# 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).

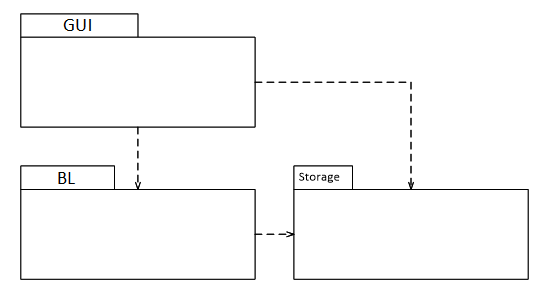


Рисунок 2.1 – Архитектура данного проекта

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.

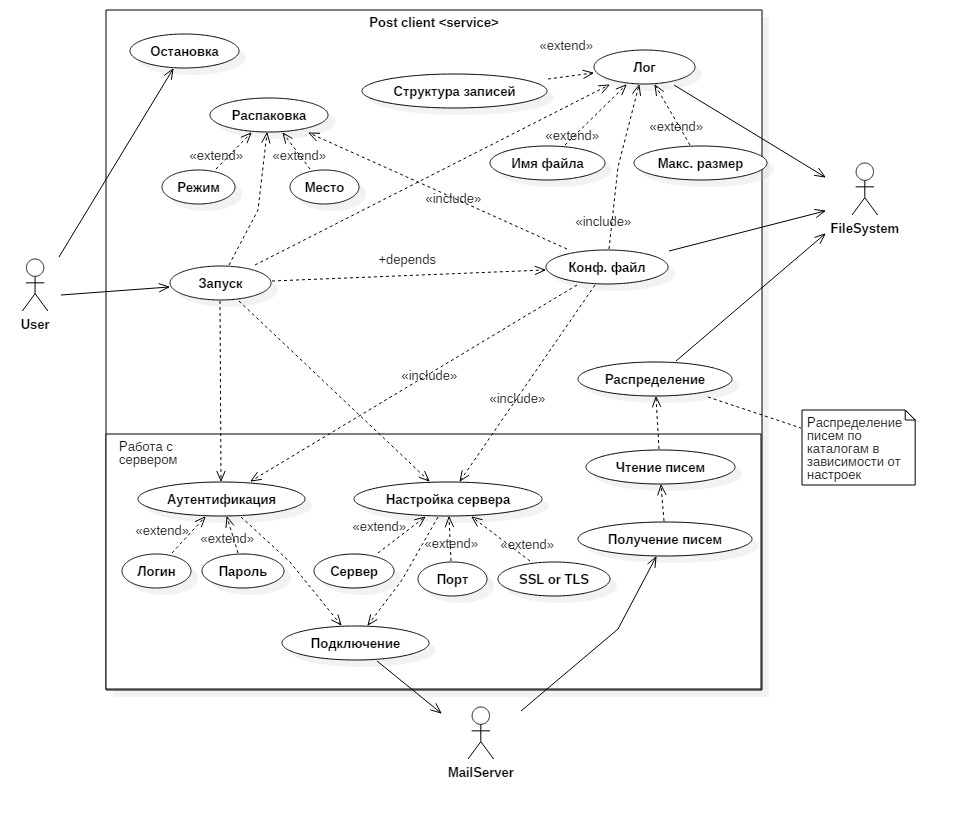


Рисунок 2.2 – Диаграмма вариантов использования

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 |

|  |  |
| --- | --- |
| **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) |
| **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].

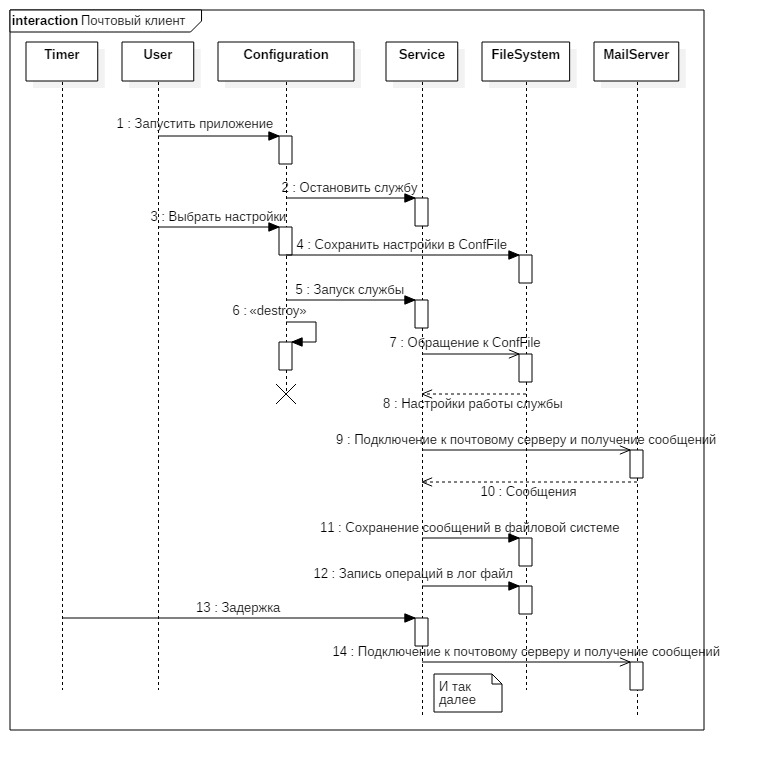


Рисунок 2.3 – Диаграмма последовательности процесса работы всех компонентов разрабатываемого ПО

