

Министерство высшего образования Российской Федерации
Пензенский государственный университет
Кафедра «Вычислительная техника»

Пояснительная записка
к курсовой работе
по курсу «Технологии программирования»
на тему «Разработка объектно-ориентированной программы с
пользовательским интерфейсом»

Выполнил:
студент группы 18BB1
Немцев А.А.

Принял:
к.т.н., доцент Дорошенко И.Н.

486
26.03.2022
Дорошенко

Пенза 2021

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет вычислительной техники

Кафедра "Вычислительная техника"

"УТВЕРЖДАЮ"

Зав. Кафедрой ВТ

 М.А. Митрохин

"__" ____ 20__ г

ЗАДАНИЕ

на курсовое проектирование по курсу

Технологии программирования
Студенту Немицву А.А. Группа 18ВВ1
Тема проекта Разработка объектно-ориентированной программы с пользовательским интерфейсом

Исходные данные (технические требования) на проектирование

Требуется разработать приложение "Почтовый клиент". Приложение должно иметь следующий функционал: отправка, получение писем, их хранение и удаление; возможность синхронизации с почтовым ящиком пользователя. В работе использовать протоколы POP3 и SMTP.

Общие требования к программе:

1. Операционная система - MS Windows;
2. Система программирования - MS Visual Studio;
3. Язык программирования - C++/C#

Объем работы по курсу

1. Расчетная часть

1. Структура данных используемых объектов
2. Описание арх. разработанного приложения и способа организации польз. интерфейса
3. Описание программы
4. Описание руководства работы и их анализ
5. Руководство пользователя

2. Графическая часть

Диаграммы ИМЛ

3. Экспериментальная часть

Отладка программы

Срок выполнения проекта по разделам

- 1 В соответствии с графиком выполнения курсовой
- 2 работы
- 3
- 4
- 5
- 6
- 7
- 8

Дата выдачи задания "18" 09. 2021

Дата защиты проекта " " "

Руководитель Доросевич

Задание получил "18" 09 2021г.

Студент Немцев

Содержание

1 Постановка задачи	6
2 Выбор решения	7
2.1 Определение структуры программы	7
2.2 Описание основного цикла работы	8
3 Описание разработки программы	9
4 Отладка и тестирование	11
5 Описание программы	12
6 Руководство пользователя	17
Заключение	24
Список используемых источников	25
Приложение А Листинг модуля MailClient.exe	26
Приложение А.1 Файл «LoginInWindow.cs»	26
Приложение А.2 Файл «NewMailForm.cs»	27
Приложение А.3 Файл «ReadMailForm.cs»	29
Приложение А.4 Файл «Userdata.cs»	30
Приложение А.5 Файл «Connections.cs»	31
Приложение А.6 Файл «LoggedInWindow.cs»	32
Приложение Б. UML-диаграммы	36
Приложение Б.1 UML-диаграмма вариантов использования	36
Приложение Б.2 UML-диаграмма состояний программы	36

Введение

В развитии электронно-вычислительной техники принято условно выделять несколько поколений. Последние обычно рассматривают во взаимосвязи с применяемой элементной базой.

Последняя информационная революция, связанная с изобретением микропроцессорной технологии и появлением персонального компьютера, выдвигает на первый план новую отрасль – информационную индустрию. Бурное развитие компьютерной техники и информационных технологий послужило толчком к развитию общества, построенного на использование различной информации и получившего название информационного общества.

Деятельность отдельных людей, групп, коллективов и организаций сейчас всё в большей степени начинает зависеть от их информированности и способности эффективно использовать имеющуюся информацию. Прежде чем провести какие-то действия, необходимо провести большую работу по сбору и переработке информации, её осмыслению и анализу. Отыскание рациональных решений в любой сфере требует обработки больших объёмов информации, что подчас невозможно без привлечения специальных технических средств.

Возрастание объёмов информации стало особенно заметным в середине XX века. Информационный кризис поставил общество перед необходимостью поиска путей выхода из создавшегося положения. Внедрение ЭВМ, современных средств переработки и передачи информации в различные сферы деятельности послужило началом нового эволюционного процесса.

1 Постановка задачи

Требуется разработать приложение «Почтовый клиент». Приложение должно иметь следующий функционал: отправка, получение писем, их хранение и удаление; возможность синхронизации с почтовым ящиком пользователя. В работе использовать протоколы pop3 и smtp.

Общие требования к программе:

1. Операционная система - MS Windows;
2. Система программирования - MS Visual Studio;
3. Язык программирования - C++/ C#.

2 Выбор решения

2.1 Определение структуры программы

Сама программа должна состоять из основных частей:

- окно аутентификации;
- совмещенное окно, представляющее из себя меню навигации и основное рабочее пространство;
- окно для чтения писем;
- окно для написания нового письма.

Сами части должны выполнять соответствующие функции. Окно аутентификации должно предоставлять пользователю возможность ввести в программу все необходимые личные данные (адрес почтового ящика и пароль от него) для входа, так же стоит предоставить пользователю возможность сохранить введенные данные.

В окне чтения писем должен отображаться текст письма, его заголовок и отправитель.

В окне для отправки писем программа должна будет предоставлять пользователю ввести все необходимые данные для отправки письма (адрес отправки, тема письма, текст письма, подпись автора и вложение). Схема данных изображена на рисунке 1.

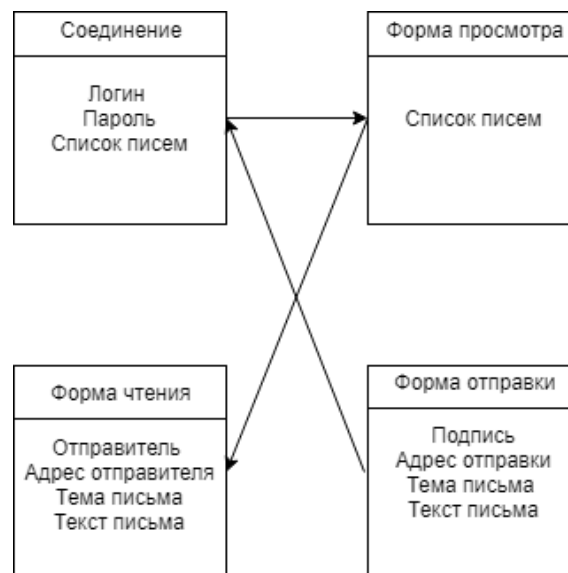


Рисунок 1 – Схема данных

2.2 Описание основного цикла работы

Основной цикл программы должен заключаться в следующем:

- отобразить пользователю окно аутентификации, предоставляющее пользователю возможность ввести в программу все необходимые личные данные.
- загрузить список писем для первой страницы из папки INBOX.
- отобразить загруженные письма в пользовательском интерфейсе.
- ожидать действий пользователя.
- если пользователь захотел просмотреть выбранное письмо, то необходимо отобразить в интерфейсе все данные, которые можно получить через протокол IMAP, а именно – отправитель письма, адрес отправителя, тема письма и текст письма.
- если пользователю необходимо написать новое письмо программа должна будет предоставлять пользователю ввести в программу все необходимые данные для отправки письма (адрес отправки, тема письма, текст письма, подпись автора и вложение).

Управление программой должно осуществляться по событийному варианту, так как автономно (независимо от действий пользователя) программа не должна ничего обрабатывать. Это значит, что все действия в программе должны осуществляться по определённым событиям, которые вызывает пользователь.

Например, окно написание нового письма должно открываться только после нажатия соответствующей клавиши, точно так же должна соблюдаться логика при других событиях.

Диаграмма вариантов использования программы и диаграмма состояний программы представлены на рисунках 15 и 16 соответственно (Приложение Б).

3 Описание разработки программы

Для написания данной программы был использован язык программирования С#. Этот язык является распространённым языком программирования. Он сочетает в себе высокую эффективность низкоуровневых языков с возможностями высокоуровневых языков (быстрое создание больших сложных проектов, сводя к минимуму отладку и тестирование). В языке С# предусмотрено много операций, непосредственно исполняемых большинством микропроцессоров; в то же время он дает программисту возможность выражать свои алгоритмы и данные наиболее подходящими средствами.

В качестве среды программирования был выбран программный продукт Visual Studio 2017.

Для передачи писем на почтовые ящики будет использоваться протокол SMTP — простой протокол передачи почты. С английского языка переводится, как *Simple Mail Transfer Protocol*. Исходя из названия, можно сделать вывод, что SMTP сервер отвечает за отправку почтовых рассылок. Его задача, как правило, состоит из двух основных функций:

1. проверка правильности настроек и выдача разрешения компьютеру, который пытается отправить электронное сообщение;
2. отправка исходящего сообщения на указанный адрес и подтверждение успешной отправки сообщения. Если доставка невозможна, сервер возвращает отправителю ответ с ошибкой отправки.

Отправляя email сообщения, SMTP-сервер отправителя устанавливает связь с тем сервером, который будет получать это сообщение. Такое "общение" происходит путем отправки и получения команд, формируя SMTP-сессию с неограниченным количеством SMTP-операций.

