МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Российский технологический университет - МИРЭА»

РТУ МИРЭА

Институт информационных технологий (ИИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3

по дисциплине

«Инструментальное ПО разработки и проектирования ИС»

на тему:

«Создание приложения для массовой отправки email писем со вложениями на языке Python»

Выполнил студент группы ИКМО-01-20 — Попов К.С.
Фамилия И.О.

Принял(а) — Куликов А.А.
Фамилия И.О.

Работы выполнены «__»_____2021 г. _____
(подпись студента)

«Зачтено» «__»_____2021 г. _____
(подпись руководителя)

Москва, 2021 г.

Оглавление

Цель практической работы

В данной работе требуется добавить в приложение по отправке email писем, разработанного на Python следующий функционал:

- Добавить на форму следующие элементы:
    - Кнопку для открытия диалогового окна файловой системы для выбора csv файла.
    - Метку для отображения названия выбранного файла.
- Добавить возможность массовой рассылки писем со вложениями.
- Реализовать чтение адресата и списка файлов из csv файла для отправки письма.

Ход выполнения работы

Для выполнения постеленных целей, была доработана форма отправки сообщений, в которую были добавлены элементы интерфейса. Форма в QtDesigner представлена на рисунке 1.

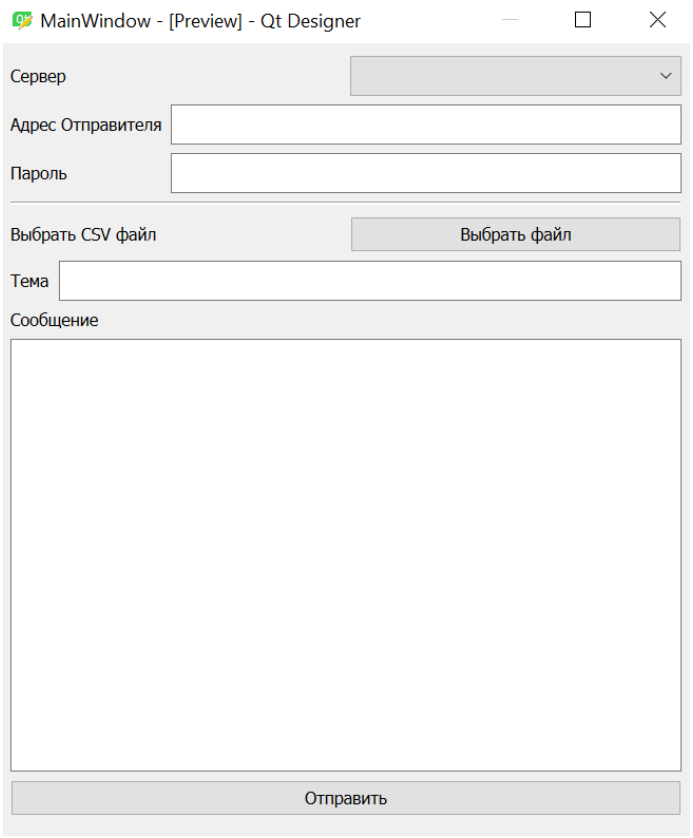

Рисунок 1 – Форма в QtDesigner

Выбор csv файла осуществляется через диалоговое окно, после чего выбранный файл отображается на форме.

```
def selectCSV(self):
    fileName = QtWidgets.QFileDialog.getOpenFileName(parent=self,
                                                     caption=u'Открыть файл',
                                                     directory='D:/',
                                                     filter='csv
(*.csv);;AllFiles (*)')[0]
    if fileName:  # Если выбран файл
        self.csvFile = fileName
        self.label_6.setText(os.path.basename(self.csvFile))
```

После чтения файла функцией csvReader, содержимое с адресатами и файлами помещается в словарь.

```
def csvReader(self):
    receiver = {}
    with open(self.csvFile, 'r') as csvFile:
        csvReader = csv.reader(csvFile)
        for row in csvReader:
            to, file = row[0].split(';')
            receiver[to] = [file]
        return receiver
```

