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

СУБД – система управления базами данных

БД – база данных

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc500172814)

[1 ПОСТАНОВКА ЗАДАЧИ 6](#_Toc500172815)

[1.1 Характеристика предметной области 6](#_Toc500172816)

[2 ПРОЕКТИРОВАНИЕ БД 7](#_Toc500172817)

[2.1 Обоснование выбора СУБД 7](#_Toc500172818)

[2.2 Логическая и физическая модель 7](#_Toc500172819)

[2.3 Описание сущностей и бизнес-правила 9](#_Toc500172820)

[3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ 12](#_Toc500172821)

[3.1 Описание клиентской части 12](#_Toc500172822)

[3.1.1 Диаграмма вариантов использования 12](#_Toc500172823)

[3.2 Архитектура клиентского ПО 18](#_Toc500172824)

[3.2.1 Диаграмма пакетов 18](#_Toc500172825)

[3.2.2 Общий алгоритм работы ПО 19](#_Toc500172826)

[3.3 Описание программных компонентов 23](#_Toc500172827)

[4 ТЕСТИРОВАНИЕ 24](#_Toc500172828)

[4.1 Выполнения тестов 24](#_Toc500172829)

[ВЫВОДЫ 26](#_Toc500172830)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc500172831)

[ПРИЛОЖЕНИЕ А. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 28](#_Toc500172832)

[ПРИЛОЖЕНИЕ В. СКРИПТЫ СОЗДАНИЯ БАЗЫ ДАННЫХ 33](#_Toc500172833)

[ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ 35](#_Toc500172834)

# 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 Характеристика предметной области

Для прогнозирования движения цены акций, разрабатываемому индикатору потребуются цены данной акции за предыдущие периоды. Индикатор может рассчитывать цены для нескольких акций. Каждая акция характеризуется именем.

У каждой акции есть множество цен с данными о дате; ценах открытия и закрытия; максимальной и минимальной цене; прирост цены; вероятность роста цены на текущий момент и значение показывающее выросла ли цена. Прирост цены и значение указывающее на то выросла ли цена, устанавливается автоматически на основе цены открытия и закрытия. Прирост цены вычисляется как разница между ценой закрытия и открытия. Если прирост цены – значение положительное, то значение указывающее выросла ли цена будет истинно, а если нет, то ложное.

У каждой цены в свою очередь есть ряд индикаторов, которые имеют название и значение вероятности роста данной цены.

На основании индикаторов, для каждой цены можно посчитать вероятность роста цены, которая будет равна усреднённому значению вероятности роста по индикаторам.

Для каждой цены можно рассчитать прогноз движения цены, у этого прогноза будет значение даты, на которую производится прогноз и предполагаемая цена.

# 2 ПРОЕКТИРОВАНИЕ БД

## 2.1 Обоснование выбора СУБД

Система управления базами данных (СУБД) — комплекс программных и лингвистических средств общего или специального назначения, реализующий поддержку создания баз данных, централизованного управления и организации доступа к ним различных пользователей в условиях принятой технологии обработки данных.

В данной работе было решено использовать одну из самых популярных СУБД Microsoft SQL Server.

Выбор обоснован следующими пунктами:

– очень высока степень интеграции с другими продуктами Microsoft, в частности с Microsoft Visual Studio;

* хорошее быстродействие и высокая стабильность;
* наличие опыта работы с данной СУБД;
* программируемость с помощью T-SQL;
* удобное ПО для администрирования БД, а так же наличие средств удалённого доступа.

## 2.2 Логическая и физическая модель

В следствии анализа предметной области была спроектирована логическая модель базы данных, которая приварена на рисунке 2.1.

Как видно из рисунка, были выделены следующие сущности:

– Stock (акция);

– Price (цена акции в конкретный момент времени);

– ExternalIndicator (информация с индикатора для конкретной цены);

– StockForecast (прогноз движения цены для конкретной акции).

Связь между таблицей Stock и Price один ко многим, т.к. у каждой акции может быть множество цен в разные моменты времени. Связь между таблицей Price и ExternalIndicator так же один ко многим, это обуславливается тем, что для каждой цены определённой акции можно посмотреть прогноз движения цены с разных индикаторов. Связь между таблицей Stock и StockForecast тоже один ко многим, т.к. для каждой акции мы можем прогнозировать движение цены в разные моменты времени.

