Universitatea *Transilvania* din Braşov
Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor
Departamentul Automatică şi Tehnologia Informaţiei

PROIECT LA PA
Client Resource Management [CRM]

Nastase Alexandru
Gr. 4LF494, AIA II

Barbu Andrei Victoras
Gr. 4LF491, AIA II

Bors Dorin
Gr. 4LF494, AIA II

Olteanu Aura Mihaela
Gr. 4LF494, AIA II

CUPRINS

Cap.1. Informaţii generale

1.1 Scopul proiectului

Scopul acestui program este de a ușura muncă celor de la Business Project Agency, o firma de consultanță financiară. Astfel, după discuții cu clientul nostru, ne-am propus să facem o agenda digitală care le asigura o muncă mult mai ușoară. Aceștia în prezent folosesc diverse pdf-uri sau documente word pentru a ține evidență clienților și datele lor, care este un chin.

1.2 Client Resource Management

Un client resource management/crm este o unealtă folosită în majoritatea firmelor care trebuie să țină evidență unor aspecte și date ale clienților săi. Mai concret, este o agenda digitală care conține nume, date calendaristice, deadline-uri, date de contact, etc. Această ajută la salvarea timpului angajaților, asigura o funcționare mai bună a comunicării dintre client și firma și scade considerabil dacă nu chiar de tot riscul că un document să nu fie depus la timp, sau o întâlnire să fie uitată.

1.3 Organizarea programului

Cele 4 echipe care colaborează la acest proiect sunt echipele 22,24,28 și 37.Noi am împărțit câte un task pe echipa, fiecare echipa având câte un responsabil.Acești „responsabili” vorbest și împart datele între ei, fiecare echipa contribuind prin task-ul care trebuie rezolvat. Acest program este format pe moment dintr-o baza de date asupra căreia acționează anumite module, și care este accesat printr-o interfață.

1.4 Module şi algoritmi

Acest program pe moment o să aibă 2 module. Modulul de acțiune și cel de trimitere date. Modulul de acțiune se ocupă de orice interacțiune între utilizatorul programului și client. Mai concret, utilizatorul poate creea o acțiune care este aplicată unui client din baza de date, și are diverse opțiuni la anumită acțiune. De exemplu, utilizatorul dorește să primească un contract până într-o anumită dată, acesta creează acțiunea, o denumește, își creează un deadline și își pune un reminder(sau mai multe) înainte de deadline. În momentul în care acțiunea a fost îndeplinită, o poate selecta că fiind terminată, această urmând să fie ștearsă din baza de date. Modulul de trimitere de date se referă mai exact la un cod care îi permite utilizatorului să trimită un model de e-mail la fiecare client ales din baza de date.

Cap.2. Interfata

Fig.1 Interfata secundara

Interfață secundară cu rolul de a trimite E-mail-uri este de tip Qwidget. Această interfață conține următoarele elemente:
-o cutie de acțiuni generale cu patru butoane de tip QPushButton;
-două cutii de acțiuni de client cu patru butoane de tip QcheckBox;
-o căsuța folosită pentru afișarea atașamentelor;
-trei widget-uri principale de tip QtableWidget unde se va scrie e-mail-ul, subiectul și mesajul către destinatar;

Interfață secundară se poate accesa prin intermediul butonului „Send E-mail” din interfață principala. La accesarea interfeței, putem crea conținutul unui e-mail prin intermediul celor 3 widget-uri: „Către”, „Subiect” și „Mesaj”

E-mail-ul poate fi editat prin intermediul butoanelor QcheckBox, astfel putem schimbă stiul textului în „**Bold**”, „*Italic”, „*Underline” sau „~~Strikeout”~~

Prin intermediul căsuței „Font text” putem schimbă font-ul textului în diferite modele cum ar fi: Calibri, T.N.R., Arial sau Courier, mărimea textului fiind ușor setată prin SpinBox-ul „Mărime”.

În final, putem trimite E-mail-ul prin apăsarea butonului „Trimite” sau îl putem salva sau șterge în cazul în care suntem în dubiu cu privire la conținutul acestuia cu ajutorul butoanelor „Salvare” sau „Șterge”.

Cap.3. Cod sursa

3.1 Descrierea algoritmilor

Limbajul de programare principal este realizat în Python cu ajutorul plugin-ului PyQt5.