Для получения данных о пользователе будет использоваться протокол IMAP (англ. Internet Message Access Protocol) — протокол прикладного уровня для доступа к электронной почте.

IMAP предоставляет пользователю широкие возможности для работы с почтовыми ящиками, находящимися на почтовом сервере. Почтовая программа, использующая этот протокол, получает доступ к хранилищу корреспонденции на сервере так, как будто эта корреспонденция расположена на компьютере получателя. Электронными письмами можно манипулировать с компьютера пользователя (клиента) без постоянной пересылки с сервера и обратно полного содержания писем.

При просмотре почты с использованием протокола POP3 все электронные письма скачиваются пользователю на компьютер и автоматически удаляются с сервера. Все дальнейшие действия с письмами будут производиться именно на компьютере пользователя. Протокол POP3 может быть удобен, если почтовым ящиком пользуется лишь один человек с одного-единственного компьютера, а доступ в интернет ограничен или непостоянен.

При использовании протокола IMAP электронные письма всегда находятся на сервере и синхронизируются с локальными клиентами. Это очень удобно, когда необходим доступ к почтовому ящику с нескольких устройств (компьютер, ноутбук, телефон, планшет), или, когда доступ должны иметь несколько человек — все операции с письмами, которые произведены на одном устройстве (пометка письма как прочтенное, перемещение писем между папками, удаление и т. п.) отобразятся на остальных. Если вы планируете наряду с почтовым клиентом использовать и веб-интерфейс, почтовый клиент надо также настраивать через IMAP.

В свою очередь графическая часть редактора будет использовать ресурсы Windows Forms (интегрированная среда разработки приложений на языке C++, разработанная корпорацией Microsoft и поставляемая либо как часть комплекта Microsoft Visual Studio, либо отдельно в виде бесплатного функционально ограниченного комплекта Visual C++ Express Edition)

При проектировании применялся нисходящий метод разработки, так как основной смысл программы состоит в выполнении отправки писем на почтовые ящики, а уже только потом графическое оформление и структурирование.

С начала реализовывались базовые функции, такие как, отправка писем и чтение данных о пользователе. После были разработаны, основываясь на базисе, функции, взаимодействующие с графическим интерфейсом, такие как, заполнение таблицы писем, отображение содержимого письма в отдельном окне.

4 Отладка и тестирование

В качестве среды разработки была выбрана программа Microsoft Visual Studio 2017. Программа предоставляет все средства, необходимые при разработке и отладке модулей и программ. В ходе отладки применялись такие возможности, как точка останова, выполнение кода по шагам, анализ содержимого локальных и глобальных переменных, анализ содержимого памяти.

Тестирование проводилось в рабочем порядке, в процессе разработки, после завершения написания программы. В ходе тестирования был добавлен вывод уведомления в случае неверного ввода.

5 Описание программы

Для реализации графической составляющей была задействована стандартная библиотека Windows Forms. Используя функции из этой библиотеки, осуществляется вывод данных на экран, получение событий клавиатуры и мыши, перемещения курсора. С помощью всех перечисленных возможностей этой библиотеки были реализованы меню и список сообщений.

Описание всех функций, которые не относятся к системным событиям (например, нажатие кнопок) приведены в таблице 1.

Таблица 1 - Спецификация функций.

Наименование функции	Описание
<code>Connections.openNewImapClient</code>	Создание нового ИМАР-клиента
<code>Connections.selectInbox</code>	Выбор новой папки для почтового ящика
<code>Connections.loadAllMessages</code>	Загрузка всех писем текущей папки
<code>Connections.loadMessagesWithOffset</code>	Загрузка определённого числа писем
<code>Connections.deleteMail</code>	Удаление письма по индексу
<code>Connections.getMessageByIndex</code>	Получить письмо по индексу
<code>Connections.getMessagesCount</code>	Получить количество писем в текущей папке
<code>UserData.getEmailAddress</code>	Получить текущий адрес пользователя
<code>UserData.getEmailPassword</code>	Получить текущий пароль пользователя
<code>UserData.getInstance</code>	Синглтон
<code>UserData.checkIsGmailEmail</code>	Проверка на принадлежность текущего почтового ящика к сервису Gmail

В данной таблице не приведены функции обработки системных событий, так как пояснение их принципа работы невозможно без контекста. Также все приведенные методы являются статическими, как и сами классы `Connections` и `UserData`.

Эти два класса не нуждаются в наследовании или инициализации, а данные, которые они предоставляют, могут потребоваться из любого участка программы.

Реализацией каждого логического модуля программы занимается отдельный класс:

- окно аутентификации обрабатывается классом `LoginInWindow`, в котором существуют методы и поля для хранения введенных данных необходимых для получения данных о пользователе и его списка писем;
- основной элемент интерфейса, в котором пользователю чаще всего приходится работать обрабатывается классом `LoginedWindow`, в котором также присутствуют специальные методы для обработки системных сообщений;
- окно для чтения писем и окно для написание нового письма обрабатываются классами `ReadMailForm` и `NewMailForm` соответственно. Их структура и внешний вид очень похожи, так как они работают практически с одной и той же формой, и набором данных.
- оставшиеся классы – `Connections` и `UserData` обеспечивают основной функционал программы, они отвечают за хранение данных о пользователе и занимаются отправкой и получением писем для определенных введенных данных.

Общую диаграммы классов можно представить следующим образом
Рисунок 2.

Большая часть программы обрабатывает системные события, такие как нажатие клавиш, нажатие кнопок и прочее. Поэтому главным фрагментом программы можно считать этап отправки письма, его схема представлена на рисунке 3.

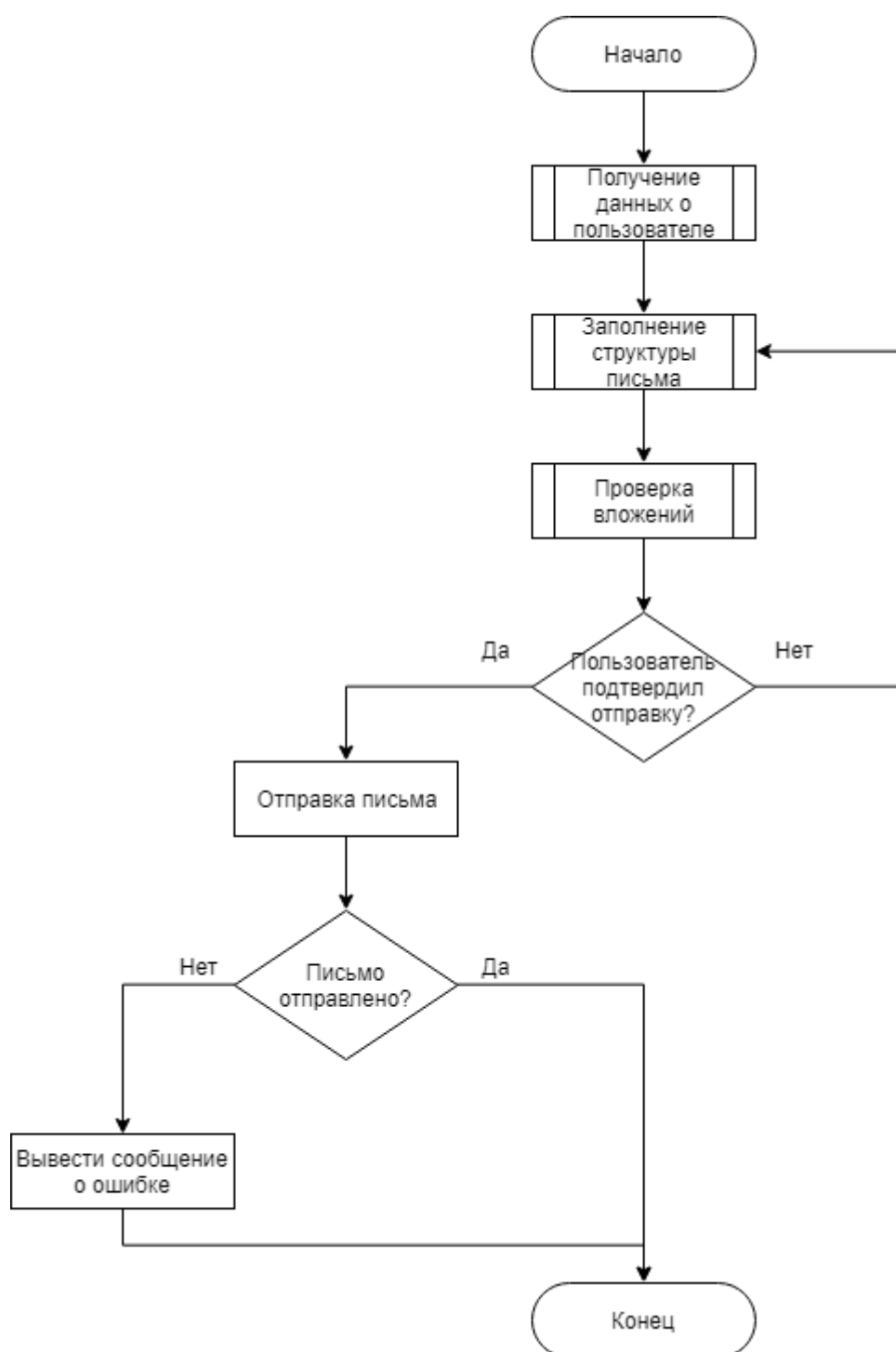


Рисунок 3 – Схема участка программы (Отправка письма)

Исходя из разбиения программы на логические модули была сформирована следующая иерархия вызовов, расположенная на рисунке 4.