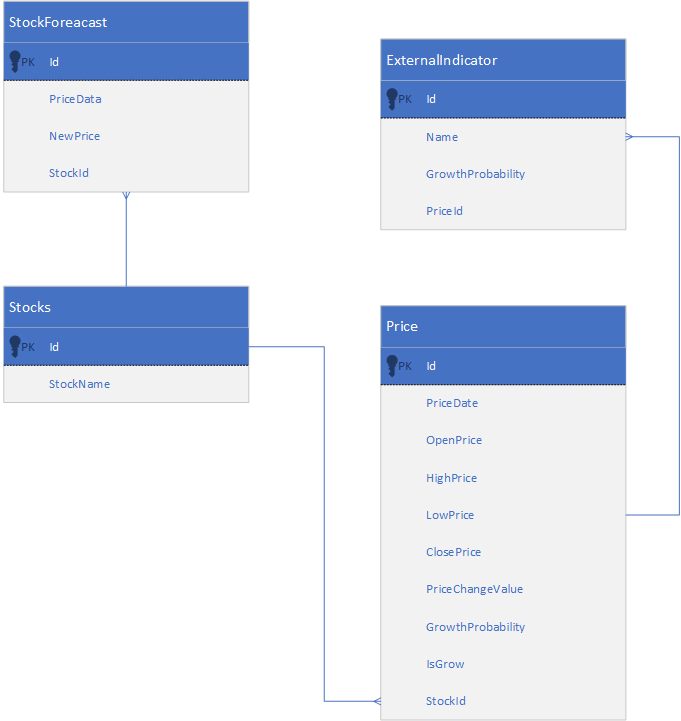


Рисунок 2.1 – Логическая модель базы данных

Физическая модель проектируемой базы данных изображена на рисунке 2.2. Типы данных используемые в ней соответствуют смысловой нагрузке полей, а так же выбранной СУБД.

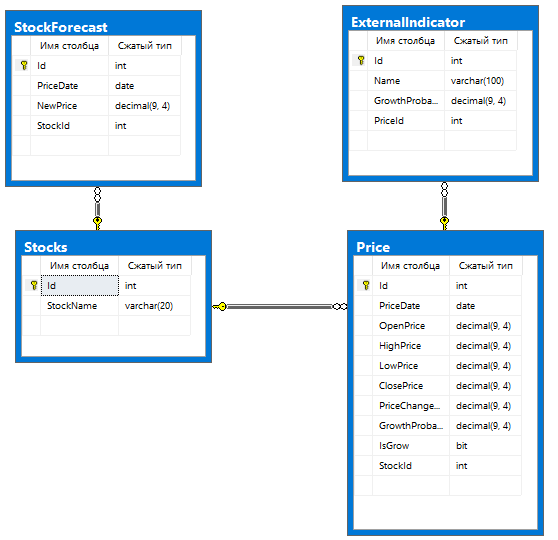


Рисунок 2.2 – Физическая модель базы данных

## 2.3 Описание сущностей и бизнес-правила

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).

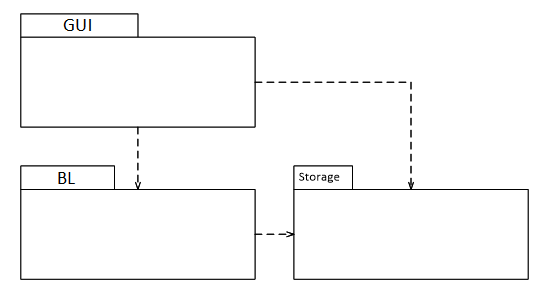


Рисунок 2.1 – Архитектура данного проекта

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.

