в роли которого в данном проекте выступает файловая система, а точнее конфигурационный файл расположенный в ней. Класс имеет методы*,* с помощью которых можно открыть конфигурационный файл; проверить его корректность; создать конф. файл с набором значений по умолчанию, если конфигурационный файл не был найден; прочитать и установить определённую настройку из /в конфигурационный файл соответственно. |
| MailClient | Данный класс реализует всю бизнес логику проекта. Он реализует службу, которая общается с сервером, получает сообщения, достаёт из них прикреплённые файлы и т.д. |

## 3.3 Общая диаграмма классов с отображением на архитектуру

На приведенной ниже гибридной диаграмме пакетов и классов показаны основные связи. Связи представлены на различных уровнях, как между пакетами, так и между классами. Связи между пакетами показаны с помощью методов, которые ссылаются на классы других пакетов, тем самым осуществляя взаимосвязь между отдельными частями архитектуры.

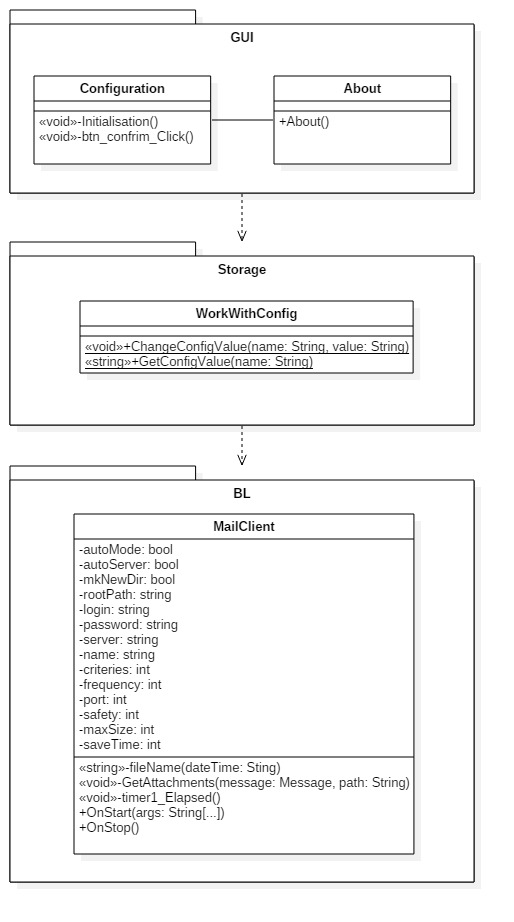


Рисунок 3.3 – Диаграмма классов проекта с проекцией на архитектуру

Благодаря рисунку 3.3 можно понять общую структуру разрабатываемого проекта. Как и описывалось ранее, пользователь с помощью графического интерфейса устанавливает все настройки для службы в конфигурационный файл. Затем служба обращается к этому файлу и в соответствии заданными настройками выполняет свои функции.

## 3.4 Разработка алгоритма работы программы

Для реализации поставленной задачи был разработан алгоритм работы программы, схема которого изображена на рисунке 3.4.



B

B А

Рисунок 3.4 – Обобщенный алгоритм работы службы (начало)



Рисунок 3.5 – Обобщенный алгоритм работы службы (конец)

B

А



Рисунок 3.6 – Алгоритм работы метода GetAttachments ()

## 3.5 Разработка методов классов

Для реализации задуманной архитектуры проекта были использованы методы и функции, которые описаны в таблице 3.2.

Таблица 3.2 – Описание методов класса MailClient

|  |  |
| --- | --- |
| Метод | Описание |
| OnStart(string[] args) | Метод, который вызывается при событии запуска службы. В данном методе осуществляется подключение к почтовому серверу, авторизация пользователя и настройка таймера службы.  − уровень доступа – protected;  − возвращаемое значение – void;  – args – параметры командной стшроки. |
| OnStop() | Метод, который вызывается при событии остановки службы. В данном методе осуществляется освобождение ресурсов.  − уровень доступа – protected;  − возвращаемое значение – void; |
| loopService(object sender, System.Timers.ElapsedEventArgs e) | Метод, который вызывается по событию тика таймера. Данный метод является основным рабочим циклом программы. В данном методе осуществляется выбор новых писем, сохранения их в нужные каталоги, выбор критерия размещения письма.  − уровень доступа – private;  − возвращаемое значение – void;  – sender – объект, который вызвал данный метод. |
| GetAttachments(ImapX.Message message, string path) | Метод предназначен для извлечения вложений из письма (или сохранения текста письма), реализует извлечение файлов архива в подкаталог или в корневой каталог письма, в зависимости от конфигураций.  − уровень доступа – private;  − возвращаемое значение – void;  – message – объект письма;  – path – путь к подкаталогу письма. |
| fileName(string dateTime) | Метод предназначен для формирования имени файла куда будет сохраняться текст письма, в котором нет вложений  − уровень доступа – private;  − возвращаемое значение – string: строка сформированная из MessageData;  – dateTime – текущая дата-время в формате MessageData. |

Продолжение таблицы 3.2

|  |  |
| --- | --- |
| GetConfInfo() | Метод предназначен для извлечения настроек из конфигурационного файла.  − уровень доступа – private;  − возвращаемое значение – void; |

Таблица 3.3 – Описание методов класса WorkWithConfig

|  |  |
| --- | --- |
| Метод | Описание |
| SetConfigValue(string name, string value) | Метод, для записи данных в конфигурационный файл.  − уровень доступа – public;  − возвращаемое значение – bool: результат записи;  – name – ключ;  – value – значение. |
| GetConfigValue(string name, string filename) | Метод, для извлечения данных из конфигурационного файла.  − уровень доступа – public;  − возвращаемое значение – string: значение запрашиваемого поля;  – name – ключ;  – filename – путь к файлу конфигураций. |

Таблица 3.4 – Описание методов класса Configuration

|  |  |
| --- | --- |
| Метод | Описание |
| Initialisation(object sender, EventArgs e) | Метод, для извлечения данных из конфигурационного файла.  − уровень доступа – private;  − возвращаемое значение – void; |
| SaveToolStripMenuItem\_Click(object sender, EventArgs e) | Метод, для добавления выбранных настроек в конфигурационный файл.  − уровень доступа – private;  − возвращаемое значение – void; |
| RestartService() | Метод, перезапуска службы после внесения изменений в конфигурационный файл.  − уровень доступа – private;  − возвращаемое значение – void; |

Таблица 3.5 – Описание переменных класса MailClient

|  |  |
| --- | --- |
| Переменная | Описание |
| private StreamWriter file | Поток для записи в файл (для ведения лога работы службы) |
| private StreamWriter error | Поток для записи в файл (для ведения лога ошибок) |
| private System.Timers.Timer timer1 | Таймер для управлением службой |
| private ImapX.ImapClient client | Объект почтового сервера |
| private const string pathToLog | Путь к лог-файлу |
| Process process | Переменная для получения реального расположения установленной службы |
| FileInfo finfo | Переменная для получения реального расположения установленной службы |
| private bool isLogined | Переменная для проверки аутентификации пользователя |
| private string criteries | Конфигурация: критерии для размещения писем (шаблон >> каталог) |
| private string rootPath | Конфигурация: путь к коревому каталогу |
| private string otherDir | Конфигурация: название подпапки, куда помещать письма, которые не подошли ни к одному критерию |
| private string login | Конфигурация: логин пользователя от почтового ящика |
| private string password | Конфигурация: пароль пользователя от почтового ящика |
| private string server | Конфигурация: почтовый сервер |
| private string name | Конфигурация: название лог файла |
| private int frequency | Конфигурация: частота обращения к почтовому серверу |
| private int port | Конфигурация: порт почтового сервера |
| private int safety | Конфигурация: протокол безопасности почтового сервера |
| private int maxSize | Конфигурация: максимальный размер лог-файла |
| private int saveTime | Конфигурация: время хранения лог-файла на диске |
| private bool autoMode | Конфигурация: настройка, отвечающая за автоматическое извлечение файлов из архива |
| private bool mkNewDir | Конфигурация: настройка, отвечающая за создание дополнительного подкаталога для извлечения содержимого архива |