Данная диаграмма показывает порядок взаимодействия элементов модели и соответствующих событий в конкретно в данном проекте. Главная линия 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.

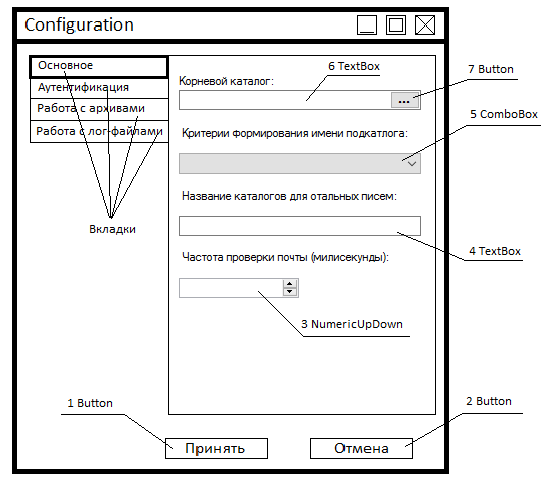


Рисунок 2.4 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Основное»)

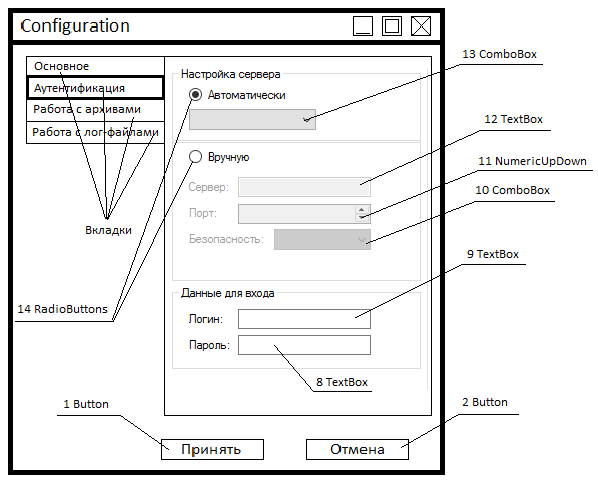


Рисунок 2.5 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Аутентификация»)

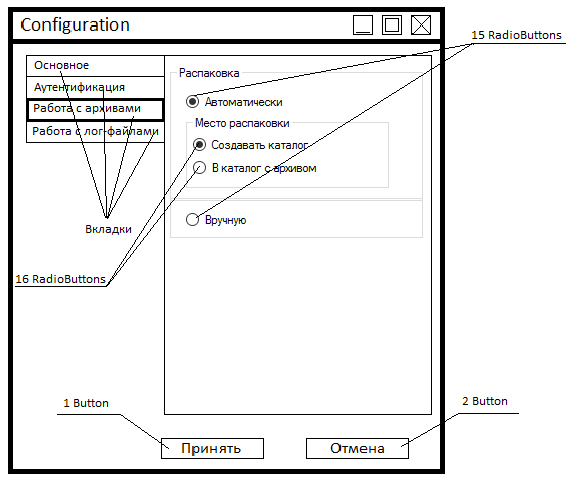


Рисунок 2.6 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Работа с архивами»)

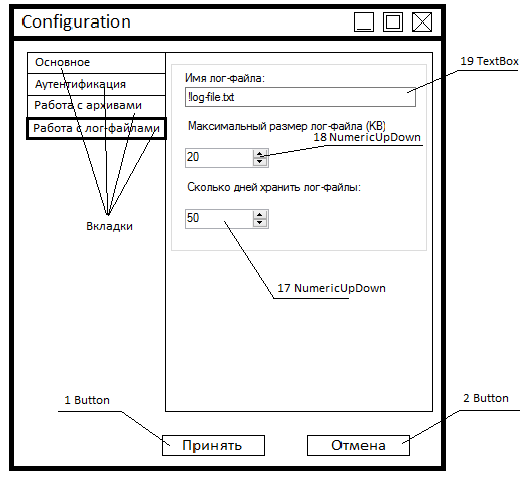


Рисунок 2.7 – Предположительный графический интерфейс программы настройки конфигурационного файла (вкладка «Работа с лог-файлами»)

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. |
| **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.