PyQt este o legare Python a setului de instrumente GUI cross-platform Qt , implementată că un plugin Python. PyQt acceptă Microsoft Windows , precum și diverse arome de UNIX , inclusiv Linux și MacOS (sau Darwin) .

PyQt implementează în jur de 440 de clase și peste 6.000 de funcții și metode, inclusiv:

- un set substanțial de widget-uri GUI
- clase pentru accesarea bazelor de date SQL (ODBC , MySQL , PostgreSQL , Oracle , SQLite)
- widget-uri conștiente de date care sunt completate automat dintr-o bază de date PyQt

3.2 Cod widgets

Pentru realizarea și punerea în practică a butoanelor: “Trimite”, “Salvează”, “Renunta” cât și QCheckBox-urile “Bold” “Italic” “Underline” și “StrikeOut” a fost folosit setul de instrucțiuni din imagine(fig 2) prin intermediul clasei EmailWindow(QWidget). Liniile de cod din fig 2 conțin variabile cu diferite atribuții implementate în aplicația QtDesigner (fig 3)

```
class EmailWindow(QWidget):
    def __init__(self):
        super().__init__()
        uic.loadUi(r"..\ui\EmailInterface.ui", self)
        self.setWindowTitle("Send E-mail")

        # Widgets
        self.Qmain_email = self.findChild(QWidget, "Qmain_email")
        self.Qcontent_attachments = self.findChild(QWidget, "Qcontent_attachments")
        self.Qtext_message = self.findChild(QTextEdit, "Qtext_message")
        self.Qtext_subject = self.findChild(QPlainTextEdit, "Qtext_subject")
        self.Qtext_to = self.findChild(QPlainTextEdit, "Qtext_to")
        self.Qbutton_send = self.findChild(QPushButton, "Qbutton_send")
        self.Qbutton_save = self.findChild(QPushButton, "Qbutton_save")
        self.Qbutton_attach = self.findChild(QPushButton, "Qbutton_attach")
        self.Qbutton_abort = self.findChild(QPushButton, "Qbutton_abort")
        self.Qcheck_bold = self.findChild(QCheckBox, "Qcheck_bold")
        self.Qcheck_italic = self.findChild(QCheckBox, "Qcheck_italic")
        self.Qcheck_underline = self.findChild(QCheckBox, "Qcheck_underline")
        self.Qcheck_strikeout = self.findChild(QCheckBox, "Qcheck_strikout")
        self.Qcheck_arial = self.findChild(QCheckBox, "Qcheck_arial")
        self.Qcheck_calibri = self.findChild(QCheckBox, "Qcheck_calibri")
        self.Qcheck_calibrik_courier = self.findChild(QCheckBox, "Qcheck_courier")
        self.Qcheck_tnr = self.findChild(QCheckBox, "Qcheck_tnr")
        self.Qsbox_size = self.findChild(QSpinBox, "Qsbox_size")
```