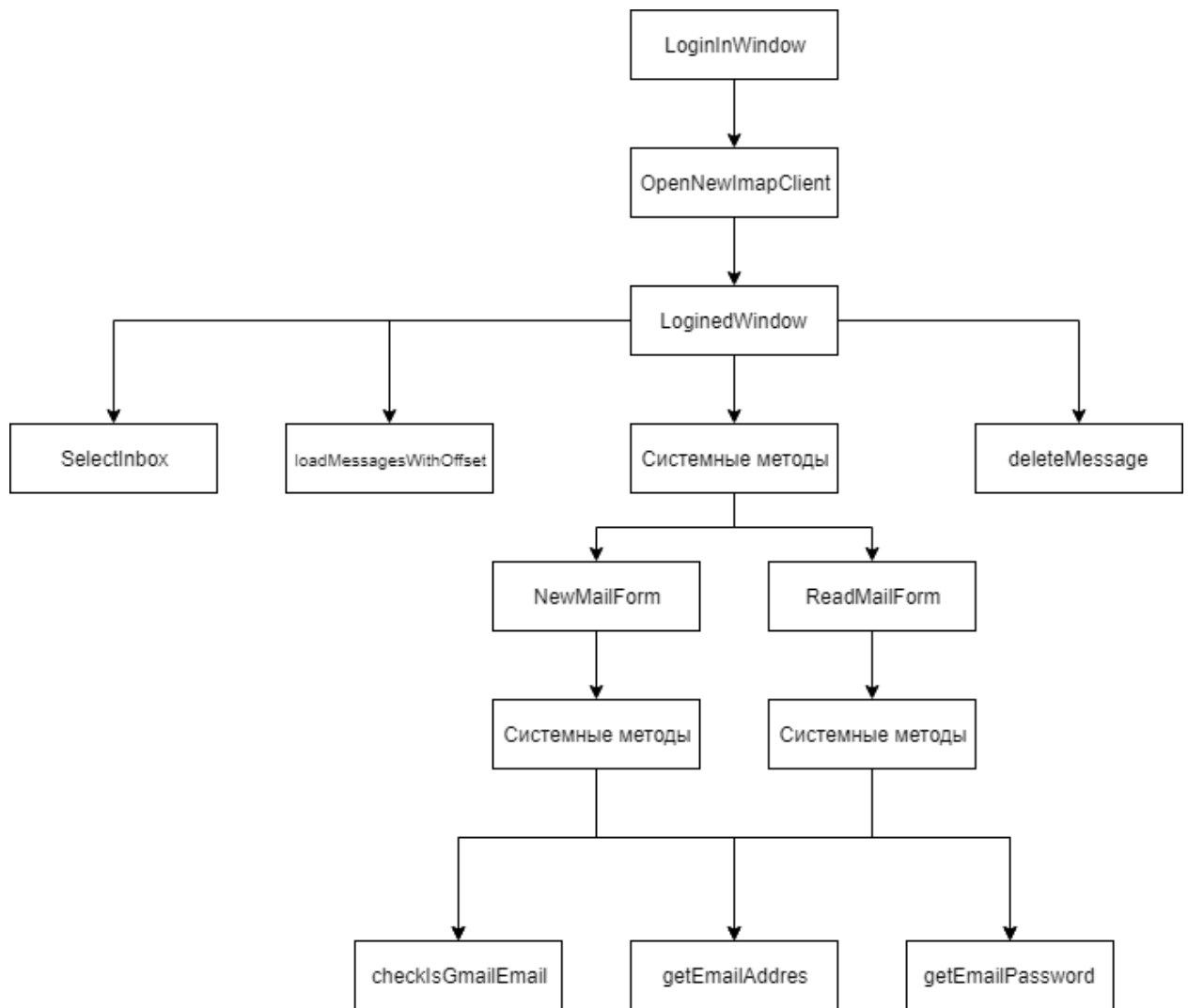


Рисунок 4 – Схема иерархии вызовов

6 Руководство пользователя

При входе в программу перед пользователем будет открыто окно аутентификации, в котором ему необходимо ввести свои данные, необходимые для входа в почту, такие как логин и пароль.

Также у пользователя есть возможность сохранить данные, позже, при следующем входе, поля будут заполнены автоматические рисунок 5

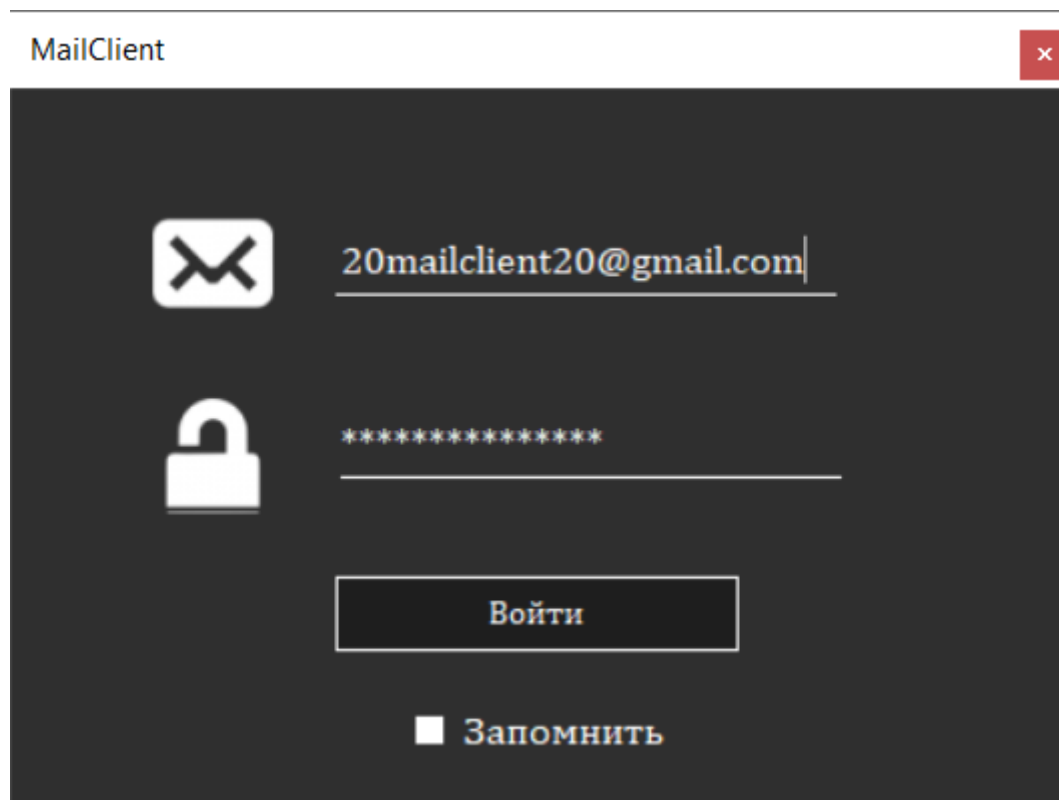


Рисунок 5 - Окно аутентификации

После входа в программу перед пользователем будет открыто следующее окно, в основе которого лежит список с входящими письмами рисунок 6.

Пользователь может просматривать сообщения на текущей странице (в каждой странице 30 сообщений), переходить к следующей странице, переходить к предыдущей странице, переходить к последней.

Общее число сообщений в текущей папки почты указано справ сверху окна, там же указана нумерация отображаемых сообщений в данный момент времени.