# 4 ВЕРИФИКАЦИЯ И ТЕСТИРОВАНИЕ

## 4.1 Верификация

В техническом задании (ПРИЛОЖЕНИЕ А), были определены следующие задачи проекта:

1. периодически, в фоновом режиме, получение почты согласно настройкам, указанным в конфигурационном файле (сервер, порт, использование SSL/TLS, логин, пароль от почтового ящика пользователя);
2. распределение писем по каталогам в зависимости от настроек пользователя;
3. распаковка приложенных архивов в каталог в зависимости от отправителя;
4. настройка конфигурационного файла через графический интерфейс, где пользователь должен будет указать:
   1. корневой каталог, где будут храниться все остальные подкаталоги;
   2. имя подкаталога и критерии, по которым необходимо помещать письма в данный подкаталог;
   3. имя подкаталога, в который будут помещаться письма, не соответствующие ни одному критерию;
   4. сервер, порт, использование SSL/TLS, логин, пароль от почтового ящика пользователя;
   5. разархивировать ли автоматически прикреплённые архивы к сообщению;
   6. создавать ли подкаталог для содержимого архива;
   7. периодичность обращения к почтовому серверу в миллисекундах.
5. ведение лога работы в текстовом файле
   1. настройка имени лог-файла;
   2. настройка максимального размера лог-файла (при достижении максимума необходимо создать новый файл);
   3. настройка автоматической очистки лог-файлов.

В результате разработки программы все вышеперечисленные задачи были решены и написаны функции для их реализации.

В таблице 4.1 приведены методы, которые подтверждают выполнение вышеперечисленных функций.

Таблица 4.1 – Верификация функций

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Требование в ТЗ  (Задача или функция) | Номер функции | Проектирование  (Прецедент) | Подтверждение выполнения. Разработка  (Класс.Метод()) | Вывод |
| 1. Периодически, в фоновом режиме, получение почты согласно конфигурациям | 1 |  |  | Pass |
| a) Получение конфигураций | 1a) | Конфигурационный файл | WorkWithConfig. GetConfigValue () | Pass |
| b) Подключение | 1b) | Подключение | MailClient.connect() | Pass |
| c) Периодическое получение почты | 1c) | Получение почты | MailClient.loopService() | Pass |
| 2. Распределение писем по каталогам в зависимости от настроек пользователя; | 2 |  |  |  |
| а) Распределение писем по каталогам | 2a) | Распределение | MailClient.MoveToFolder() | Pass |
| 3. Распаковка приложенных архивов в каталог в зависимости от отправителя | 3 |  |  | Pass |
| a) Распаковка приложенных архивов в каталог | 3a) | Распаковка | MailClient.GetAttachments() | Pass |
| b) Формирование файла из текста письма если нет вложений | 3b) | Распаковка | MailClient.FileName() | Pass |
| 4. Настройка конфигурационного файла через графический интерфейс | 4 |  |  | Pass |
| a) Выбор корневого каталога | 4a) | Конфигурационный файл | WorkWithConfig. SetConfigValue() | Pass |
| b) Выбор критериев размещения писем | 4b) | Конфигурационный файл | WorkWithConfig. SetConfigValue() | Pass |

Продолжение таблицы 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| c) Выбор каталога для писем которые не подходят критериям | 4c) | Конфигурационный файл | WorkWithConfig. SetConfigValue() | Pass |
| d) Установка сервера, порта, протокола безопасности, | 4d) | Настройка сервера | WorkWithConfig. SetConfigValue() | Pass |
| e) Настройки архиватора | 4e) | Распаковка | WorkWithConfig. SetConfigValue() | Pass |
| f) Установка логина и пароля | 4h) | Аутентификация | WorkWithConfig. SetConfigValue() | Pass |
| g) Выбор периодичности обращения к почтовому серверу | 4g) | Конфигурационный файл | WorkWithConfig. SetConfigValue() | Pass |
| 5. Ведение лога работы в текстовом файле | 5 |  |  | Pass |
| a) Настройки лог-файлов | 5a) | Лог | WorkWithConfig. SetConfigValue() | Pass |
| b) Добавить запись в лог | 5b) | Лог | MailClient.AddLog() | Pass |
| c) Ротация логов | 5c) | Лог | MailClient.LogRotate() | Pass |

Помимо основных функций было решено запрашивать права администратора для перезапуска службы после внесения изменений в конфигурационный файл. Данное решение было предпринято для удобства конфигурирования службы. Также в секцию «Аутентификация» был добавлен пункт «Выбрать сервер», в котором пользователь может выбрать существующий сервер, а не вводит информацию (сервер, порт и т.д.) вручную.

Была реализована возможность вложенности подкаталогов в критериях распределения писем. Это позволит пользователю более точно позиционировать письма и повысит точность системы (подробнее в ПРИЛОЖЕНИИ В. Руководство оператора).

В результате верификации проекта было установлено, что все задачи решаются теми или иными методами, подтверждение приведено в разделе 4.2 тестирование.

## 4.2 Тестирование

Для проверки работоспособности программы для тестирования были выделены следующие функции:

1) Запись данных в конфигурационный файл

2) Перезапуск службы с конфигуратора

3) Подключение к почтовому серверу

4) Аутентификация

5) Помещение писем или их вложений в соответствующую подпапку

6) Извлечение вложений и их распаковка

7) Ведение лог-файлов

Таблица 4.2 – Тестирование программы и её результаты

|  |  |  |  |
| --- | --- | --- | --- |
| Тестируемая функция | Номер теста | Цель теста | Результат теста |
| 1) Запись данных в конфигурационный файл | 1 | Проверка наличия конфигурационного файла | Pass |
| 2 | Проверка синтаксиса введённых критериев размещения писем | Pass |
| 3 | Проверка на выбор несуществующей директории | Pass |
| 2) Перезапуск службы с конфигуратора | 4 | Проверка реакции конфигуратора на отсутствие службы | Pass |
| 3) Подключение к почтовому серверу | 5 | Проверка подключения к несуществующему серверу | Pass |
| 6 | Проверка на подключение к существующему серверу | Pass |
| 4) Аутентификация | 7 | Проверка авторизации по не правильный логину и паролю | Pass |
| 8 | Проверка авторизации по правильный логину и паролю | Pass |
| 5) Помещение писем или их вложений в соответствующую подпапку | 9 | Проверка обработки критериев распределения писем по не валидному критерию | Pass |
| 10 | Проверка обработки критериев распределения писем по валидному критерию | Pass |
| 11 | Проверка на отсутствие корневого каталога | Pass |
| 12 | Проверка размещения письма которое подходит по двум критериям | Pass |

Продолжение таблицы 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| 6) Извлечение вложений и их распаковка | 13 | Проверка попытки распаковки не архива | Pass |
| 14 | Проверка распаковки архива в корень подпапки письма | Pass |
| 15 | Проверка распаковки архива в создаваемую для содержимого архива подпапку | Pass |
| 16 | Проверка отсутствия вложений | Pass |
| 7) Ведение лог-файлов | 17 | Проверка записи в лог информации о текущей операции | Pass |
| 18 | Проверка создания лог файла по заданному в конфигурации имени | Pass |
| 19 | Проверка на переполнение заданного размера лог-файла | Pass |
| 20 | Проверка на истечение даты хранения | Pass |
| 8) Частота выполнения основного цикла программы | 21 | Проверка соответствия частоты работы службы | Fail |

В результате тестирования программы была обнаружена ошибка, рабочее тело службы выполняется с заданной частотой (по таймеру) и по истечении заданного промежутка времени начинает своё выполнение с начала вне зависимости от того была ли завершена текущая операция. Эта ошибка будет исправлена в последующих версиях данного ПО.

Как следует из таблиц 4.1 и 4.2 верификация и тестирование программы выполнены успешно.

# ВЫВОДЫ

В результате выполнения данной работы был реализован проект, предоставляющий возможность получения писем и распределение их по каталогам в зависимости от файла настроек с возможностью разархивирования прикреплённых архивов.

Результат выполнения данной работы:

1) проведен анализ предметной области и существующих решений, в результате чего было сформировано постановку задачи;

2) на основе постановки задачи сформировано техническое задание;

3) проведено проектирование программы:

– выбрана архитектура проекта;

– построено диаграмма вариантов использования;

– построено диаграмма последовательности;

­– разработан алгоритм работы программы.