Fig.2. Cod Widgets

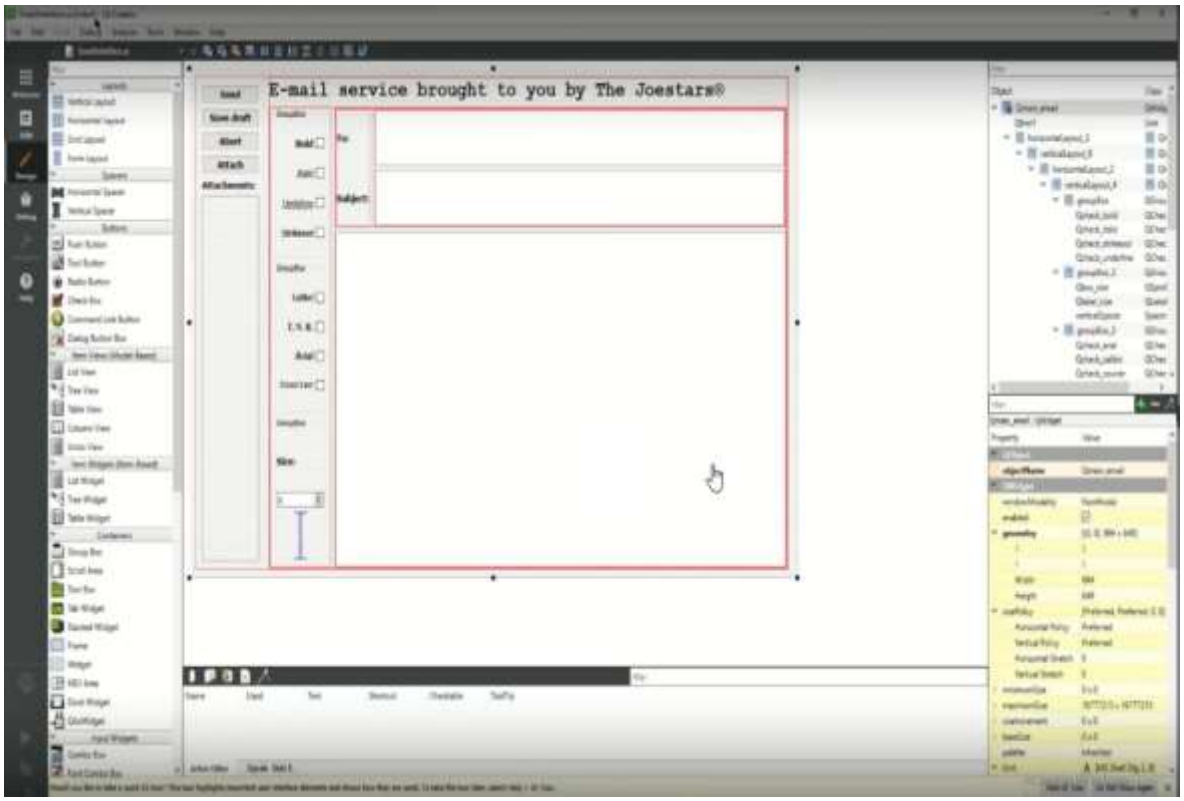

Fig 3. Qt Designer

3.3 Functia „loadUi”

```
uic.loadUi(r"..\ui\EmailInterface.ui", self)
```
Fig 4 LoadUi

Funcția de mai sus are că scop încărcarea interfeței realizată în aplicația Qt Designer. Toate variabilele create în aceast program se vor apela cu ajutorul funcției “Self”.

3.4 Butonul „Trimite”

Butonul „Trimite” prinde viață cu ajutorul liniilor de cod prezente în fig 5.

Prin apelarea funcției „def clicked_button_send(self)” variabilele „subiect”, „mesaj” și „msg['To']” preiau mesajul scris din căsuțele respective și le trimit către adresa de E-mail din secțiunea „To”.

Trimiterea E mail-ului poate fi realizată cu ajutorul modulului Smtplib

```
def clicked_button_send(self):
    print("send clicked")
    self.EMAIL_ADDRESS = 'itizicadevarat@gmail.com'
    self.EMAIL_PASSWORD = 'Testparola1'

    self.subiect = self.Qtext_subject.toPlainText()
    print(self.subiect)
    self.mesaj = self.Qtext_message.toHtml()
    print(self.mesaj)
    self.msg = EmailMessage()
    self.msg['Subject'] = self.subiect
    self.msg['From'] = self.EMAIL_ADDRESS
    self.msg['To'] = self.Qtext_to.toPlainText()
    self.msg.set_content(self.mesaj, subtype='html')
    print(self.msg['To'])
    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
        smtp.login(self.EMAIL_ADDRESS, self.EMAIL_PASSWORD)
        smtp.send_message(self.msg)
```

Fig 5 Butonul Trimite

3.5 Smtplib

Limbajul Python vine instalat cu modulul Smtplib pentru a trimite mailuri folosind Simple Mail Transfer Protocol (SMTP).
Funcția „smtp.login(self.EMAIL_ADDRESS, self.EMAIL_PASSWORD)” permite logarea în contul de E-mail accesând username-ul și parolă.
Funcția „smtp.send_message(self.msg)” trimite conținutul Email-ului. Structura msg conține subiectul, adresa destinatarului și mesajul.