И при отправке письма для каждого адресата формируется пул файлов для отправки в функции sendMail.

```
def sendMail(self):
    try:
        if self.SenderLineEdit.text().split('@')[1] !=
self.ServerComboBox.currentText():
            self.showMsgBox('Введенная почта и сервер не совпадают!')
        else:
            self.smtpServer =
smtplib.SMTP(self.servers[self.ServerComboBox.currentText()])  # Создание
объекта SMTP
            self.myEMAIL = self.SenderLineEdit.text()
            self.myPassword = self.PasswordLineEdit.text()
            self.smtpServer.starttls()  # Шифрование
            self.smtpServer.login(self.myEMAIL, self.myPassword)  # Ввод
учетных данных
            for k, v in self.csvReader().items():
                myMessage = MIMEMultipart()  # Создание объекта стандарта
MIME
                myMessage['Subject'] = self.SubjectLineEdit.text()  # Тема
сообщения
                myMessage['From'] = self.myEMAIL  # Отправитель
                myMessage['To'] = k  # Получатель
                messageContent = self.MessageEdit.toPlainText()  # Получение
текста с формы
                myMessage.attach(MIMEText(messageContent, 'plain'))  #
Добавление текста в сообещение
                for file in self.processAttachment(v):
                    myMessage.attach(file)
                self.smtpServer.send_message(myMessage)  # Отправка сообщения
    except:
        self.showMsgBox('Что то пошло не так')
```

Результат выполнения работы программы представлен на рисунках 2 и 3.