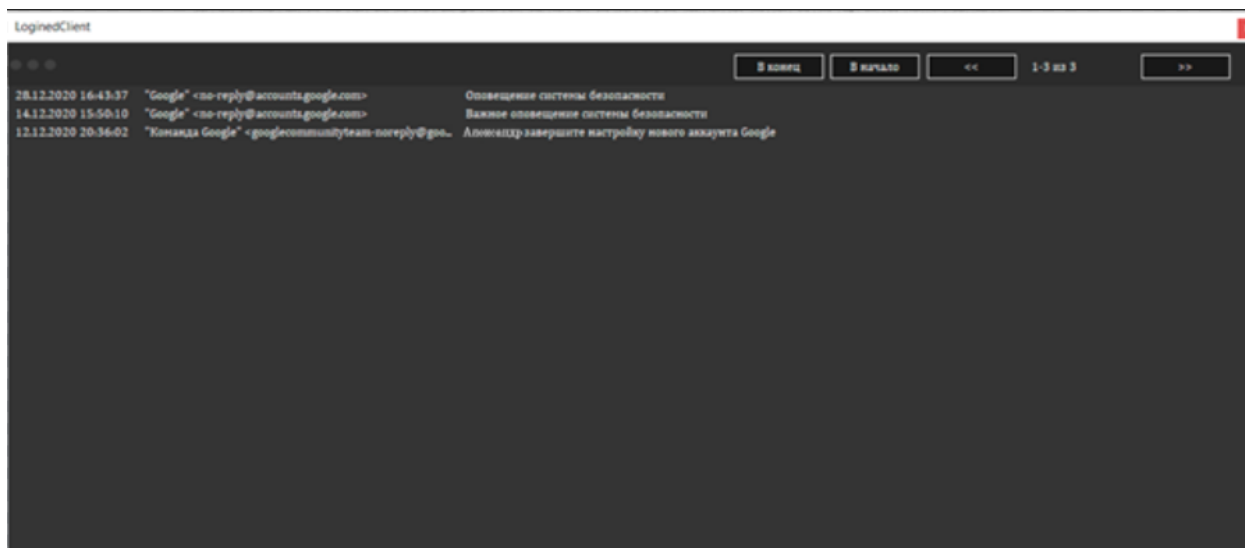


Рисунок 6 - Главное окно программы

Для того чтобы прочитать сообщение пользователю необходимо дважды нажать по выбранной строчке в списке, при наведении на строчку содержимое выбранного письма будет отображаться в подсказке.

После перед пользователем будет отображено окно с содержимым письма, так как современные почтовые сервисы в основном передают не просто текст сообщения, а упорядоченную html-структуру, была предусмотрена возможность отображение текста сообщения в браузере рисунок 7. Для смены режима пользователю необходимо нажать соответствующую клавишу в окне отображения письма.

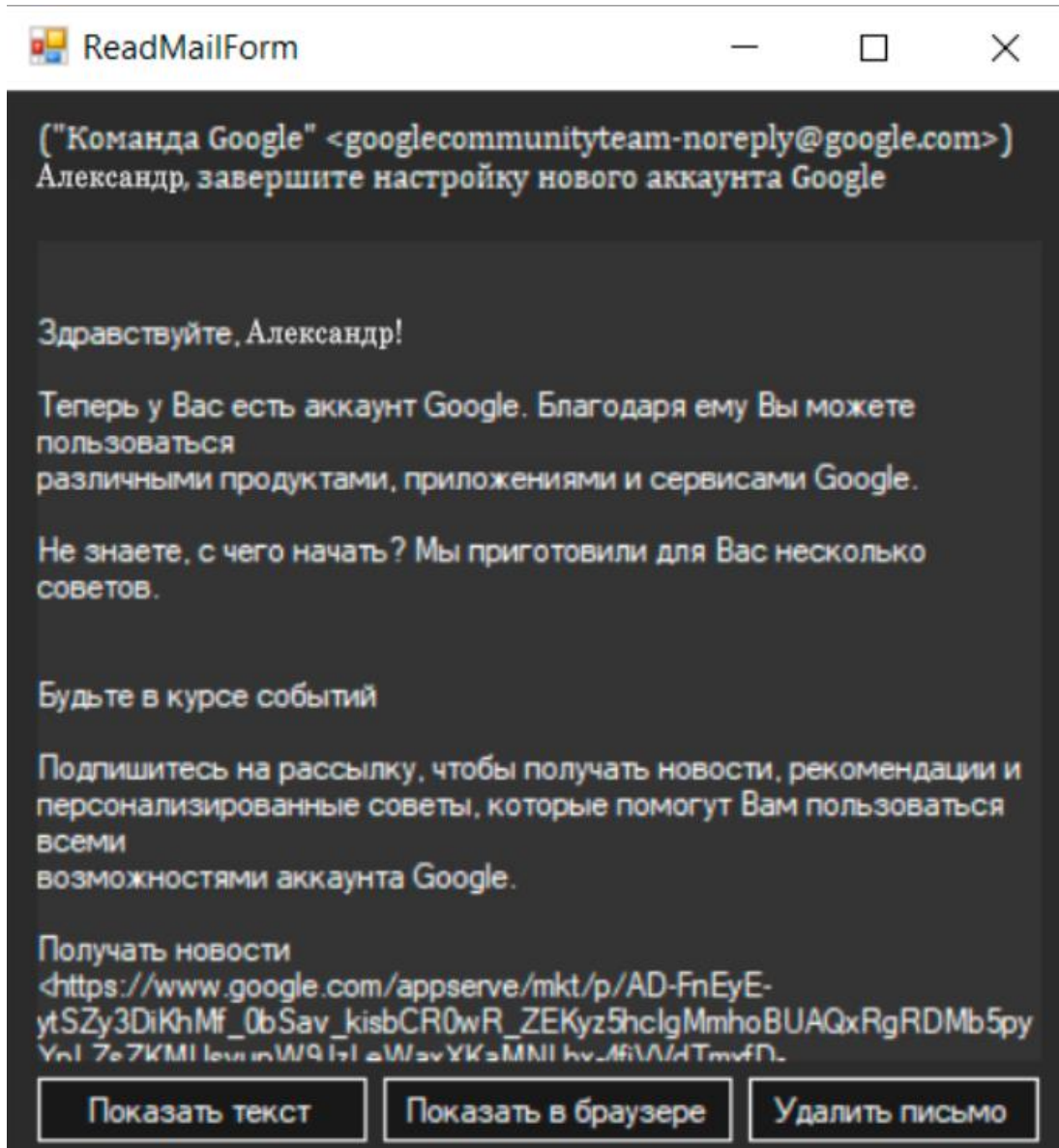


Рисунок 7 - Окно чтения писем

Также, при желании, пользователь может удалить текущее письмо, при нажатии на кнопку Удалить. Перед этим программа уточнит ввод пользователя для предотвращения случайных нажатий рисунок 8.

Текущее письмо удалиться не только в данном клиенте, но и на почте самого пользователя.

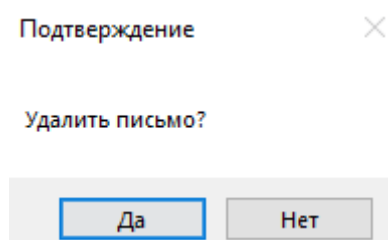


Рисунок 8 - Окно подтверждения удаления

Если пользователю необходимо отфильтровать сообщения по другим папкам (“Входящие”, “Отправленные”, “Важное”), то для этого ему необходимо вызвать дополнительное меню в боковой панели справа. для этого необходимо нажать на изображение троеточия в левом верхнем углу, те же действия нужно произвести для скрытия этого меню.

В дополнительном меню также предоставляется возможность обновить список сообщений и перейти к написанию нового письма рисунок 9.

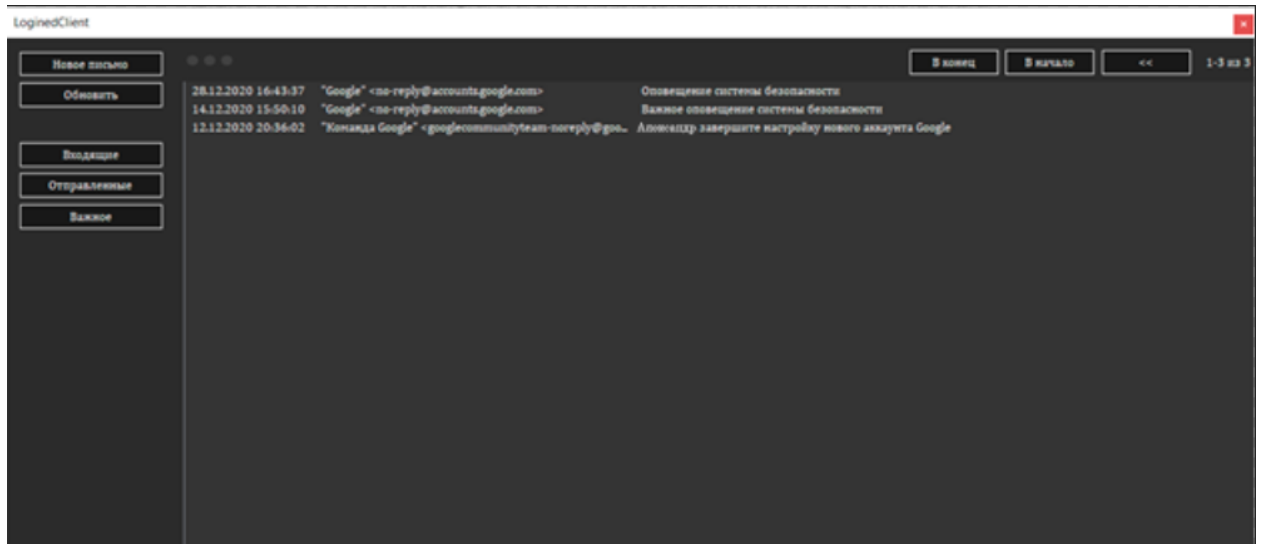


Рисунок 9 - Боковое меню

Для отправки письма пользователю необходимо заполнить все поля, требуемые для корректной работы SMTP протокола, такие как: адрес отправки, тема письма, отображаемое имя и сам текст письма.