4) в ходе разработки программы были выполнены такие шаги:

– реализована архитектура проекта;

– разработана диаграмма пакетов;

– разработана диаграмма классов;

­– был оптимизирован изначальный алгоритм работы программы.

5) проведена успешная верификация программы;

6) проведено успешное тестирование программы;

7) написано руководство оператора;

8) сделана презентация.

Разработанная программа верифицирована и протестирована.

Существует несколько путей развития данного ПО, вот некоторые из них:

1) решить проблему с периодом обращения службы к почтовому серверу;

2) реализовать поддержку большего числа видов архивов;

3) повысить скорость работы данного ПО;

4) реализовать более удобный синтаксис создания критериев распределения писем;

5) реализовать вторую часть системы – автоматическое отправление писем, в результате возникновения каких-либо событий.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Почтовый сервер для начинающих. Структура и принцип работы. [Электронный ресурс]. – Режим доступа: <https://interface31.ru/tech_it/2010/09/pochtovyj-server-struktura-i-princip-raboty.html>. (Дата обращения: 06.02.2017).

2 Обзор 5 самых популярных почтовых клиентов [Электронный ресурс] . – Режим доступа: <http://softcatalog.info/ru/obzor/pochtovye-klienty-dlya-windows>. (Дата обращения: 06.02.2017).

3 Совершенный код. Мастер-класс / Пер. с англ. — М. : Издательство «Русская редакция», 2010. — 41с.

4 Грейди Буч. Язык UML. Руководство пользователя / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. — СПб.: Питер, 2004. — 432 с.

5 Язык UML. Диаграммы вариантов использования [Электронный ресурс]. – Режим доступа: [http://www.its.kpi.ua/itm/tkot/Students/Lec2\_1\_UML[p].pdf](http://www.its.kpi.ua/itm/tkot/Students/Lec2_1_UML%5bp%5d.pdf). (Дата обращения: 10.03.2017).

6 Язык UML в анализе и проектировании систем и бизнес-процессов [Электронный ресурс]. – Режим доступа: http://www.slideshare.net/DEVTYPE/3-48780912.(Дата обращения: 10.03.2017).

7 ImapX documentation [Электронный ресурс]. – Режим доступа: http://imapx.codeplex.com/documentation.(Дата обращения: 24.11.2016).

# ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1 Введение

1.1 Наименование программы

Наименование ­– "Почтовый клиент (получение почты)".

1.2 Краткая характеристика области применения

Разрабатываемая служба может быть использована для добавления возможности получения входной информации по электронной почте для систем, которые по умолчанию получают эту информацию из файлов, находящихся в каталогах файловой системы (старые системы).

2 Основания для разработки

2.1 Основание для проведения разработки

Основание для разработки – задание на курсовой проект по дисциплине "Системное программное обеспечение", в целях закрепления, углубления и обобщения знаний, полученных за время обучения, и их применения к комплексному решению конкретной задачи.

2.2 Наименование и условное обозначение темы разработки

Наименование темы разработки – "Почтовый клиент (получение почты)".

Условное обозначение темы разработки (шифр темы) – "ПК-01".

3 Назначение разработки

3.1 Функциональное назначение

Данный программный продукт не предполагает использование в какой-то конкретной отрасли человеческой деятельности.

Функциональным назначением программы является получение писем и распределение их по каталогам в зависимости от файла настроек с возможностью разархивирования прикреплённых архивов.

3.2 Эксплуатационное назначение

Данная программа является службой и работает в автоматическом режиме.

Условия эксплуатации программы – офис и домашние условия.

Обслуживание данного ПО не предполагается.

Для установки данной программы необходимо реализовать специальный инсталлятор.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам

4.1.1 Требования к составу выполняемых функций

Программа должна обеспечивать возможность решения задач:

1. периодически, в фоновом режиме, получение почты согласно настройкам, указанным в конфигурационном файле (сервер, порт, использование SSL/TLS, логин, пароль от почтового ящика пользователя);
2. распределение писем по каталогам в зависимости от настроек пользователя;
3. распаковка приложенных архивов в каталог в зависимости от отправителя;
4. настройка конфигурационного файла через графический интерфейс, где пользователь должен будет указать:
   1. корневой каталог, где будут храниться все остальные подкаталоги;
   2. имя подкаталога и критерии, по которым необходимо помещать письма в данный подкаталог;
   3. имя подкаталога, в который будут помещаться письма, не соответствующие ни одному критерию;
   4. сервер, порт, использование SSL/TLS, логин, пароль от почтового ящика пользователя;
   5. разархивировать ли автоматически прикреплённые архивы к сообщению;
   6. создавать ли подкаталог для содержимого архива;
   7. периодичность обращения к почтовому серверу в миллисекундах.
5. ведение лога работы в текстовом файле
   1. настройка имени лог-файла;
   2. настройка максимального размера лог-файла (при достижении максимума необходимо создать новый файл);
   3. настройка автоматической очистки лог-файлов.

4.1.2 Требования к организации входных данных

Входные данные программы должны быть организованы в виде конфигурационного файла в формате XML, который формируется из указанных пользователем настроек в окне программы «Настройка конфигурационного файла». Данный файл должен находиться в выбранном пользователем корневом каталоге в подкаталоге «System» наряду с подкаталогами с письмами.

С другой стороны входными данными являются электронные письма, получаемые с помощью программы, посредством IMAP протокола.

4.1.3 Требования к организации выходных данных

Выходные данные будут представлены в виде файлов, прикреплённых к полученным письмам. Эти файлы могут быть любого типа. Если файл не прикреплен, то необходимо сохранять текст самого письма в текстовый файл. Выходные файлы должны храниться на локальных или съемных носителях в выбранном пользователем каталоге в подкаталоге относительно отправителя. А также лог-файлы в формате txt.

4.1.4 Требования к временным характеристикам

Требования к временным характеристикам программы не предъявляются.

4.2 Требования к составу и параметрам технических средств

Для исполнения программы необходим универсальный компьютер любого типа, для которого получен двоичный код, представленный в виде исполнимого файла (.exe-файла), способного выполняться установленным в ПК процессором.

Для запуска программы на компьютерах соответствующего типа должен быть получен исполнимый файл, способный выполняться на компьютерах именно этого типа. Для этого необходимо откомпилировать исходные файлы на компьютерах того типа, на которых предполагается исполнять результирующую программу.

В состав технических средств должен входить х64 совместимый компьютер, включающий в себя:

1) процессор с тактовой частотой 2.7Ghz;

2) оперативную память объемом 4GB, не менее;

3) свободного дискового пространства 100MB;

4) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

5) клавиатуру и мышь;

6) жесткий диск;

7) сетевой адаптер;

8) если предполагается работа со сменными дисками, то необходим USB-адаптер.

4.3 Требования к информационной и программной  
совместимости

4.3.1 Требования к информационным структурам и методам решения

Входные данные должны быть представлены в виде сообщений, посылаемых по электронной почте

4.3.2 Требования к исходным кодам и языкам  
программирования

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда VisualStudio .NET.

4.3.3 Требования к программным средствам, используемым программой

На ПК должны быть установлены:

1) Операционная системаWindows 10x64;

2) .NetFramework4.5;

3) Windows Forms 2.0.

4.3.4 Требования к защите информации и программ

Требования к защите информации и программ не предъявляются.

5 Требования к программной документации

5.1 Предварительный состав программной документации

В результате разработки программы должна быть представлена следующая программная документация:

а) техническое задание;

б) схемы алгоритмов;

в) тексты программ;

г) план тестирования и верификации;

д) руководство оператора;

е) пояснительная записка.

6 Стадии и этапы разработки

6.1 Стадии разработки

Разработка должна быть проведена в две стадии:

1) разработка технического задания;

2) рабочее проектирование.

6.2 Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии рабочего проектирования должны быть выполнены следующие этапы работ:

1) разработка программы;

2) разработка программной документации;

3) испытания программы.

6.3 Содержание работ по этапам

На этапе разработки технического задания должны быть выполнены такие работы:

1) постановка задачи;

2) определение и уточнение требований к техническим средствам;

3) определение требований к программе;

4) определение стадий, этапов и сроков разработки программы и документации на неё;

5) выбор языков программирования;

6) согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка требуемых программных документов.