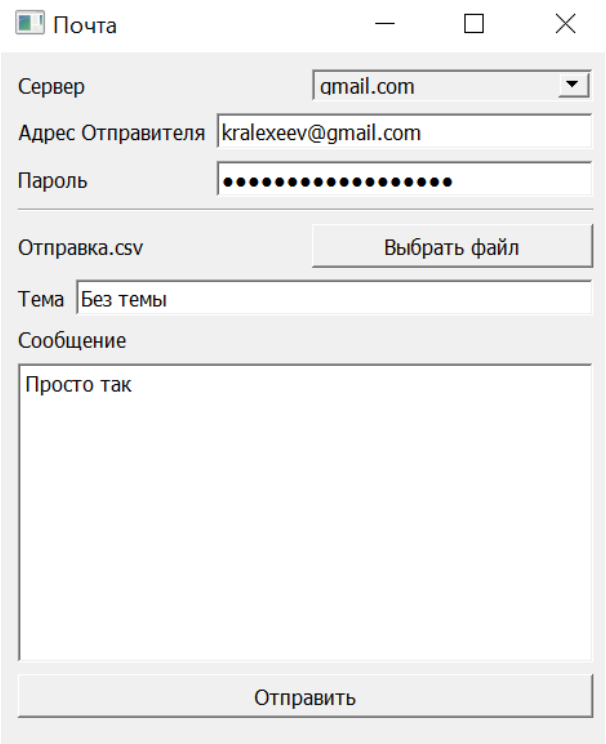

Рисунок 2 – Запущенное приложение

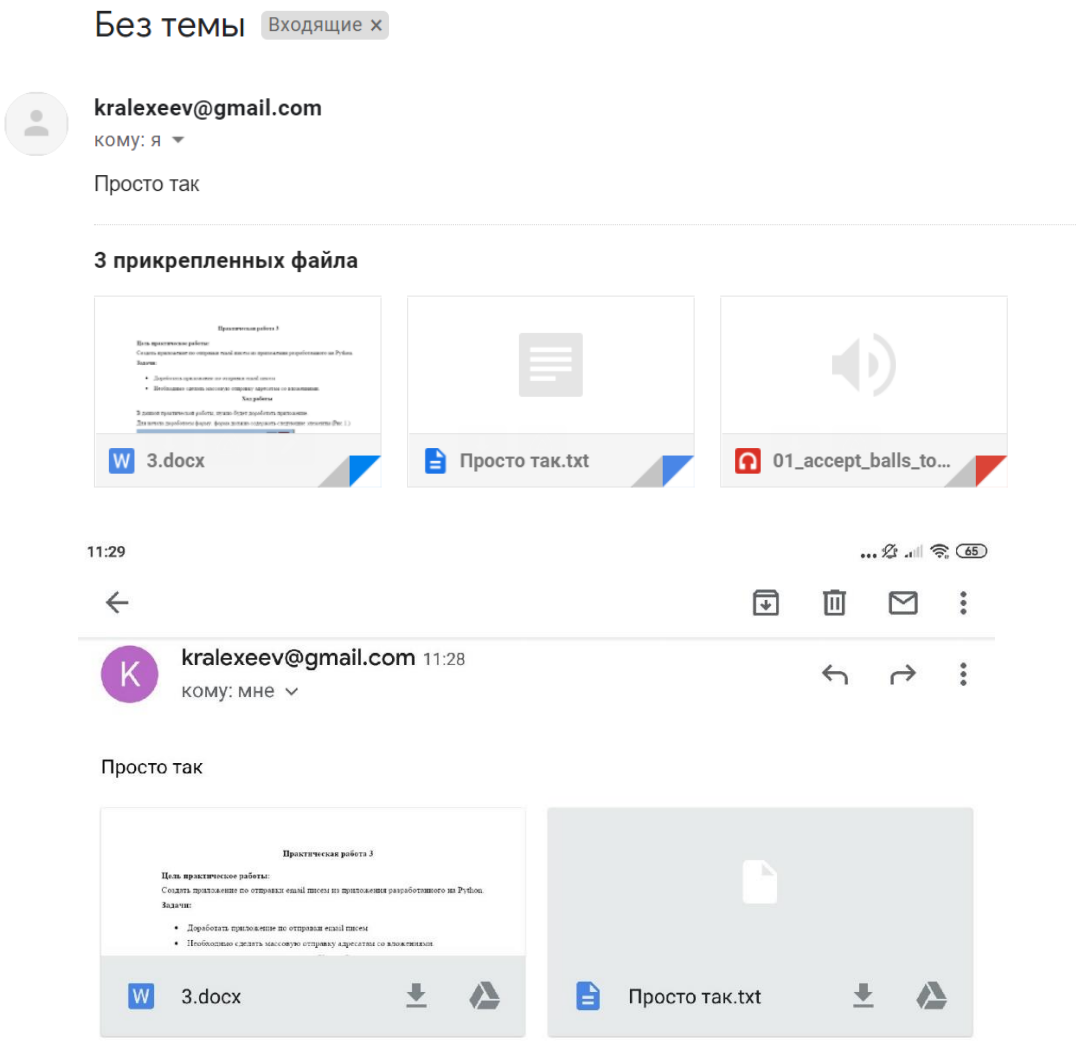

Рисунок 3 – Полученные письма

Содержимое csv файла представлено на рисунке 4.

Отправка.csv

```
1  kralexeev@gmail.com;D:/3.docx;D:/Просто так.txt;D:/01_accept_balls_to_the_wall.mp3
2  ilkir0208@gmail.com;D:/3.docx;D:/Просто так.txt;
3
```

Рисунок 4 – Содержимое файла «Отправка.csv»

Листинг программы

Модуль main:

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.audio import MIMEAudio
```

```
from email.mime.base import MIMEBase
import email.encoders as encoders
from PyQt5 import QtWidgets
import mailSenderUI
import sys
import os
import mimetypes
import csv


class MailAPP(QtWidgets.QMainWindow, QtWidgets.QFileDialog,
mailSenderUI.Ui_MainWindow):
    def __init__(self):
        super().__init__()
        self.setupUi(self)
        self.setWindowTitle('Почта')
        self.setFocus()
        self.myEMAIL = ''
        self.myPassword = ''
        self.PasswordLineEdit.setEchoMode(QtWidgets.QLineEdit.Password)
        services = ['gmail.com', 'mail.ru', 'yandex.ru']
        self.servers = {'gmail.com': 'smtp.gmail.com:587', 'mail.ru':
'smtp.yandex.ru:467',
                        'yandex.ru': 'smtp.mail.ru:25'}
        for service in services:
            self.ServerComboBox.addItem(service)
        self.SendPushButton.clicked.connect(self.sendMail)  # self.sendMail
        self.AddCSVFilePushButton.clicked.connect(self.selectCSV)
        self.msgBox = QtWidgets.QMessageBox()
        self.csvFile = ''

    def showMsgBox(self, msg: str):
        self.msgBox.setText(msg)
        self.msgBox.exec_()

    def selectCSV(self):
        fileName = QtWidgets.QFileDialog.getOpenFileName(parent=self,
                                                         caption=u'Открыть
файл',
                                                         directory='D:/',
                                                         filter='csv
(*.csv);;AllFiles (*)')[0]
        if fileName:  # Если выбран файл
            self.csvFile = fileName
            self.label_6.setText(os.path.basename(self.csvFile))