3.6 Functia „def on_changed_bold(self)”, „def on_changed_italic(self)”, „def on_changed_underline(self)”, „def on_changed_strikeout(self)”

Funcțiile „def on_changed_... (self)” verifică prin secvență „if self. Qcheck_(...).isChecked()” dacă căsuțele din dreptul cuvintelor **Bold,** *Italic*, Underline si ~~Strikeout~~ sunt bifate**.**
In urma verificării, toate variabilele de tip Qcheck se vor debifa cu exceptia casutei respective.

```
def on_changed_bold(self):
    if self.Qcheck_bold.isChecked():
        self.Qcheck_italic.setChecked(False)
        self.Qcheck_underline.setChecked(False)
        self.Qcheck_strikeout.setChecked(False)
        self.Qcheck_tnr.setChecked(False)
        self.Qcheck_arial.setChecked(False)
        self.Qcheck_calibri.setChecked(False)
        self.Qcheck_calibrik_courier.setChecked(False)
```

Fig 6 def on_changed_(...)(self)

3.7 Formatarea textului

Interfata ne permite sa alegem stilul, fontul si marimea textului dupa cum putem vedea si in fig... In momentul in care acestea trei sunt selectate, in chenarul dedicat mesajului, interfata ne ofera o previzualizare a continutului e-mail-ului ce urmeaza a fi trimis.

Functia def on_changed_(...)(self) verifica daca o casuta din dreptul fontului ales este bifata astfel incat aceasta sa nu permita selectarea mai multor casute simultan.

In urma bifarii unei casute, comanda „mesaj=self.Qtext_message.toHtml()” preia textul anterior astfel incat informatia sa nu fie stearsa urmand ca mai apoi sa treaca printr-o serie de conditii si instructiuni.

Prin intermediul functiilor „if/elif „ programul verifica ce stil si font au fost bifate pentru a le pune in aplicare cu ajutorul liniei de cod:

```
"""{}<span style="font-family:'Calibri';font-size:{}px;"><u> {}</u></span>"""
                .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
```

Linia de cod difera pentru fiecare caz in parte, schimband doar denumirea fontului.

Fig.7 Exemplu

```
def on_changed_calibri(self):
    if self.Qcheck_calibri.isChecked():
        self.Qcheck_tnr.setChecked(False)
        self.Qcheck_arial.setChecked(False)
        self.Qcheck_calibrik_courier.setChecked(False)
        calibri = True
        mesaj = self.Qtext_message.toHtml()
        if self.Qcheck_underline.isChecked():
            self.Qtext_message.setHtml(
                """{}<span style="font-family:'Calibri';font-size:{}px;"><u> {}</u></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
        elif self.Qcheck_strikeout.isChecked():
            self.Qtext_message.setHtml(
                """{}<span style="font-family:'Calibri';font-size:{}px;"><s> {}</s></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))

        elif self.Qcheck_italic.isChecked():
            self.Qtext_message.setHtml(
                """{}<span style="font-family:'Calibri';font-size:{}px;"><i> {}</i></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
        elif self.Qcheck_bold.isChecked():
            self.Qtext_message.setHtml(
                """{}<span style="font-family:'Calibri';font-size:{}px;"><b> {}</b></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
        else:
            self.Qtext_message.setHtml(
                """{}<span style="font-family:'Calibri';font-size:{}px;"> {}</span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
    else:
        calibri = False
```

Fig.8 Formatarea textului

3.8 Butonul Renunta

În momentul în care butonul "Renunta" este apăsat, interfață de E-mail este închisă cu ajurorul funcției self.close și vei fi redirecționat către interfață principala "Main Window" a CRM-ului .

```
self.Qbutton_abort.clicked.connect(self.close)
```