На этапе испытаний программы должны быть выполнены следующие виды работ:

1) разработка, согласование и утверждение программы и методики испытаний;

2) проведение приемосдаточных испытаний;

3) корректировка программы и программной документации по результатам испытаний.

# ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММ

Класс Configuration

public partial class Configuration : Form

{

public Configuration()

{

InitializeComponent();

}

/// <summary>

/// Обработчик события загрузки формы

/// </summary>

private void Initialisation(object sender, EventArgs e)

{

if (WorkWithConfig.GetConfigValue("main\_frequency") == "Error")

{

Close();

}

for (int i = 0; i < WorkWithConfig.GetConfigValue("main\_criteries").Split('+').Length; i++)

{

criteries.Text += WorkWithConfig.GetConfigValue("main\_criteries").Split('+')[i];

if (i < WorkWithConfig.GetConfigValue("main\_criteries").Split('+').Length-1)

criteries.Text += "\r\n";

}

mailServers.DropDownStyle = ComboBoxStyle.DropDownList;

safety.DropDownStyle = ComboBoxStyle.DropDownList;

rootPath.Text = WorkWithConfig.GetConfigValue("main\_rootPath");

otherDir.Text = WorkWithConfig.GetConfigValue("main\_otherDir");

frequency.Value = Int32.Parse(WorkWithConfig.GetConfigValue("main\_frequency"));

//

autoServer.Checked = Boolean.Parse(WorkWithConfig.GetConfigValue("server\_autoServer"));

noAutoServer.Checked = !autoServer.Checked;

mailServers.SelectedIndex = Int32.Parse(WorkWithConfig.GetConfigValue("server\_mailServer"));

server.Text = WorkWithConfig.GetConfigValue("server\_server");

port.Value = Int32.Parse(WorkWithConfig.GetConfigValue("server\_port"));

safety.SelectedIndex = Int32.Parse(WorkWithConfig.GetConfigValue("server\_safety"));

login.Text = WorkWithConfig.GetConfigValue("server\_login");

password.Text = WorkWithConfig.GetConfigValue("server\_password");

//

autoUnzip.Checked = Boolean.Parse(WorkWithConfig.GetConfigValue("archive\_autoMode"));

noAutoUnzip.Checked = !autoUnzip.Checked;

mkNewDir.Checked = Boolean.Parse(WorkWithConfig.GetConfigValue("archive\_mkNewDir"));

noMkNewDir.Checked = !mkNewDir.Checked;

//

logName.Text = WorkWithConfig.GetConfigValue("log\_name");

maxSize.Value = Int32.Parse(WorkWithConfig.GetConfigValue("log\_maxSize"));

saveTime.Value = Int32.Parse(WorkWithConfig.GetConfigValue("log\_saveTime"));

}

/// <summary>

/// Обработчик события выбора вкладки

/// </summary>

private void tabControl\_Style(object sender, DrawItemEventArgs e)

{

Graphics g;

string sText;

int iX;

float iY;

SizeF sizeText;

g = e.Graphics;

Pen pencil = new Pen(Color.Blue);

sText = tabControl1.TabPages[e.Index].Text;

sizeText = g.MeasureString(sText, tabControl1.Font);

iX = e.Bounds.Left + 6;

iY = e.Bounds.Top + (e.Bounds.Height - sizeText.Height) / 2;

g.DrawString(sText, tabControl1.Font, Brushes.Black, iX, iY);

}

/// <summary>

/// Обработчик события нажатия на кнопку для добавления выбранных настроек в конфигурационный файл

/// </summary>

private void SaveToolStripMenuItem\_Click(object sender, EventArgs e)

{

if (!WorkWithConfig.SetConfigValue("main\_rootPath", ""))

{

Close();

}

if (string.IsNullOrWhiteSpace(rootPath.Text) || string.IsNullOrWhiteSpace(otherDir.Text) || string.IsNullOrWhiteSpace(server.Text)

|| string.IsNullOrWhiteSpace(login.Text )|| string.IsNullOrWhiteSpace(password.Text) || string.IsNullOrWhiteSpace(logName.Text))

{

MessageBox.Show("Заполните все поля!", "Ошибка");

return;

}

//Основные настройки

WorkWithConfig.SetConfigValue("main\_rootPath", rootPath.Text);

string pattern = "[+]+";

WorkWithConfig.SetConfigValue("main\_criteries", Regex.Replace(criteries.Text.Replace("\r\n", "+"),pattern,"+").TrimEnd('+'));

WorkWithConfig.SetConfigValue("main\_otherDir", otherDir.Text);

WorkWithConfig.SetConfigValue("main\_frequency", frequency.Value.ToString());

//Настройки сервера

WorkWithConfig.SetConfigValue("server\_login", login.Text);

WorkWithConfig.SetConfigValue("server\_password", password.Text);

WorkWithConfig.SetConfigValue("server\_autoServer", autoServer.Checked.ToString());

WorkWithConfig.SetConfigValue("server\_mailServer", mailServers.SelectedIndex.ToString());

if (!autoServer.Checked)

{

WorkWithConfig.SetConfigValue("server\_server", server.Text);

WorkWithConfig.SetConfigValue("server\_port", port.Value.ToString());

WorkWithConfig.SetConfigValue("server\_safety", safety.SelectedIndex.ToString());

}

else

{

switch (mailServers.SelectedIndex)

{

case 0:

WorkWithConfig.SetConfigValue("server\_server", "imap.gmail.com");

WorkWithConfig.SetConfigValue("server\_port", "993");

WorkWithConfig.SetConfigValue("server\_safety", "0");

break;

case 1:

WorkWithConfig.SetConfigValue("server\_server", "imap.ukr.net");

WorkWithConfig.SetConfigValue("server\_port", "993");

WorkWithConfig.SetConfigValue("server\_safety", "0");

break;

case 2:

WorkWithConfig.SetConfigValue("server\_server", "imap.mail.yahoo.com");

WorkWithConfig.SetConfigValue("server\_port", "993");

WorkWithConfig.SetConfigValue("server\_safety", "0");

break;

default:

break;

}

}

//Настройки распаковки архивов

WorkWithConfig.SetConfigValue("archive\_autoMode", autoUnzip.Checked.ToString());

WorkWithConfig.SetConfigValue("archive\_mkNewDir", mkNewDir.Checked.ToString());

//Настройка лог-файлов

WorkWithConfig.SetConfigValue("log\_name", logName.Text);

WorkWithConfig.SetConfigValue("log\_maxSize", maxSize.Value.ToString());

WorkWithConfig.SetConfigValue("log\_saveTime", saveTime.Value.ToString());

if (MessageBox.Show("Вы изменили настройки, хотите перезапустить службу?", "Предупреждение", MessageBoxButtons.OKCancel) == DialogResult.OK)

{

Thread myThread = new Thread(RestartService);

myThread.Start();

Close();

}

}

/// <summary>

/// Метод, перезапуска службы после внесения изменений в конфигурационный файл

/// </summary>

private void RestartService()

{

ServiceController service = new ServiceController("MailClient");

try

{

var efergf = service.ServiceHandle;

}

catch (InvalidOperationException e)

{

MessageBox.Show("Указанная служба не установлена","Ошибка");

Close();

}

if (service.Status == ServiceControllerStatus.Running)

{

service.Stop();

service.WaitForStatus(ServiceControllerStatus.Stopped);

service.Start();

service.WaitForStatus(ServiceControllerStatus.Running);

}

else

service.Start();

}

}

Класс WorkWithConfig

public class WorkWithConfig

{

/// <summary>

/// Метод для записи информации в конфигурационный файл

/// </summary>

/// <param name="name">Ключ</param>

/// <param name="value">Значение</param>

/// <returns></returns>

static public bool SetConfigValue(string name, string value)

{

try

{

string filename = @".\config\Settings.config";

XmlDocument xmldoc = new XmlDocument();

xmldoc.Load(filename);

XmlNodeList nodeList = xmldoc.DocumentElement.ChildNodes;

foreach (XmlElement element in nodeList)

if (element.Name.ToLower() == "appSettings".ToLower())

{

XmlNodeList node = element.ChildNodes;

if (node.Count > 0)

foreach (XmlElement el in node)

if (el.Attributes["key"].InnerText == name)

{

el.Attributes["value"].InnerText = value;

break;

}

break;

}

xmldoc.Save(filename);

return true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка");

}

return false;

}

/// <summary>

/// Метод для считывания информации из конфигурационного файла

/// </summary>

/// <param name="name">Ключ</param>

/// <param name="filename">Путь к файлу конфигураций</param>

/// <returns></returns>

static public string GetConfigValue(string name, string filename=".")