    def processAttachment(self, files: list):
        filesList = []
        for file in files:
            fileName = ''
            try:
                fileName = os.path.basename(file)
                ctype, encoding = mimetypes.guess_type(file)
                if ctype is None or encoding is not None:
                    ctype = 'application/octet-stream'
                maintype, subtype = ctype.split('/', 1)
                if maintype == 'text':
                    with open(file) as fp:
                        file = MIMEText(fp.read(), _subtype=subtype)
                        fp.close()
                elif maintype == 'image':
                    with open(file, 'rb') as fp:
                        file = MIMEImage(fp.read(), _subtype=subtype)
                        fp.close()
```

```
                elif maintype == 'audio':
                    with open(file, 'rb') as fp:
                        file = MIMEAudio(fp.read(), _subtype=subtype)
                        fp.close()
                else:
                    with open(file, 'rb') as fp:
                        file = MIMEBase(maintype, subtype)
                        file.set_payload(fp.read())
                        fp.close()
                        encoders.encode_base64(file)
                file.add_header('Content-Disposition', 'attachment',
filename=fileName)
                filesList.append(file)
            except:
                self.showMsgBox('Ошибка добавления файла: ' + fileName)
        return filesList

def csvReader(self):
    receiver = {}
    with open(self.csvFile, 'r') as csvFile:
        csvReader = csv.reader(csvFile)
        for row in csvReader:
            to = row[0].split(';')[0]
            file = row[0].split(';')[1:]
            receiver[to] = file
            print(receiver)
        return receiver



def sendMail(self):
    try:
        if self.SenderLineEdit.text().split('@')[1] !=
self.ServerComboBox.currentText():
            self.showMsgBox('Введенная почта и сервер не совпадают!')
        else:
            self.smtpServer =
smtplib.SMTP(self.servers[self.ServerComboBox.currentText()])  # Создание
объекта SMTP
            self.myEMAIL = self.SenderLineEdit.text()
            self.myPassword = self.PasswordLineEdit.text()
            self.smtpServer.starttls()  # Шифрование
            self.smtpServer.login(self.myEMAIL, self.myPassword)  # Ввод
учетных данных
            for k, v in self.csvReader().items():
                myMessage = MIMEMultipart()  # Создание объекта стандарта
MIME
                myMessage['Subject'] = self.SubjectLineEdit.text()  # Тема
сообщения
                myMessage['From'] = self.myEMAIL  # Отправитель
                myMessage['To'] = k  # Получатель
                messageContent = self.MessageEdit.toPlainText()  # Получение
текста с формы
                myMessage.attach(MIMEText(messageContent, 'plain'))  #
Добавление текста в сообщение
                for file in self.processAttachment(v):
                    myMessage.attach(file)
                self.smtpServer.send_message(myMessage)  # Отправка сообщения
    except:
        self.showMsgBox('Что то пошло не так')


def main():
    app = QtWidgets.QApplication(sys.argv)
```

```
    window = MailAPP()
    window.show()
    sys.exit(app.exec_())


if __name__ == '__main__':
    main()
```

Сконвертированный модуль Python формы mailSenderUI.

```
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'mailSenderUI.ui'
#
# Created by: PyQt5 UI code generator 5.15.0
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets


class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(330, 390)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.verticalLayout_5 = QtWidgets.QVBoxLayout(self.centralwidget)
        self.verticalLayout_5.setObjectName("verticalLayout_5")
        self.horizontalLayout_2 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_2.setObjectName("horizontalLayout_2")
        self.label_3 = QtWidgets.QLabel(self.centralwidget)
        self.label_3.setObjectName("label_3")
        self.horizontalLayout_2.addWidget(self.label_3)
        self.ServerComboBox = QtWidgets.QComboBox(self.centralwidget)
        self.ServerComboBox.setObjectName("ServerComboBox")
        self.horizontalLayout_2.addWidget(self.ServerComboBox)
        self.verticalLayout_5.addLayout(self.horizontalLayout_2)
        self.horizontalLayout = QtWidgets.QHBoxLayout()
        self.horizontalLayout.setObjectName("horizontalLayout")
        self.verticalLayout_4 = QtWidgets.QVBoxLayout()
        self.verticalLayout_4.setObjectName("verticalLayout_4")
        self.label_4 = QtWidgets.QLabel(self.centralwidget)
        self.label_4.setObjectName("label_4")
        self.verticalLayout_4.addWidget(self.label_4)
        self.label_5 = QtWidgets.QLabel(self.centralwidget)
        self.label_5.setObjectName("label_5")
        self.verticalLayout_4.addWidget(self.label_5)
        self.horizontalLayout.addLayout(self.verticalLayout_4)
        self.verticalLayout_3 = QtWidgets.QVBoxLayout()
        self.verticalLayout_3.setObjectName("verticalLayout_3")
        self.SenderLineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.SenderLineEdit.setObjectName("SenderLineEdit")
        self.verticalLayout_3.addWidget(self.SenderLineEdit)
        self.PasswordLineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.PasswordLineEdit.setObjectName("PasswordLineEdit")
        self.verticalLayout_3.addWidget(self.PasswordLineEdit)
        self.horizontalLayout.addLayout(self.verticalLayout_3)
        self.verticalLayout_5.addLayout(self.horizontalLayout)
        self.line = QtWidgets.QFrame(self.centralwidget)
        self.line.setFrameShape(QtWidgets.QFrame.HLine)
```

```
        self.line.setFrameShadow(QtWidgets.QFrame.Sunken)
        self.line.setObjectName("line")
        self.verticalLayout_5.addWidget(self.line)
        self.verticalLayout_2 = QtWidgets.QVBoxLayout()
        self.verticalLayout_2.setObjectName("verticalLayout_2")
        self.horizontalLayout_6 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_6.setObjectName("horizontalLayout_6")
        self.label_6 = QtWidgets.QLabel(self.centralwidget)
        self.label_6.setObjectName("label_6")
        self.horizontalLayout_6.addWidget(self.label_6)
        self.AddCSVFilePushButton = QtWidgets.QPushButton(self.centralwidget)
        self.AddCSVFilePushButton.setObjectName("AddCSVFilePushButton")
        self.horizontalLayout_6.addWidget(self.AddCSVFilePushButton)
        self.verticalLayout_2.addLayout(self.horizontalLayout_6)
        self.verticalLayout_5.addLayout(self.verticalLayout_2)
        self.horizontalLayout_3 = QtWidgets.QHBoxLayout()
        self.horizontalLayout_3.setObjectName("horizontalLayout_3")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setObjectName("label_2")
        self.horizontalLayout_3.addWidget(self.label_2)
        self.SubjectLineEdit = QtWidgets.QLineEdit(self.centralwidget)
        self.SubjectLineEdit.setObjectName("SubjectLineEdit")
        self.horizontalLayout_3.addWidget(self.SubjectLineEdit)
        self.verticalLayout_5.addLayout(self.horizontalLayout_3)
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setObjectName("label")
        self.verticalLayout.addWidget(self.label)
        self.MessageEdit = QtWidgets.QTextEdit(self.centralwidget)
        self.MessageEdit.setObjectName("MessageEdit")
        self.verticalLayout.addWidget(self.MessageEdit)
        self.SendPushButton = QtWidgets.QPushButton(self.centralwidget)
        self.SendPushButton.setObjectName("SendPushButton")
        self.verticalLayout.addWidget(self.SendPushButton)
        self.verticalLayout_5.addLayout(self.verticalLayout)
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 330, 18))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.label_3.setText(_translate("MainWindow", "Сервер"))
        self.label_4.setText(_translate("MainWindow", "Адрес Отправителя"))
        self.label_5.setText(_translate("MainWindow", "Пароль"))
        self.label_6.setText(_translate("MainWindow", "Выбрать CSV файл"))
        self.AddCSVFilePushButton.setText(_translate("MainWindow", "Выбрать
файл"))
        self.label_2.setText(_translate("MainWindow", "Тема"))
        self.label.setText(_translate("MainWindow", "Сообщение "))
        self.SendPushButton.setText(_translate("MainWindow", "Отправить"))


if __name__ == "__main__":
    import sys
```

```
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Заключение

В результате выполнения было проведено ознакомление с теоретическим аппаратом MDA подходом при разработке и проектировании архитектур программного обеспечения.