У пользователя есть возможность вложить какой-либо файл, который будет отправлен, как вложение в письмо рисунок 10.

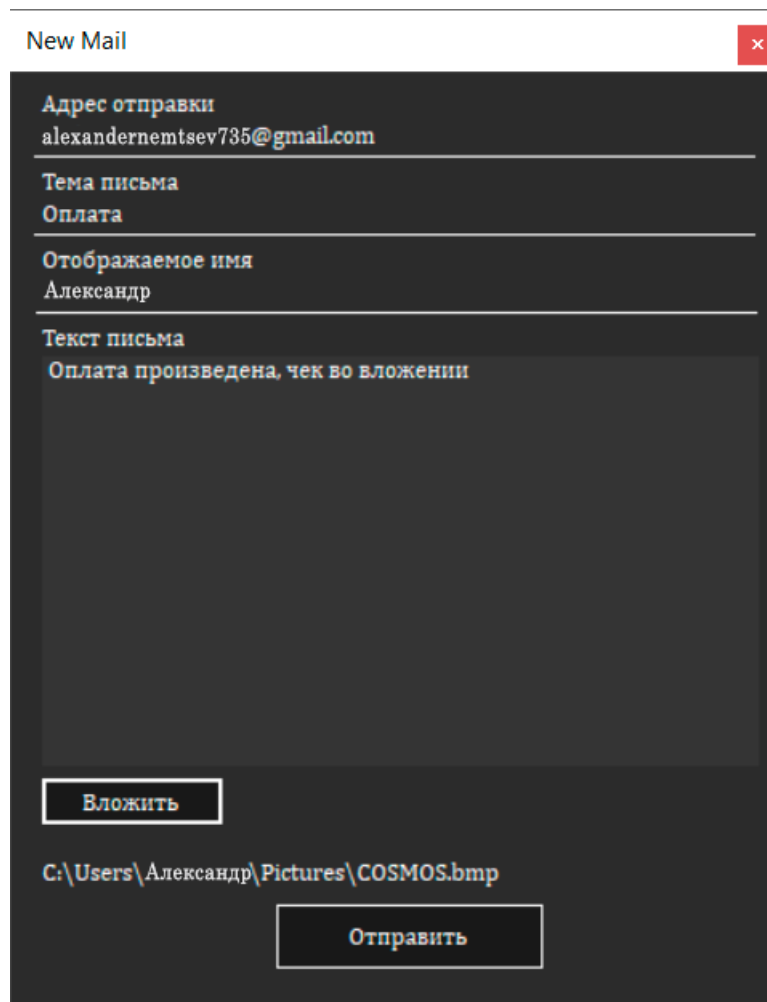


Рисунок 10 - Окно написания нового письма

Перед отправкой программа уточнит желание пользователя отправить письмо рисунок 11.

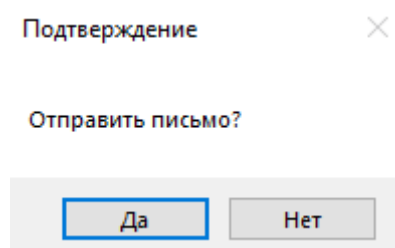


Рисунок 11- Окно подтверждения отправки

Отправленное таким образом письмо можно просмотреть в папке Отправленные рисунок 12.

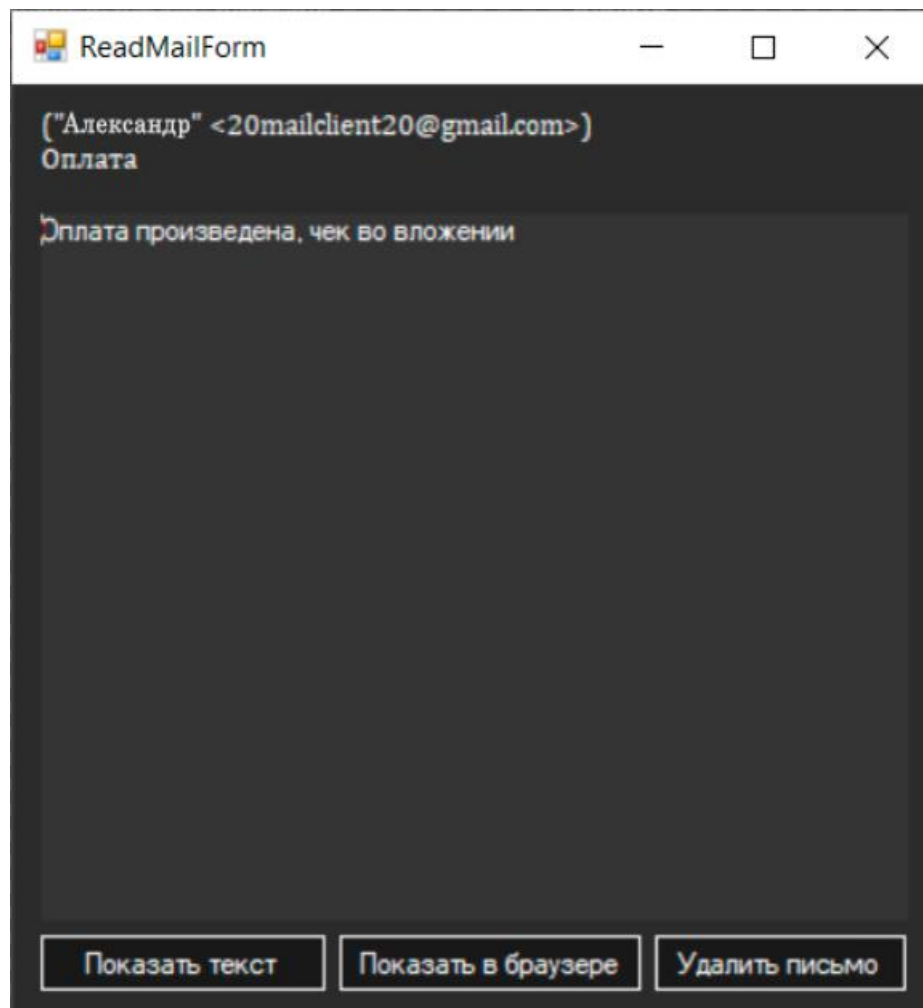


Рисунок 12 - Чтение отправленного письма

Ниже представлено отправленное письмо при просмотре из официальных почтовых клиентов рисунок 13 и 14.

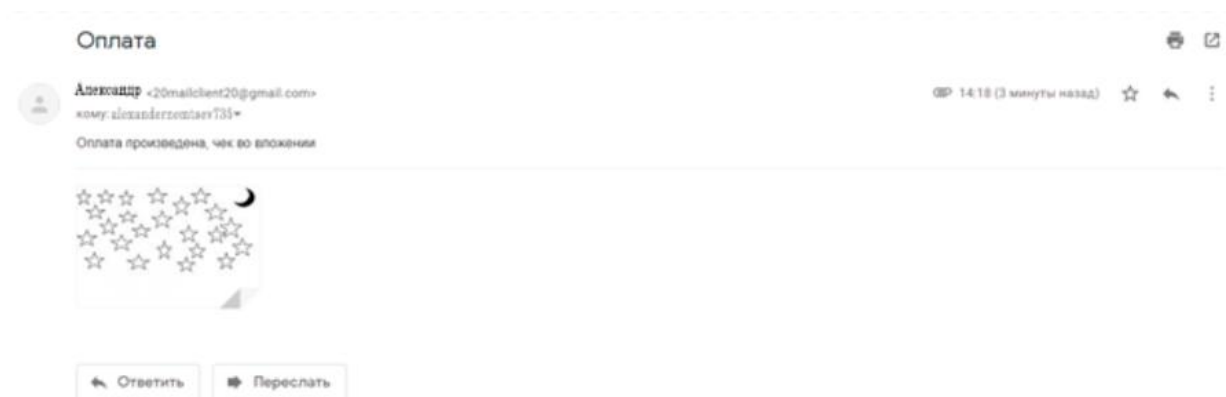


Рисунок 13 - Письмо на Gmail почте (отправитель)

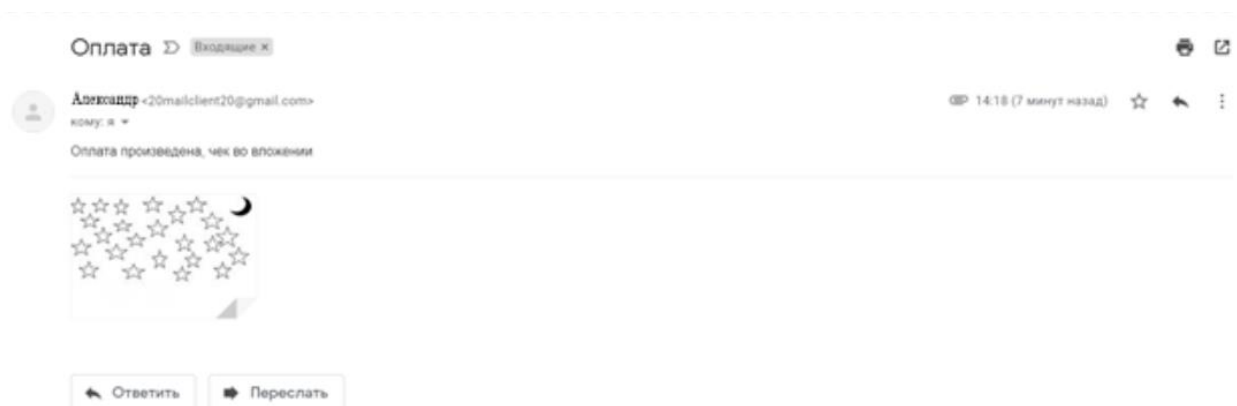


Рисунок 14 - Письмо на Mail почте (получатель)

Заключение

При выполнении данной курсовой работы были получены навыки разработки программ на языке высокого уровня. Были освоены приемы создания графического интерфейса. Усвоены механизмы реализации меню. Изучены функции работы с файлами. Получены базовые навыки программирования на языке C#. Изучены основные возможности среды программирования Visual Studio 2017. Получены навыки отладки и тестирования программ.