{

bool isServiceAccess = false;

try

{

if (filename == ".")

{

filename += @"\config\Settings.config";

}

else

{

isServiceAccess = true;

filename += @"\..\config\Settings.config";

}

XmlDocument xmldoc = new XmlDocument();

xmldoc.Load(filename);

XmlNodeList nodeList = xmldoc.DocumentElement.ChildNodes;

foreach (XmlElement element in nodeList)

if (element.Name.ToLower() == "appSettings".ToLower())

{

XmlNodeList node = element.ChildNodes;

if (node.Count > 0)

foreach (XmlElement el in node)

if (el.Attributes["key"].InnerText == name)

{

return el.Attributes["value"].InnerText;

}

break;

}

}

catch (Exception ex)

{

if (!isServiceAccess)

{

MessageBox.Show(ex.Message, "Ошибка");

}

}

return "Error";

}

}

Класс MailClient

public partial class MailClient : ServiceBase

{

public MailClient()

{

InitializeComponent();

}

private StreamWriter file, errors; // Поток для записи в файл для ведения лога

private FileStream fileStream;

private System.Timers.Timer timer1; // Таймер для управлением службой

private Encoding uniString = Encoding.Unicode; //

private string fromEmail;

private bool isExist, isLogined; //Переменная для проверки аутентификации пользователя

private ImapX.ImapClient client; // Объект почтового сервера

private const string pathToLog = @"\..\log\"; // Путь к лог-файлу

Process process; // Переменная для получения реального расположения установленной службы

FileInfo finfo; // Переменная для получения реального расположения установленной службы

private string criteries, // Конфигурация: критерии для размещения писем (шаблон >> каталог)

rootPath, // Конфигурация: путь к коревому каталогу

otherDir, // Конфигурация: название подпапки, куда помещать письма, которые не подошли ни к одному критерию

login, // Конфигурация: логин пользователя от почтового ящика

password, // Конфигурация: пароль пользователя от почтового ящика

server, // Конфигурация: почтовый сервер

name; // Конфигурация: название лог файла

private int frequency, // Конфигурация: частота обращения к почтовому серверу

port, // Конфигурация: порт почтового сервера

safety, // Конфигурация: протокол безопасности почтового сервера

maxSize, // Конфигурация: максимальный размер лог-файла

saveTime; // Конфигурация: время хранения лог-файла на диске

private bool autoMode, // Конфигурация: настройка, отвечающая за автоматическое извлечение файлов из архива

mkNewDir; // Конфигурация: настройка, отвечающая за создание дополнительного подкаталога для извлечения содержимого архива

/// <summary>

/// Метод, который вызывается при событии запуска службы. В данном методе осуществляется

/// подключение к почтовому серверу, авторизация пользователя и настройка таймера службы.

/// </summary>

/// <param name="args"></param>

protected override void OnStart(string[] args)

{

process = Process.GetCurrentProcess();

finfo = new FileInfo(process.MainModule.FileName);

errors = new StreamWriter(new FileStream(finfo.DirectoryName + pathToLog + "ErrorLog.txt", System.IO.FileMode.Append));

GetConfInfo();

fileStream = LogRotate();

file = new StreamWriter(fileStream);

AddLog("MailClient стартовал");

client = new ImapX.ImapClient(server, port, safety == 0 ? true : false);

client.Behavior.MessageFetchMode = MessageFetchMode.Basic | MessageFetchMode.Attachments;

if (client.Connect())

{

if (login == "" || password == "")

{

AddLog("Введите логин и пароль!");

OnStop();

}

if (!client.Login(login, password))

{

AddLog("Ошибка авторизации");

OnStop();

}

AddLog("Успешная аутентиффикация");

isLogined = true;

client.Folders.Add(otherDir);

Directory.CreateDirectory(rootPath + otherDir);

}

else

{

AddLog("Ошибка соединения");

OnStop();

}

this.timer1 = new System.Timers.Timer(frequency);

this.timer1.AutoReset = true;

this.timer1.Elapsed += new System.Timers.ElapsedEventHandler(this.loopService);

this.timer1.Start();

}

/// <summary>

/// Метод, который вызывается при событии остановки службы. В данном методе осуществляется освобождение ресурсов.

/// </summary>

protected override void OnStop()

{

AddLog("MailClient остановлен");

this.file.Close();

this.errors.Close();

this.timer1.Stop();

}

/// <summary>

/// Метод, который вызывается по событию тика таймера. Данный метод является основным рабочим циклом программы.

/// В данном методе осуществляется выбор новых писем, сохранения их в нужные каталоги, выбор критерия размещения письма.

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void loopService(object sender, System.Timers.ElapsedEventArgs e)

{

if (isLogined)

{

var messages = client.Folders["INBOX"].Search("ALL");

foreach (ImapX.Message message in messages)

{

AddLog("Найдено новое сообщение");

isExist = false;

fromEmail = "";

for (int i = 0; i < criteries.Split('+').Length; i++)

{

if (message.From.Address.ToString().Contains(criteries.Split('+')[i].Split(new string[] { " >> " }, StringSplitOptions.None)[0]))

{

fromEmail = criteries.Split('+')[i].Split(new string[] { " >> " }, StringSplitOptions.None)[1];

}

}

MoveToFolder(message);

}

}

}

/// <summary>

/// Метод предназначен для извлечения вложений из письма (или сохранения текста письма в файл),

/// реализует извлечение файлов архива в подкаталог или в корневой каталог письма, в зависимости от конфигураций.

/// </summary>

/// <param name="message">Объект письма</param>

/// <param name="path">Путь к подкаталогу письма</param>

private void GetAttachments(ImapX.Message message, string path)

{

bool attachExist = false;

string body;

foreach (var attachments in message.Attachments)

{

attachments.Download();

attachments.Save(rootPath + path);

if (autoMode)

{

if (attachments.FileName.ToString().Split('.')[1] == "rar" || attachments.FileName.ToString().Split('.')[1] == "7z" || attachments.FileName.ToString().Split('.')[1] == "zip")

{

string s;

if (!mkNewDir)

s = rootPath + path.Replace("/", @"\") + @"\";

else

{

Directory.CreateDirectory(rootPath + path.Replace('/', '\\') + @"\" + attachments.FileName.ToString().Split('.')[0]);

s = rootPath + path.Replace("/",@"\") + @"\" + attachments.FileName.ToString().Split('.')[0] + @"\";

AddLog("Создан подкаталог для содержимого вложения " + attachments.FileName.ToString().Split('.')[0]);

}

SevenZipExtractor.SetLibraryPath(finfo.DirectoryName + @"\7z.dll");

SevenZipExtractor sze = new SevenZipExtractor(rootPath + path + @"\" + attachments.FileName.ToString());

sze.ExtractArchive(s);

AddLog("Содержимое архива успешно извлечено");

File.Delete(rootPath + path + @"\" + attachments.FileName.ToString());

}

else

{

AddLog("Найденый прикреплённый файл не являеться архивом");

}

}

attachExist = true;

}

if (!attachExist)

{

AddLog("Прикреплённых файлов нет");

body = message.Body.Text;

var newfile = File.Create(rootPath + path + @"\" + FileName(message.Date.ToString()));

newfile.Write(uniString.GetBytes(body), 0, uniString.GetByteCount(body));

newfile.Close();

AddLog("Был сохранён текст сообщения в файл " + FileName(message.Date.ToString()));

}

}

/// <summary>

/// Метод предназначен для формирования имени файла куда будет сохраняться текст письма, в котором нет вложений

/// </summary>

/// <param name="dateTime">Текущая дата-время в формате </param>

/// <returns>Строка сформированная из MessageData</returns>

private string FileName(string dateTime)