# 3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

## 3.1 Описание клиентской части

### 3.1.1 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.

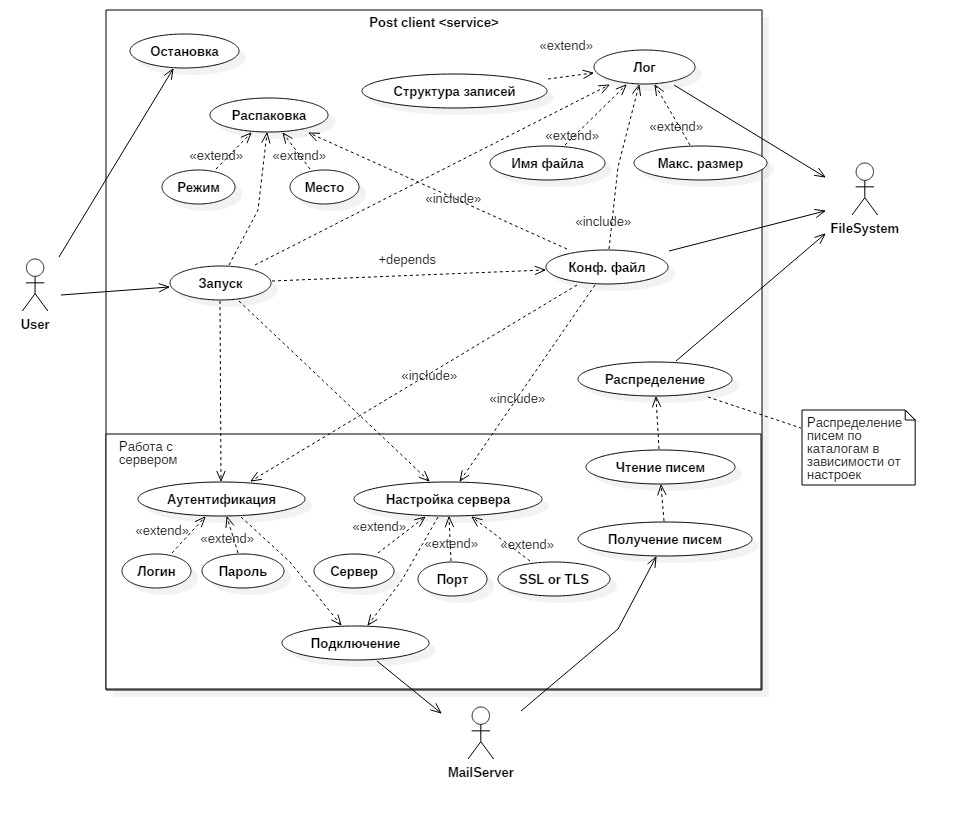


Рисунок 2.2 – Диаграмма вариантов использования

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» |

## 3.2 Архитектура клиентского ПО

### 3.2.1 Диаграмма пакетов

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

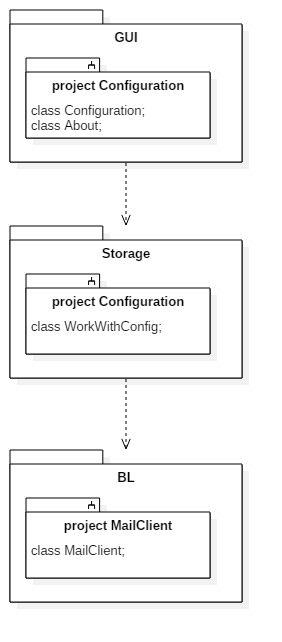


Рисунок 3.1 – Диаграмма пакетов проекта

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.

### 3.2.2 Общий алгоритм работы ПО

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 ()

## 3.3 Описание программных компонентов

# 4 ТЕСТИРОВАНИЕ

## 4.1 Выполнения тестов

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 ­– "Создание базы данных для разработки индикатора технического анализа на основе машинного обучения".

1.2 Краткая характеристика области применения

Разрабатываемое ПО может быть использовано для анализа рынка акций и построения прогноза движения цены определённого актива. Прогноз формируется на основе использования алгоритмов машинного обучения.

2 Основания для разработки

2.1 Основание для проведения разработки

Основание для разработки – задание на курсовой проект по дисциплине "Организация баз данных", 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 – "Создание базы данных для разработки индикатора технического анализа".

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

3 Назначение разработки

3.1 Функциональное назначение

Предполагается, что программный продукт будет использоваться в финансовой сфере, а именно для трейдинга. Данное ПО позволит трейдеру, с определённой вероятностью, прогнозировать движение цены.

Функциональным назначением программы является прогнозирование цены акции на основе предыдущих цен этого же актива и сохранение этого значения в БД.

3.2 Эксплуатационное назначение

Режим работы данного ПО – ручной режим.

Длительность работы данного ПО – однократный запуск программы, выполнение своих функций и завершение.

Условия эксплуатации программы – офис и домашние условия.

Обслуживание данного ПО не предполагается.

Для установки данной программы необходимо запустить на выполнение 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. добавление отслеживаемой акции в БД по имени;
2. загрузка списка цен на определённую акцию из файла .csv (файл должен иметь содержание, предоставляемое Yahoo Finance) и добавление их в БД;
3. добавление индикатора (имя индикатора и вероятность роста цены) для цены за конкретный момент времени в БД;
4. расчёт вероятности роста конкретной цены по данным из индикаторов добавленным для неё и обновление соответствующей записи в БД;
5. прогнозирование движения цены на основе имеющихся данных с помощью машинного обучения и добавление прогноза в БД.