В рамках выполнения курсовой работы был написан почтовый клиент, основанный на обработке данных, считываемых по протоколам передачи электронных писем, и редактор для соответствующих писем.

Программу можно улучшить, оптимизировав код для более рационального использования графических ресурсов.

Список используемых источников

1. Windows Forms Develop. Авторы: Henrik Vogelius Hansson, Johann Haller 2013 г.
2. Основы программирования на С# 2009 г. Автор: Стенли Липпман.
3. Программирование на платформе Microsoft .NET Framework на языке С#. 4-е изд. Автор: Дж. Рихтер. 2011.
4. Изучаем С#. 3-е изд. Авторы: Э. Стиллмен, Дж. Грин. 2010 г.

Приложение А

Листинг модуля MailClient.exe

Приложение А.1

Файл «LoginInWindow.cs»

```
using AE.Net.Mail;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Mailclient
{
    public partial class MainWindow : Form
    {
        protected BinaryWriter bw;
        protected BinaryReader br;
        public MainWindow()
        {
            InitializeComponent();
            EmailPasswordTxt.Text = "";
            try
            {
                br = new BinaryReader(new FileStream("data.bin", FileMode.Open));
            }
            catch (IOException e){
                return;
            }
            try
            {
                EmailAddressTxt.Text = br.ReadString();
                EmailPasswordTxt.Text = br.ReadString();
            }
            catch (IOException e)
            {
                return;
            }
        }

        private void LoginInBtn_Click(object sender, EventArgs e)
        {
            UserData.GetInstance(EmailAddressTxt.Text, EmailPasswordTxt.Text);
            if (Connections.openNewImapClient())
            {
                Connections.loadMessagesWithOffset();
                LoggedWindow logged = new LoggedWindow();

                if (RememberChk.Checked)
                {
                    bw = new BinaryWriter(new FileStream("data.bin", FileMode.Create));
                    bw.Write(EmailAddressTxt.Text);
                    bw.Write(EmailPasswordTxt.Text);
                    bw.Close();
                }
            }
        }
    }
}
```

```

    }

    this.Hide();
    logged.Show();
    logged.updateCurrentPageLb();
    logged.addMessageToList();
}
else
    ErrorLb.Visible = true;
}

private void MainWindow_Load(object sender, EventArgs e)
{
    this.BackColor = Color.FromArgb(47,47,47);
    EmailAddressTxt.BackColor = Color.FromArgb(47, 47, 47);
    EmailPasswordTxt.BackColor = Color.FromArgb(47, 47, 47);
    LoginInBtn.BackColor = Color.FromArgb(30, 30, 30);
}
}
}

```

Приложение A.2

Файл «NewMailForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net;
using System.IO;
using System.Net.Mail;
namespace Mailclient
{
    public partial class NewMailForm : Form
    {
        public static string AttachmentPath, AttachmentContent;

        public NewMailForm()
        {
            InitializeComponent();
        }
        private void SendNewMailButton_Click(object sender, EventArgs e)
        {
            SendingMailResult.Text = "";
            NetConsoleApp.MailSMTPClient mailsSMTPClient = new
NetConsoleApp.MailSMTPClient();
            string message = "Отправить письмо?";
            string caption = "Подтверждение";
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;
            DialogResult result;

            result = MessageBox.Show(message, caption, buttons);
            if (result == System.Windows.Forms.DialogResult.Yes)
            {
                if (NetConsoleApp.MailSMTPClient.SendNewMail(DisplayedName.Text,
                    NewMailAddress.Text,
                    NewMailSubject.Text,

```

```

        NewMailText.Text))
    {
        SendingMailResult.Text = "Письмо отправлено";
    }
    else
    {
        SendingMailResult.Text = "Письмо не отправлено";
    }
}

}

private void NewMailForm_Load(object sender, EventArgs e)
{
    NewMailAddress.BackColor = Color.FromArgb(47, 47, 47);
    NewMailSubject.BackColor = Color.FromArgb(47, 47, 47);
    NewMailText.BackColor = Color.FromArgb(55, 55, 55);
    DisplayedName.BackColor = Color.FromArgb(47, 47, 47);
    SendNewMailButton.BackColor = Color.FromArgb(30, 30, 30);
    AddFile.BackColor = Color.FromArgb(30, 30, 30);
    panel1.BackColor = Color.FromArgb(47, 47, 47);
    this.BackColor = Color.FromArgb(47, 47, 47);
}

private void AddFile_Click(object sender, EventArgs e)
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
    {
        openFileDialog.InitialDirectory = "c:\\";
        openFileDialog.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*";
        openFileDialog.FilterIndex = 2;
        openFileDialog.RestoreDirectory = true;

        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            AttachmentPath = openFileDialog.FileName;

            var fileStream = openFileDialog.OpenFile();

            using (StreamReader reader = new StreamReader(fileStream))
            {
                AttachmentContent = reader.ReadToEnd();
            }
        }
        FilePathLabel.Text = AttachmentPath;
    }
}

}

namespace NetConsoleApp
{
    class MailSMTPClient
    {
        public static bool SendNewMail(string DisplayName,
            string AddressTo,
            string Subject,
            string Body) {
            if (String.IsNullOrEmpty(DisplayedName) ||
                String.IsNullOrEmpty(AddressTo) ||
                String.IsNullOrEmpty(Subject) ||
                String.IsNullOrEmpty(Body)) {
                return false;
            }
            MailAddress from = new MailAddress(Mailclient.UserData.getEmailAddress(),
                DisplayName);
            MailAddress to = new MailAddress(AddressTo);
            MailMessage m = new MailMessage(from, to);
            m.Subject = Subject;

```

```

        m.Body = Body;
        m.IsBodyHtml = true;
        SmtplibClient smtp = new SmtplibClient("smtp.gmail.com", 587);
        smtp.Credentials = new
NetworkCredential(Mailclient.UserData.getEmailAddress(),
Mailclient.UserData.getEmailPassword());
        smtp.EnableSsl = true;
        if(!String.IsNullOrEmpty(Mailclient.NewMailForm.AttachmentPath))
            m.Attachments.Add(new Attachment(Mailclient.NewMailForm.AttachmentPath));

        smtp.Send(m);
        return true;
    }
}
}

```

Приложение А.3 Файл «ReadMailForm.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Mailclient
{
    public partial class ReadMailForm : Form
    {
        AE.Net.Mail.MailMessage mCurrentMail;
        public void DisplayMail(int IndexInArray) {
            mCurrentMail = Connections.getMessageByIndex(IndexInArray);
            AddressLabel.Text = mCurrentMail.Sender + "(" + mCurrentMail.From + ") \n"
+mCurrentMail.Subject;
            MailBody.Text = mCurrentMail.Body;
            WebBrowser.DocumentText = mCurrentMail.Body;
        }

        public ReadMailForm()
        {
            InitializeComponent();
        }

        private void DisplayTextBtn_Click(object sender, EventArgs e){
            MailBody.Visible = true;
            WebBrowser.Visible = false;
        }

        private void DisplayHTMLBtn_Click(object sender, EventArgs e){
            MailBody.Visible = false;
            WebBrowser.Visible = true;
        }

        private void DeleteMailBtn_Click(object sender, EventArgs e){
            NetConsoleApp.MailSMTPClient mailSMTPClient = new
NetConsoleApp.MailSMTPClient();
            string message = "Удалить письмо?";
            string caption = "Подтверждение";
            MessageBoxButtons buttons = MessageBoxButtons.YesNo;