{

string date = dateTime.Split(' ')[0];

string[] arr;

arr = dateTime.Split(' ');

arr = arr[1].Split(':');

arr[0] = arr[0] + "h";

arr[1] = arr[1] + "m";

arr[2] = arr[2] + "s";

return date + " " + arr[0] + arr[1] + arr[2] + ".txt";

}

/// <summary>

/// Метод предназначен для извлечения настроек из конфигурационного файла

/// </summary>

private void GetConfInfo()

{

if (Configuration.WorkWithConfig.GetConfigValue("main\_rootPath", finfo.DirectoryName) == "Error")

{

this.errors.WriteLine("Проблемы с конфигурационным файлом");

this.errors.Flush();

OnStop();

}

rootPath = Configuration.WorkWithConfig.GetConfigValue("main\_rootPath", finfo.DirectoryName) + @"\";//

otherDir = Configuration.WorkWithConfig.GetConfigValue("main\_otherDir", finfo.DirectoryName);

login = Configuration.WorkWithConfig.GetConfigValue("server\_login", finfo.DirectoryName);//

password = Configuration.WorkWithConfig.GetConfigValue("server\_password", finfo.DirectoryName);//

server = Configuration.WorkWithConfig.GetConfigValue("server\_server", finfo.DirectoryName);//

name = Configuration.WorkWithConfig.GetConfigValue("log\_name", finfo.DirectoryName);//

criteries = Configuration.WorkWithConfig.GetConfigValue("main\_criteries", finfo.DirectoryName);//

frequency = Int32.Parse(Configuration.WorkWithConfig.GetConfigValue("main\_frequency", finfo.DirectoryName));//

port = Int32.Parse(Configuration.WorkWithConfig.GetConfigValue("server\_port", finfo.DirectoryName));//

safety = Int32.Parse(Configuration.WorkWithConfig.GetConfigValue("server\_safety", finfo.DirectoryName));//

maxSize = Int32.Parse(Configuration.WorkWithConfig.GetConfigValue("log\_maxSize", finfo.DirectoryName));

saveTime = Int32.Parse(Configuration.WorkWithConfig.GetConfigValue("log\_saveTime", finfo.DirectoryName));

autoMode = Boolean.Parse(Configuration.WorkWithConfig.GetConfigValue("archive\_autoMode", finfo.DirectoryName));//

mkNewDir = Boolean.Parse(Configuration.WorkWithConfig.GetConfigValue("archive\_mkNewDir", finfo.DirectoryName));//

}

/// <summary>

/// Метод для реализации ротации лог-файлов

/// </summary>

/// <returns></returns>

private FileStream LogRotate()

{

FileStream fs;

string lastLogFile;

lastLogFile = finfo.DirectoryName + pathToLog + name + ".txt";

string[] logs = Directory.GetFiles(finfo.DirectoryName + pathToLog, name + "\*");

foreach (var logfile in logs)

{

if (new FileInfo(logfile).CreationTime.CompareTo((new FileInfo(logfile).CreationTime.AddDays(saveTime))) > 0)

{

File.Delete(logfile);

}

if (new FileInfo(logfile).CreationTime.CompareTo(new FileInfo(lastLogFile).CreationTime) > 0)

{

lastLogFile = logfile;

}

}

fs = new FileStream(lastLogFile, System.IO.FileMode.Append);

if (new FileInfo(Path.GetFullPath(lastLogFile)).Length.CompareTo(Int64.Parse((maxSize \* 1024).ToString())) > 0)

{

string[] logNameParts = lastLogFile.Split('.');

int logVersion;

if (lastLogFile != finfo.DirectoryName + pathToLog + name + ".txt")

{

logVersion = int.Parse(logNameParts[logNameParts.Length - 2]);

logVersion++;

logNameParts[logNameParts.Length - 2] = logVersion.ToString();

}

else

{

logNameParts[logNameParts.Length - 2] += ".1";

}

fs = new FileStream(String.Join(".", logNameParts), System.IO.FileMode.Append);

}

return fs;

}

/// <summary>

/// Метод для добавления записи в лог-файл

/// </summary>

/// <param name="text">Текст для записи в лог-файл</param>

private void AddLog(string text)

{

this.file.WriteLine(text);

this.file.Flush();

}

/// <summary>

/// Метод для переноса сообщения в нужную папку

/// </summary>

/// <param name="message"></param>

private void MoveToFolder(ImapX.Message message)

{

if (fromEmail != "")

{

foreach (var folder in client.Folders)

{

if (folder.Name == fromEmail.Split('/')[0])

{

message.MoveTo(client.Folders[fromEmail.Split('/')[0]]);

AddLog("Сообщение помещено в " + fromEmail);

GetAttachments(message, fromEmail);

isExist = true;

break;

}

}

if (!isExist)

{

client.Folders.Add(fromEmail.Split('/')[0]);

Directory.CreateDirectory(rootPath + fromEmail);

AddLog("Была создана директория " + fromEmail);

message.MoveTo(client.Folders[fromEmail.Split('/')[0]]);

AddLog("Сообщение помещено в " + fromEmail);

GetAttachments(message, fromEmail);

}

}

else

{

message.MoveTo(client.Folders[otherDir]);

AddLog("Сообщение помещено в " + otherDir);

GetAttachments(message, otherDir);

}

}

}

# ПРИЛОЖЕНИЕ В. РУКОВОДСТВО ОПЕРАТОРА

**П.В.1 Назначение программы**

Данное программное обеспечение может быть использовано для добавления возможности получения входной информации по электронной почте для систем, которые по умолчанию получают эту информацию из файлов, находящихся в каталогах файловой системы (старые системы).

Функциональным назначением программы является получение писем и распределение их по каталогам в зависимости от файла настроек с возможностью разархивирования прикреплённых архивов.

**П.В.2 Условия выполнения программы**

В состав технических средств должен входить х64 совместимый компьютер, включающий в себя:

1) процессор с тактовой частотой 2.7Ghz;

2) оперативную память объемом 4GB, не менее;

3) свободного дискового пространства 100MB

4) дисплей и видеоадаптер любого типа, позволяющий отображать информацию в текстовом режиме;

5) клавиатуру и мышь;

6) жесткий диск;

7) сетевую плату.

На ПК должны быть установлены:

1) ОСWindows 10x64;

2) .Net 4.5;

3) WF 2.0.

**П.В.3 Установка и удаление программы**

П.В.3.1 Установка

Для установки программы нужно выполнить следующие действия:

1. Запустить установщик «Setup.exe» и следовать инструкциям;
2. После установки запустить на выполнение от имени администратора файл «Install.bat», который будет находиться в папке с установленной программой (этот шаг установит службу MailClient).

Содержимое файла «Install.bat»:

*net stop MailClient*

*sc delete MailClient*

*pause*

П.В.3.2 Удаление

Для удаления программы нужно выполнить следующие действия:

1. Запустить на выполнение от имени администратора файл «uninstalServ.bat» (он находится в папке с установленной программой) это удалит службу компьютера.

Содержимое файла «uninstalServ.bat»:

*C:\Windows\Microsoft.NET\Framework\v4.0.30319\installutil.exe "%~dp0\MailClient\MailClient.exe"*

*net start MailClient*

*pause*

1. Чтобы удалить конфигуратор нужно запустить деинсталятор «unins000.exe» (он находится в папке с установленной программой) и следовать инструкциям.

**П.В.4 Выполнение программы**

1. Запустить приложение «Configuration.exe»;
2. Указать корневой каталог (здесь будут храниться подпапки с файлами из писем)

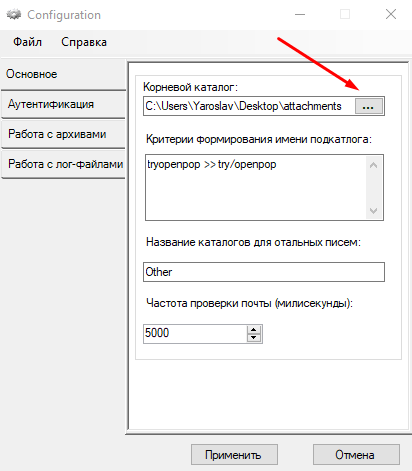


Рисунок П.В.1 – Корневой каталог

1. Указать критерии распределения писем. Синтаксис следующий:

*шаблон отправителя* >> *каталог, в который помещать письма ,*

где шаблон отправителя – подстрока, которая будет находиться в имени отправителя.