4.1.2 Требования к организации входных данных

Данные которые мы получаем из файла (цены для акции) должны находиться в файле формата .csv, эти файлы могут быть получены из ресурса Yahoo Finance. Они должны иметь определённое количество колонок, размещённые в определённом формате. Пример файла находиться в ПРИЛОЖЕНИИ Б.

4.1.3 Требования к организации выходных данных

Выходные данные будут представлены как новые записи в БД.

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

Информационная структура входного файла, должна включать в себя разметку, описанную в пункте 4.1.2 ПРИЛОЖЕНИЕ А. Также входные данные хранятся в БД, поддерживаемой СУБД MS SQL.

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. с поддержкой СУБД MS SQL.

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) SQL Server 2014 Management Studio.

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) скрипты создания БД и её объектов;

г) план тестирования и верификации;

д) 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 базы данных;

2) разработка программы;

3) разработка программной документации;

4) 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) выбор СУБД;

7) согласование и утверждение технического задания.

На этапе разработки базы данных должна быть выполнена работа по созданию БД и её объектов.

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. ФОРМАТ СОДЕРЖИМОГО ФАЙЛА ЦЕН



# ПРИЛОЖЕНИЕ В. СКРИПТЫ СОЗДАНИЯ БАЗЫ ДАННЫХ

CREATE DATABASE StockIndicatorDB;

GO

USE StockIndicatorDB

CREATE TABLE Stocks(

Id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

StockName VARCHAR(20) NOT NULL

);

CREATE TABLE Price(

Id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

PriceDate DATE,

OpenPrice DECIMAL(9,4),

HighPrice DECIMAL(9,4),

LowPrice DECIMAL(9,4),

ClosePrice DECIMAL(9,4),

PriceChangeValue DECIMAL(9,4),

GrowthProbability DECIMAL(9,4),

IsGrow BIT, -- 1 is UP, 0 is DOWN

StockId INT FOREIGN KEY REFERENCES Stocks(Id)

);

CREATE TABLE ExternalIndicator(

Id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(100),

GrowthProbability DECIMAL(9,4),

PriceId INT NOT NULL FOREIGN KEY REFERENCES Price(Id)

);

CREATE TABLE StockForecast(

Id INT NOT NULL IDENTITY(1,1) PRIMARY KEY,

PriceDate DATE,

NewPrice DECIMAL(9,4),

StockId INT NOT NULL FOREIGN KEY REFERENCES Stocks(Id)

);

GO

CREATE FUNCTION GetTotalProb (@priceId INT)

RETURNS DECIMAL(9,4)

AS

BEGIN

DECLARE @res DECIMAL(9,4) = 0

SELECT @res = AVG(GrowthProbability)FROM ExternalIndicator WHERE PriceId = @priceId AND GrowthProbability != 0

RETURN @res

END

GO

CREATE PROCEDURE dbo.GetGrowthProbabiliries @priceId INT

AS

SELECT \* FROM ExternalIndicator WHERE PriceId = @priceId

GO

CREATE PROCEDURE dbo.SetGrowthProb @priceId INT

AS

UPDATE Price SET Price.GrowthProbability = dbo.GetTotalProb(@priceId) WHERE Price.Id = @priceId

GO

CREATE TRIGGER SetIsGrow ON Price AFTER INSERT AS

BEGIN

DECLARE @openPrice DECIMAL(9,4)

SELECT @openPrice = OpenPrice FROM inserted

DECLARE @closePrice DECIMAL(9,4)

SELECT @closePrice = ClosePrice FROM inserted

IF(@closePrice>@openPrice)

UPDATE Price SET IsGrow = 1 WHERE Price.Id = (SELECT id FROM inserted)

ELSE

UPDATE Price SET IsGrow = 0 WHERE Price.Id = (SELECT id FROM inserted)

END

GO

CREATE TRIGGER SetPriceChangeValue ON Price AFTER INSERT AS

BEGIN

DECLARE @openPrice DECIMAL(9,4)

SELECT @openPrice = OpenPrice FROM inserted

DECLARE @closePrice DECIMAL(9,4)

SELECT @closePrice = ClosePrice FROM inserted

UPDATE Price SET PriceChangeValue = @closePrice - @openPrice WHERE Price.Id = (SELECT id FROM inserted)

END

GO

# ПРИЛОЖЕНИЕ Г. ПРЕЗЕНТАЦИЯ