```

```

        DialogResult result;

        result = MessageBox.Show(message, caption, buttons);
        if (result == System.Windows.Forms.DialogResult.Yes)
        {
            Connections.deleteMail(mCurrentMail);
            this.Close();
        }
    }
    private void ReadMailForm_Load(object sender, EventArgs e)
    {
        panel1.BackColor = Color.FromArgb(47, 47, 47);
        this.BackColor = Color.FromArgb(47, 47, 47);
        DisplayHTMLBtn.BackColor = Color.FromArgb(30, 30, 30);
        DisplayTextBtn.BackColor = Color.FromArgb(30, 30, 30);
        DeleteMailBtn.BackColor = Color.FromArgb(30, 30, 30);
        MailBody.BackColor = Color.FromArgb(55, 55, 55);
        WebBrowser.BackColor = Color.FromArgb(47, 47, 47);
    }
}
}

```

Приложение A.4

Файл «Userdata.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Mailclient
{
    public partial class ReadMailForm : Form
    {
        AE.Net.Mail.MailMessage mCurrentMail;
        public void DisplayMail(int IndexInArray) {
            mCurrentMail = Connections.getMessageByIndex(IndexInArray);
            AddressLabel.Text = mCurrentMail.Sender + "(" + mCurrentMail.From + ")\n"
+mCurrentMail.Subject;
            MailBody.Text = mCurrentMail.Body;
            WebBrowser.DocumentText = mCurrentMail.Body;
        }

        public ReadMailForm()
        {
            InitializeComponent();
        }

        private void DisplayTextBtn_Click(object sender, EventArgs e){
            MailBody.Visible = true;
            WebBrowser.Visible = false;
        }

        private void DisplayHTMLBtn_Click(object sender, EventArgs e){
            MailBody.Visible = false;

```

```

        WebBrowser.Visible = true;
    }

    private void DeleteMailBtn_Click(object sender, EventArgs e){
        NetConsoleApp.MailSMTPClient mailSMTPClient = new
NetConsoleApp.MailSMTPClient();
        string message = "Удалить письмо?";
        string caption = "Подтверждение";
        MessageBoxButtons buttons = MessageBoxButtons.YesNo;
        DialogResult result;

        result = MessageBox.Show(message, caption, buttons);
        if (result == System.Windows.Forms.DialogResult.Yes)
        {
            Connections.deleteMail(mCurrentMail);
            this.Close();
        }
    }
    private void ReadMailForm_Load(object sender, EventArgs e)
    {
        panel1.BackColor = Color.FromArgb(47, 47, 47);
        this.BackColor = Color.FromArgb(47, 47, 47);
        DisplayHTMLBtn.BackColor = Color.FromArgb(30, 30, 30);
        DisplayTextBtn.BackColor = Color.FromArgb(30, 30, 30);
        DeleteMailBtn.BackColor = Color.FromArgb(30, 30, 30);
        MailBody.BackColor = Color.FromArgb(55, 55, 55);
        WebBrowser.BackColor = Color.FromArgb(47, 47, 47);
    }
}
}
}

```

Приложение A.5

Файл «Connections.cs»

```

using AE.Net.Mail;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Mailclient
{
    class Connections
    {
        public static readonly int PAGE_OFFSET = 30;
        private static readonly string GMAIL_IMAP_SERVER = "imap.gmail.com";
        private static readonly string MAIL_IMAP_SERVER = "imap.mail.ru";

        public static readonly string SEND_GMAIL_MAILBOX = "[Gmail]/Отправленные";
        public static readonly string SPAM_GMAIL_MAILBOX = "[Gmail]/Важное";
        public static readonly string ALL_GMAIL_MAILBOX = "[Gmail]/Вся почта";
        public static readonly string RECYCLEBIN_GMAIL_MAILBOX = "[Gmail]/Помеченные";
        public static readonly string INBOX_MAILBOX = "INBOX";

        public static readonly string SEND_MAIL_MAILBOX = "[Mail]/Отправленные";
        public static readonly string SPAM_MAIL_MAILBOX = "[Mail]/Важное";
        public static readonly string ALL_MAIL_MAILBOX = "[Mail]/Вся почта";
        public static readonly string RECYCLEBIN_MAIL_MAILBOX = "[Mail]/Помеченные";

        private static ImapClient mImapClient;
    }
}

```

```

public static int mCurrentPage = 1;
public static MailMessage[] mMailMessages;

public static bool openNewImapClient() {
    try
    {
        if(UserData.checkIsGmailEmail())
            mImapClient = new ImapClient(GMAIL_IMAP_SERVER,
            UserData.getEmailAddress(),
            UserData.getEmailPassword(), AE.Net.Mail.AuthMethods.Login, 993,
            true);
        else
            mImapClient = new ImapClient(MAIL_IMAP_SERVER,
            UserData.getEmailAddress(),
            UserData.getEmailPassword(), AE.Net.Mail.AuthMethods.Login, 993,
            true);
        mImapClient.SelectMailbox(INBOX_MAILBOX);
        AE.Net.Mail.Imap.Mailbox[] mailBoxes =
        mImapClient.ListMailboxes(string.Empty, "*");
        foreach (var mailBox in mailBoxes)
        {
            var mailboxName = mailBox.Name;
        }
    }
    catch (Exception e)
    {
        return false;
    }
    return true;
}

public static void selectInbox(string pInbox) {
    mImapClient.SelectMailbox(pInbox);
    loadMessagesWithOffset();
}

public static void loadAllMessages() {
    mMailMessages = mImapClient.GetMessages(mImapClient.GetMessageCount(), 0);
}

public static void loadMessagesWithOffset(){
    mMailMessages = mImapClient.GetMessages(mImapClient.GetMessageCount() -
    (PAGE_OFFSET * (mCurrentPage-1)),
    mImapClient.GetMessageCount() - (PAGE_OFFSET*mCurrentPage));
}

public static void deleteMail(AE.Net.Mail.MailMessage m) {
    mImapClient.DeleteMessage(m);
}

public static MailMessage getMessageByIndex(int Index) {
    if (Index < 0)
        Index = 0;
    return mImapClient.GetMessages(mImapClient.GetMessageCount() - Index,
    mImapClient.GetMessageCount() - Index - 1, false).ToList()[0];
}

public static int getMassagesCount() {
    return mImapClient.GetMessageCount();
}
}
}

```

Приложение А.6

Файл «LoginedWindow.cs»

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Mailclient
{
    public partial class LoggedWindow : Form
    {
        public LoggedWindow()
        {
            InitializeComponent();

            private int mSliderWidth = 175;
            private bool isHidenSlider = true;
            public void addMessageToList() {
                MessagesListView.Items.Clear();
                foreach (var mess in Connections.mMailMessages) {
                    ListViewItem item = new ListViewItem(mess.Date.ToString());
                    if (mess.From != null)
                        item.SubItems.Add(mess.From.ToString());
                    else
                        item.SubItems.Add("");
                    if (mess.From != null)
                        item.SubItems.Add(mess.Subject.ToString());
                    else
                        item.SubItems.Add("");
                    /*item.SubItems.Add(mess.Body.ToString());*/
                    MessagesListView.Items.Insert(0, item);
                }
            }
            private void SlideTimer_Tick(object sender, EventArgs e)
            {
                if (isHidenSlider)
                {
                    SlideMenu.Width += 25;
                    if (SlideMenu.Width >= mSliderWidth)
                    {
                        SlideTimer.Stop();
                        isHidenSlider = false;
                        this.Refresh();
                    }
                }
                else {
                    SlideMenu.Width -= 25;
                    if (SlideMenu.Width <= 0)
                    {
                        SlideTimer.Stop();
                        isHidenSlider = true;
                        this.Refresh();
                    }
                }
            }
            private void NewMailButton_Click(object sender, EventArgs e)
            {
                NewMailForm MailForm = new NewMailForm();
                MailForm.Show();
            }
            private void PictureBox1_Click(object sender, EventArgs e)
            {
                SlideTimer.Start();
            }
        }
    }
}

```

```

        private void MessagesListView_ItemSelectionChanged(object sender,
ListViewItemSelectionChangedEventArgs e)
        {
            if (MessagesListView.SelectedItems.Count > 0)
            {
                ReadMailForm readMailForm = new ReadMailForm();

readMailForm.DisplayMail(MessagesListView.Items.IndexOf(MessagesListView.SelectedItems[0]
)+
                (Connections.PAGE_OFFSET*(Connections.mCurrentPage-1)));
                readMailForm.TopMost = true;
                readMailForm.Show();
                MessagesListView.SelectedItems.Clear();
            }
        }
        public void updateCurrentPageLb() {
            CurrentPageLb.Text = "" + ((Connections.mCurrentPage - 1) *
(Connections.PAGE_OFFSET)+1) + "-" +
                ((Connections.mCurrentPage * Connections.PAGE_OFFSET)+1) + "
из " + Connections.getMessagesCount();
        }
        private void NextPageBtn_Click(object sender, EventArgs e){
            if (Connections.mCurrentPage != (int)Connections.getMessagesCount() /
Connections.PAGE_OFFSET){
                Connections.mCurrentPage++;
                RefreshBtn_Click(null, null);
            }
            else
                return;
        }
        private void PrevPageBtn_Click(object sender, EventArgs e){
            if (Connections.mCurrentPage != 1){
                Connections.mCurrentPage--;

                RefreshBtn_Click(null, null);
            }
            else
                return;
        }

        private void RefreshBtn_Click(object sender, EventArgs e){
            updateCurrentPageLb();
            Connections.loadMessagesWithOffset();
            addMessageToList();
        }

        private void ToBeginBtn_Click(object sender, EventArgs e){
            Connections.mCurrentPage = 1;
            RefreshBtn_Click(null, null);
        }

        private void ToEndBtn_Click(object sender, EventArgs e){
            Connections.mCurrentPage = (int) Connections.getMessagesCount() /
Connections.PAGE_OFFSET;
            RefreshBtn_Click(null, null);
        }

        private void LoggedInWindow_FormClosing(object sender, FormClosingEventArgs e){
            Application.Exit();
        }

        private void MessagesListView_ColumnWidthChanging(object sender,
ColumnWidthChangingEventArgs e)
        {
            e.Cancel = true;
            e.NewWidth = MessagesListView.Columns[e.ColumnIndex].Width;
        }

        private void InboxBtn_Click(object sender, EventArgs e)