В имени каталогов возможна вложенность с помощью ‘/’.

*Пример:*

*@gmail.com >> gmail* (файлы письма сохранятся в каталоге “gmail”)

*example >> example/mails* (файлы письма сохранятся в каталоге “example” в подкаталоге “mails”)

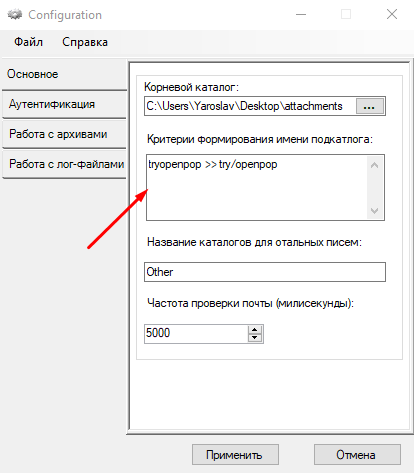


Рисунок П.В.2 – Критерии распределения

1. Указать название каталога для писем, которые не подошли ни под один критерий;

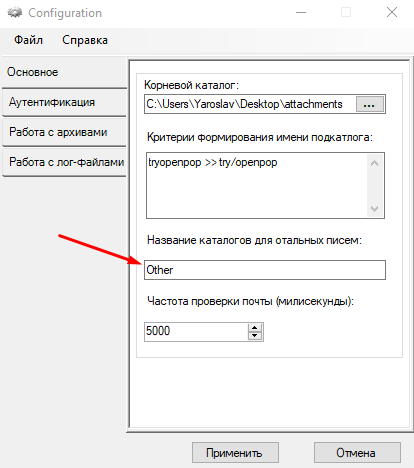


Рисунок П.В.3 – Каталог для остальных писем

1. Указать частоту, с которой служба будет обращаться к почтовому серверу;

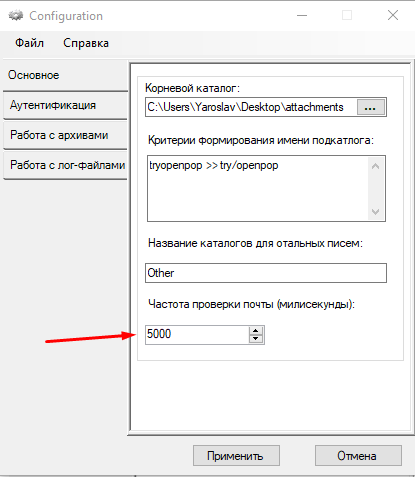


Рисунок П.В.4 – Частота обращения к почтовому серверу

1. Выбрать вкладку «Аутентификация», там указать всю запрашиваемую информацию;

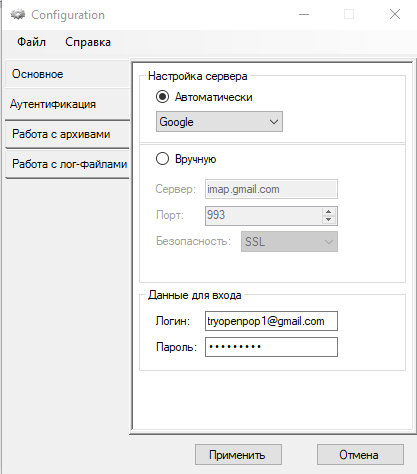


Рисунок П.В.5 – Аутентификация

1. Выбрать вкладку «Работа с архивами» и выбрать режим, в котором Вы хотите распаковывать прикреплённый архив;

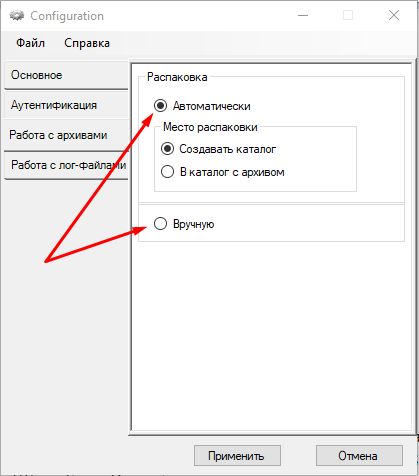


Рисунок П.В.6 – Режим распаковки

1. Если Вы выбрали автоматическое извлечение архивов, то можно выбрать функцию создания дополнительного подкаталога для содержимого распаковываемого архива;

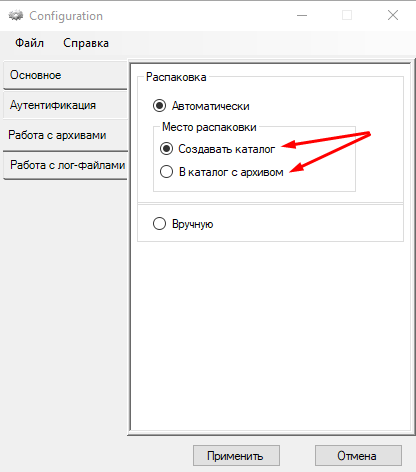


Рисунок П.В.7 – Место распаковки

1. Выбрать вкладку «Работа с лог файлами», там указать имя лог-файлов; максимальный размер, по достижению которого будет создаваться новый и количество дней, в течение которого необходимо хранить лог-файлы;

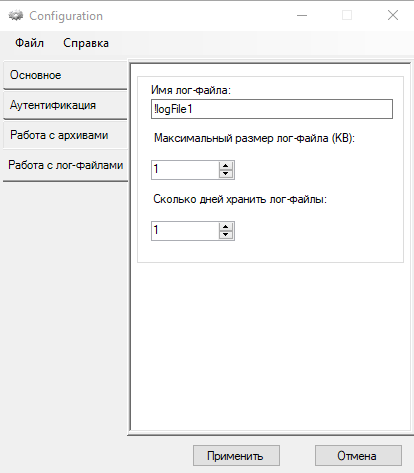


Рисунок П.В.8 – Работа с лог файлами

1. Нажать кнопку «Применить» и подтвердить свои действия.

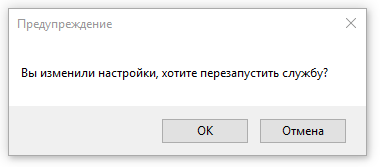


Рисунок П.В.9 – Перезапуск службы

Все записи о работе корректной работе службы будет вестись в лог файле “папка\_с\_приложением /log/имя\_лог\_файла.txt”. А записи об ошибках запуска службы в “папка\_с\_приложением /log/ErrorLog.txt”.

Файл конфигурации (“папка\_с\_приложением /config/Settings.config”) служит для настройки работы службы, сюда записываются настройки, которые выбрал пользователь в конфигураторе.

В папке программы также присутствуют библиотеки для работы с архивами («7zip.dll», «SevenZipSharp.dll»), с помощью которых будет проводиться извлечение файлов из архива.

Библиотека «ImapX.dll» служит для работы с IMAP сервером.

**Важно! Нарушать структуру или удалять файлы проекта нельзя!**

**П.В.4 Сообщениея оператору**

В ходе выполнения программы оператору могут быть выданы следующие сообщения:

1) При запуске конфигуратора, если из папки config был удалён файл конфигурации, будет выдано сообщение об ошибке, представленное на рисунке П.В.10. При появлении такой ошибки, файл с конфигурациями утерян, и исправить это можно только переустановив программу.

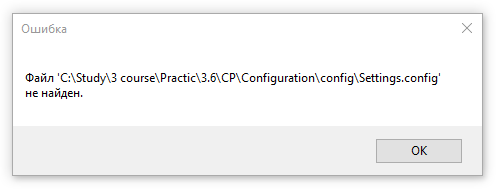


Рисунок П.В.10 – Ошибка при отсутствии файла конфигураций

2) При нажатии на кнопку «Принять» в конфигураторе будет выдано диалоговое окно подтверждения действий, рисунок П.В.11;

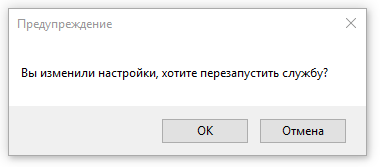


Рисунок П.В.11 – Подтверждение перезапуска службы

# ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ



Рисунок П.Г.1 – Слайд 1



Рисунок П.Г.2 – Слайд 2

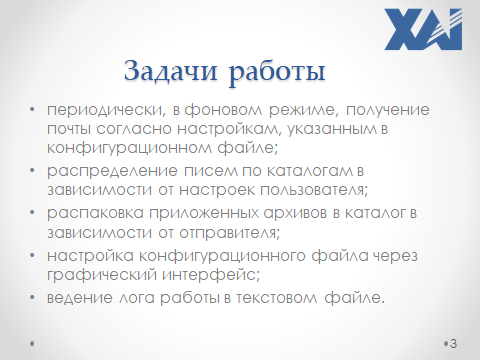


Рисунок П.Г.3 – Слайд 3

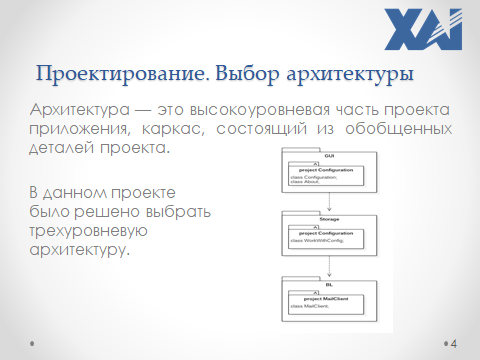


Рисунок П.Г.4 – Слайд 4



Рисунок П.Г.5 – Слайд 5



Рисунок П.Г.6 – Слайд 6

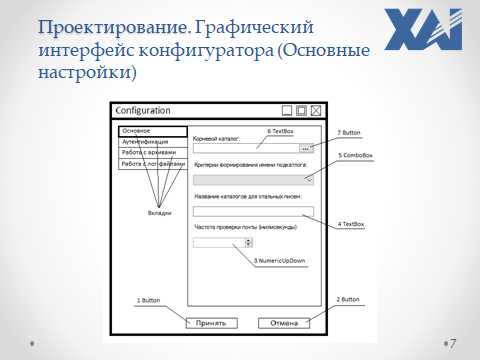


Рисунок П.Г.7 – Слайд 7

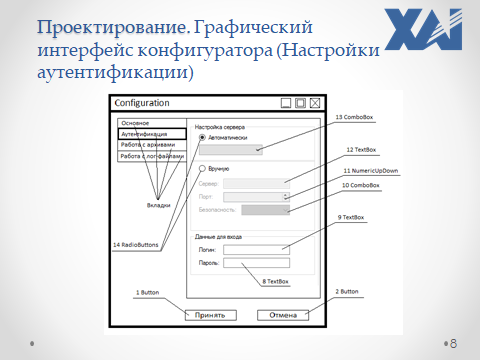


Рисунок П.Г.8 – Слайд 8

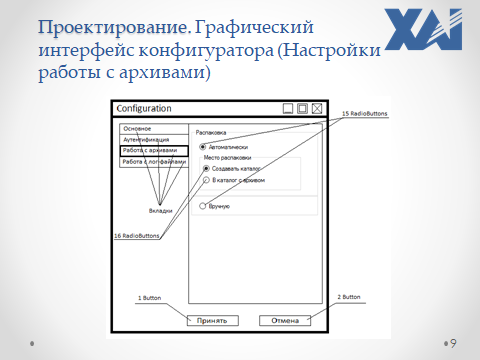


Рисунок П.Г.9 – Слайд 9

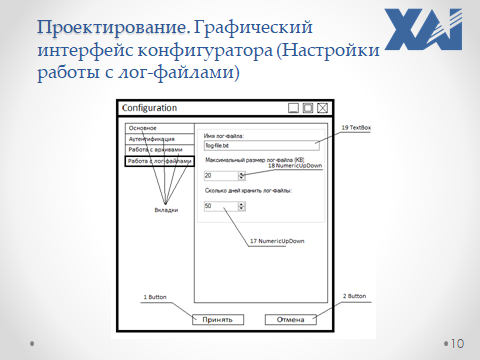


Рисунок П.Г.10 – Слайд 10

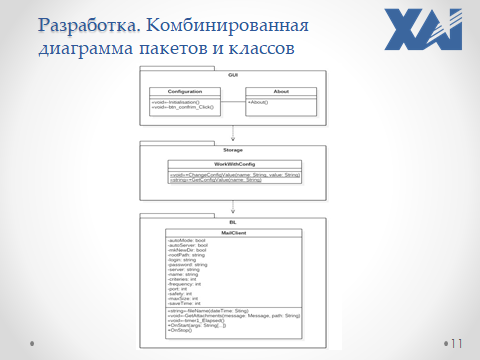


Рисунок П.Г.11 – Слайд 11

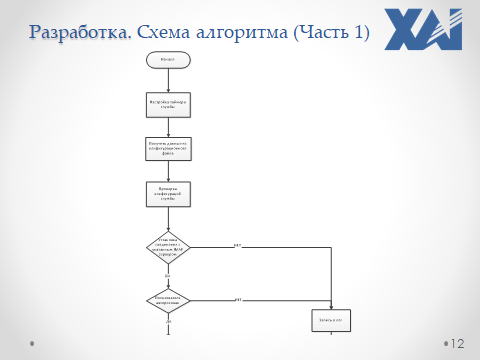


Рисунок П.Г.12 – Слайд 12

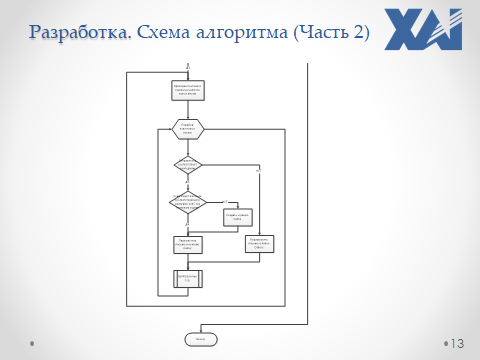


Рисунок П.Г.13 – Слайд 13

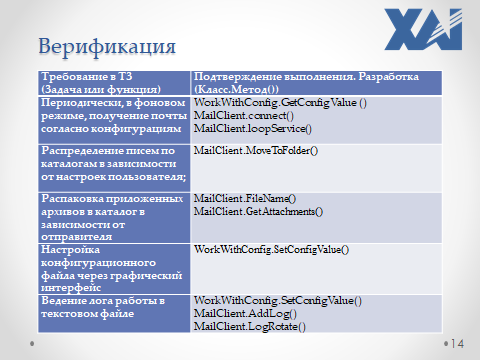


Рисунок П.Г.14 – Слайд 14



Рисунок П.Г.15 – Слайд 15



Рисунок П.Г.16 – Слайд 16

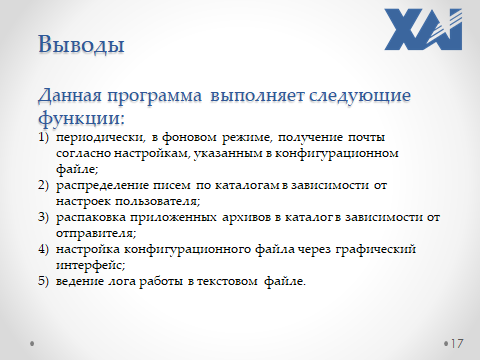


Рисунок П.Г.17 – Слайд 17

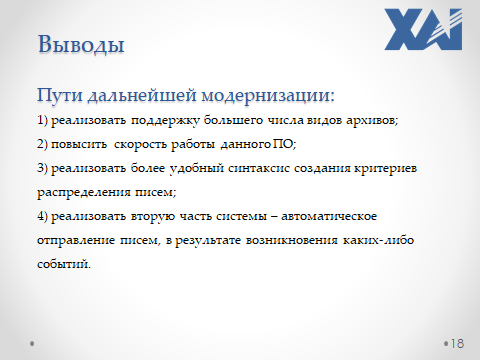


Рисунок П.Г.18 – Слайд 18



Рисунок П.Г.19 – Слайд 19