```

```

    {
        Connections.selectInbox(Connections.INBOX_MAILBOX);
        RefreshBtn_Click(null, null);
    }

    private void SendBtn_Click(object sender, EventArgs e)
    {
        Connections.selectInbox(Connections.SEND_GMAIL_MAILBOX);
        RefreshBtn_Click(null, null);
    }

    private void SpamBtn_Click(object sender, EventArgs e)
    {
        Connections.selectInbox(Connections.SPAM_GMAIL_MAILBOX);
        RefreshBtn_Click(null, null);
    }

    private void RecyclebinBtn_Click(object sender, EventArgs e)
    {
        Connections.selectInbox(Connections.RECYCLEBIN_GMAIL_MAILBOX);
        RefreshBtn_Click(null, null);
    }

    private void LoginedWindow_Load(object sender, EventArgs e)
    {
        SlideMenu.BackColor = Color.FromArgb(47, 47, 47);
        TopPnl.BackColor = Color.FromArgb(47, 47, 47);
        MessagesListView.BackColor = Color.FromArgb(55, 55, 55);

        InboxBtn.BackColor = Color.FromArgb(30, 30, 30);
        SendBtn.BackColor = Color.FromArgb(30, 30, 30);
        SpamBtn.BackColor = Color.FromArgb(30, 30, 30);
        NewMailButton.BackColor = Color.FromArgb(30, 30, 30);
        RefreshBtn.BackColor = Color.FromArgb(30, 30, 30);
        PrevPageBtn.BackColor = Color.FromArgb(30, 30, 30);
        NextPageBtn.BackColor = Color.FromArgb(30, 30, 30);
        ToEndBtn.BackColor = Color.FromArgb(30, 30, 30);
        ToBeginBtn.BackColor = Color.FromArgb(30, 30, 30);
    }
}
}

```

Приложение Б. UML-диаграммы

Приложение Б.1 UML-диаграмма вариантов использования

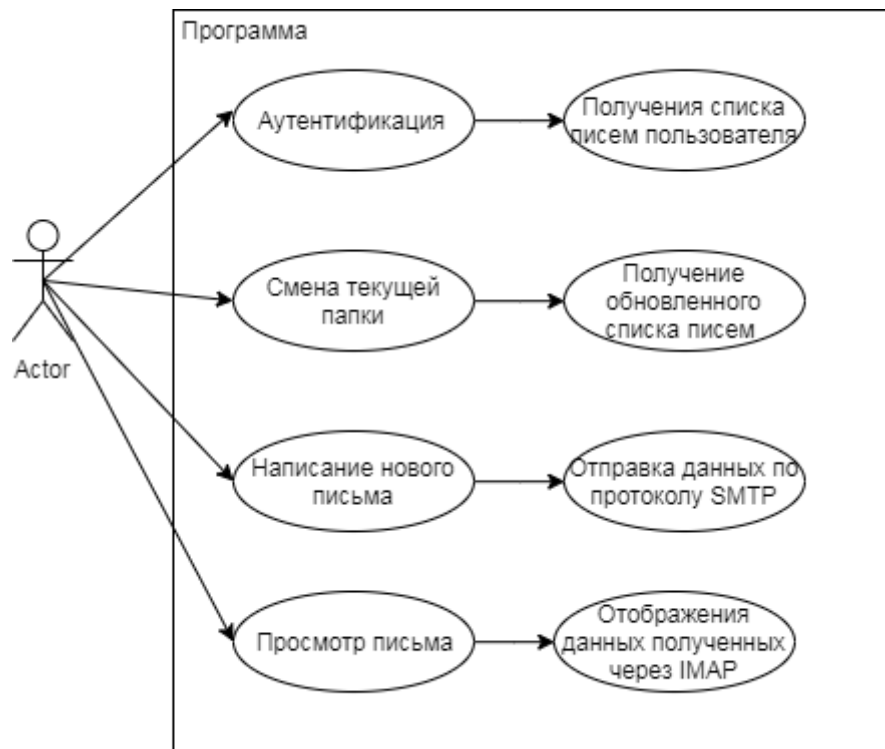


Рисунок 15 – UML – диаграмма вариантов использования

Приложение Б.2 UML-диаграмма состояний программы

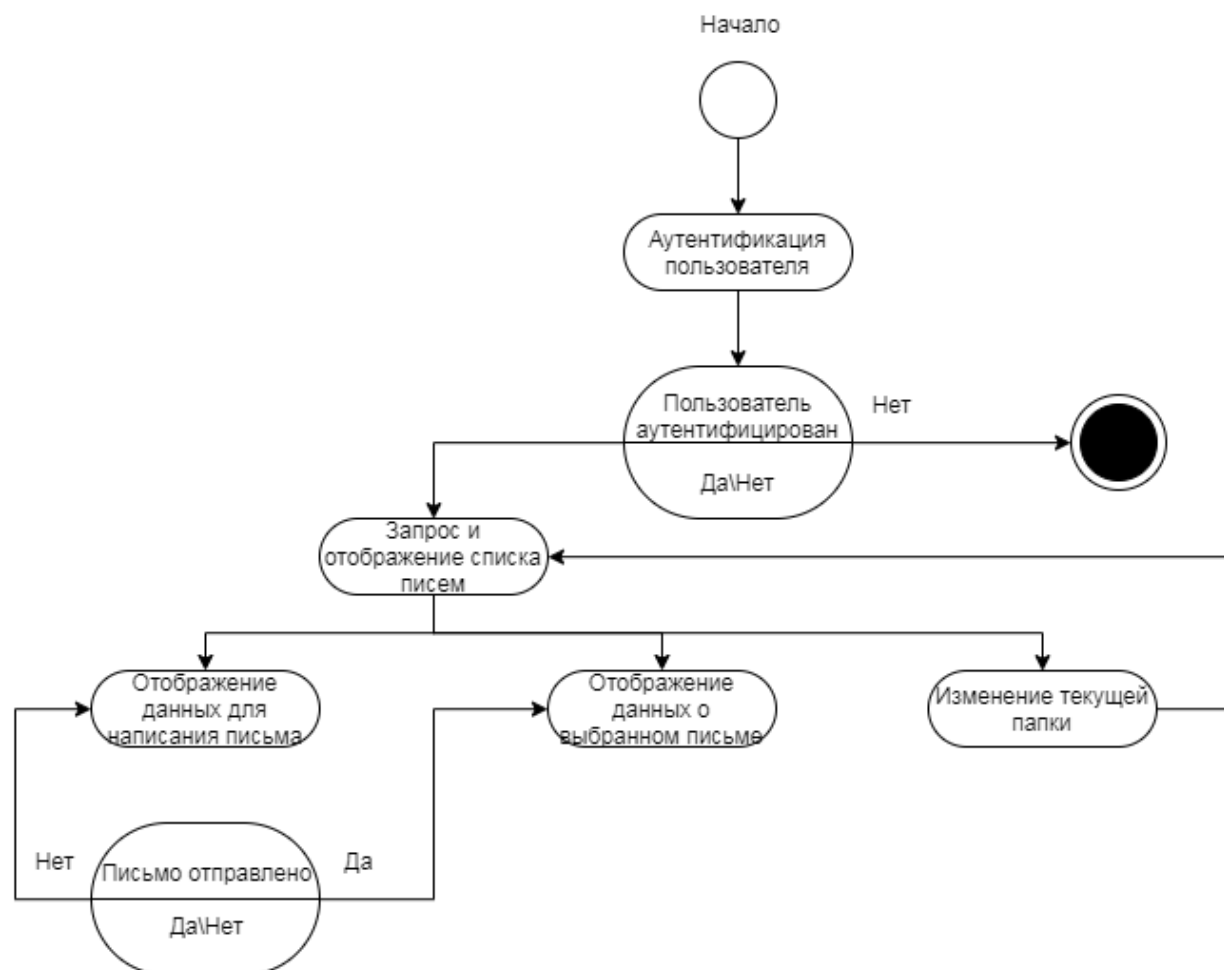


Рисунок 16 – UML – диаграмма состояний программы