3.9 Cod complet

```
import self as self
from PyQt5 import uic, QtWidgets
from PyQt5.QtGui import QTextCharFormat
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
import sys
import os
import smtplib
from email.message import EmailMessage
import imghdr
import time
import threading


class EmailWindow(QWidget):
    def __init__(self):
        super().__init__()
        uic.loadUi(r"..\ui\EmailInterface.ui", self)
        self.setWindowTitle("Send E-mail")

        # Widgets
        self.Qmain_email = self.findChild(QWidget, "Qmain_email")
        self.Qcontent_attachments = self.findChild(QWidget, "Qcontent_attachments")
        self.Qtext_message = self.findChild(QTextEdit, "Qtext_message")
        self.Qtext_subject = self.findChild(QPlainTextEdit, "Qtext_subject")
        self.Qtext_to = self.findChild(QPlainTextEdit, "Qtext_to")
        self.Qbutton_send = self.findChild(QPushButton, "Qbutton_send")
        self.Qbutton_save = self.findChild(QPushButton, "Qbutton_save")
        self.Qbutton_attach = self.findChild(QPushButton, "Qbutton_attach")
        self.Qbutton_abort = self.findChild(QPushButton, "Qbutton_abort")
        self.Qcheck_bold = self.findChild(QCheckBox, "Qcheck_bold")
        self.Qcheck_italic = self.findChild(QCheckBox, "Qcheck_italic")
        self.Qcheck_underline = self.findChild(QCheckBox, "Qcheck_underline")
        self.Qcheck_strikeout = self.findChild(QCheckBox, "Qcheck_strikeout")
        self.Qcheck_arial = self.findChild(QCheckBox, "Qcheck_arial")
        self.Qcheck_calibri = self.findChild(QCheckBox, "Qcheck_calibri")
        self.Qcheck_calibrik_courier = self.findChild(QCheckBox, "Qcheck_courier")
        self.Qcheck_tnr = self.findChild(QCheckBox, "Qcheck_tnr")
        self.Qsbox_size = self.findChild(QSpinBox, "Qsbox_size")

        self.Qbutton_attach.clicked.connect(self.clicked_button_attach)
        self.Qbutton_send.clicked.connect(self.clicked_button_send)
        self.Qbutton_abort.clicked.connect(self.close)
        self.Qcheck_bold.stateChanged.connect(self.on_changed_bold)
        self.Qcheck_italic.stateChanged.connect(self.on_changed_italic)
        self.Qcheck_underline.stateChanged.connect(self.on_changed_underline)
        self.Qcheck_strikeout.stateChanged.connect(self.on_changed_strikeout)

        self.Qcheck_calibri.stateChanged.connect(self.on_changed_calibri)
        self.Qcheck_tnr.stateChanged.connect(self.on_changed_tnr)
        self.Qcheck_arial.stateChanged.connect(self.on_changed_arial)
        self.Qcheck_calibrik_courier.stateChanged.connect(self.on_changed_calibrik_courier)
```

```
        self.Qsbox_size.valueChanged.connect(self.valueChanged_size)

    def clicked_button_attach(self):
        print("Attach clicked")

        # TODO

    def on_changed_bold(self):
        if self.Qcheck_bold.isChecked():
            self.Qcheck_italic.setChecked(False)
            self.Qcheck_underline.setChecked(False)
            self.Qcheck_strikeout.setChecked(False)
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibri.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)

    def on_changed_italic(self):
        if self.Qcheck_italic.isChecked():
            self.Qcheck_bold.setChecked(False)
            self.Qcheck_underline.setChecked(False)
            self.Qcheck_strikeout.setChecked(False)
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibri.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)
            italic = True
            # mesaj = self.Qtext_message.toHtml()
            # self.Qtext_message.setHtml('{}<i> {}</i>'.format(mesaj, self.Qtext_message))

        else:
            italic = False

    def on_changed_underline(self):
        if self.Qcheck_underline.isChecked():
            self.Qcheck_italic.setChecked(False)
            self.Qcheck_bold.setChecked(False)
            self.Qcheck_strikeout.setChecked(False)
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibri.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)
            underline = True
            # mesaj = self.Qtext_message.toHtml()
            # self.Qtext_message.setHtml('{}<u> {}</u>'.format(mesaj, self.Qtext_message))
        else:
            underline = False

    def on_changed_strikeout(self):
        if self.Qcheck_strikeout.isChecked():
            self.Qcheck_italic.setChecked(False)
            self.Qcheck_underline.setChecked(False)
            self.Qcheck_bold.setChecked(False)
```

```
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibri.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)
            # strikeout = True
            # mesaj=self.Qtext_message.toHtml()
            # self.Qtext_message.setHtml('{}<s> {}</s>'.format(mesaj, self.Qtext_message))
        else:
            strikeout = False

    def on_changed_calibri(self):
        if self.Qcheck_calibri.isChecked():
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)
            calibri = True
            mesaj = self.Qtext_message.toHtml()
            if self.Qcheck_underline.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Calibri';font-size:{}px;"><u> {}</u></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_strikeout.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Calibri';font-size:{}px;"><s> {}</s></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))

            elif self.Qcheck_italic.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Calibri';font-size:{}px;"><i> {}</i></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_bold.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Calibri';font-size:{}px;"><b> {}</b></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            else:
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Calibri';font-size:{}px;"> {}</span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
        else:
            calibri = False

    def on_changed_tnr(self):
        if self.Qcheck_tnr.isChecked():
            self.Qcheck_calibri.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)
            tnr = True
            mesaj = self.Qtext_message.toHtml()
            if self.Qcheck_underline.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Times New Roman';font-size:{}px;"><u> {}</u></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_strikeout.isChecked():
                self.Qtext_message.setHtml(
```

```
                """{}<span style="font-family:'Times New Roman';font-size:{}px;"><s> {}</s></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_italic.isChecked():
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Times New Roman';font-size:{}px;"><i> {}</i></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_bold.isChecked():
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Times New Roman';font-size:{}px;"><b> {}</b></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            else:
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Times New Roman';font-size:{}px;"> {}</span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            # self.Qtext_message.setHtml(
            #     "{}<span style=\"font-family:'Times New Roman';font-size:{}px;\"> {}</span>"
            #         .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
        else:
            tnr = False

    def on_changed_arial(self):
        if self.Qcheck_arial.isChecked():
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_calibri.setChecked(False)
            self.Qcheck_calibrik_courier.setChecked(False)
            arial = True
            mesaj = self.Qtext_message.toHtml()
            if self.Qcheck_underline.isChecked():
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Arial';font-size:{}px;"><u> {}</u></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_strikeout.isChecked():
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Arial';font-size:{}px;"><s> {}</s></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_italic.isChecked():
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Arial';font-size:{}px;"><i> {}</i></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_bold.isChecked():
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Arial';font-size:{}px;"><b> {}</b></span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            else:
                self.Qtext_message.setHtml(
                """{}<span style="font-family:'Arial';font-size:{}px;"> {}</span>"""
                    .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            # self.Qtext_message.setHtml("""{}<span style="font-family:'Arial';font-size:{}px;"> {} </span>"""
            #                            .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
        else:
            arial = False

    def on_changed_calibrik_courier(self):
        if self.Qcheck_calibrik_courier.isChecked():
```

```
            self.Qcheck_tnr.setChecked(False)
            self.Qcheck_arial.setChecked(False)
            self.Qcheck_calibri.setChecked(False)
            calibrik_courier = True
            mesaj = self.Qtext_message.toHtml()
            if self.Qcheck_underline.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Courier New';font-size:{}px;"><u> {}</u></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_strikeout.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Courier New';font-size:{}px;"><s> {}</s></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_italic.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Courier New';font-size:{}px;"><i> {}</i></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            elif self.Qcheck_bold.isChecked():
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Courier New';font-size:{}px;"><b> {}</b></span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            else:
                self.Qtext_message.setHtml(
                    """{}<span style="font-family:'Courier New';font-size:{}px;"> {}</span>"""
                        .format(mesaj, self.Qsbox_size.value(), self.Qtext_message))
            # self.Qtext_message.setHtml("""{}<span style="font-family:'Calibri'; font-size:{}px;"> {}
</span>"""
            #                            .format(mesaj,self.Qsbox_size.value(),  self.Qtext_message))
        else:
            calibrik_courier = False

    def valueChanged_size(self):
        value = self.Qsbox_size.value()
        self.Qsbox_size.setMaximum(72)
        self.Qcheck_italic.setChecked(False)
        self.Qcheck_underline.setChecked(False)
        self.Qcheck_strikeout.setChecked(False)
        self.Qcheck_bold.setChecked(False)
        self.Qcheck_tnr.setChecked(False)
        self.Qcheck_arial.setChecked(False)
        self.Qcheck_calibri.setChecked(False)
        self.Qcheck_calibrik_courier.setChecked(False)

    def clicked_button_send(self):
        print("send clicked")
        self.EMAIL_ADDRESS = 'itizicadevarat@gmail.com'
        self.EMAIL_PASSWORD = 'Testparola1'

        self.subiect = self.Qtext_subject.toPlainText()
        print(self.subiect)
        self.mesaj = self.Qtext_message.toHtml()
        print(self.mesaj)
        self.msg = EmailMessage()
        self.msg['Subject'] = self.subiect
```

```
        self.msg['From'] = self.EMAIL_ADDRESS
        self.msg['To'] = self.Qtext_to.toPlainText()
        self.msg.set_content(self.mesaj, subtype='html')
        print(self.msg['To'])
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            smtp.login(self.EMAIL_ADDRESS, self.EMAIL_PASSWORD)
            smtp.send_message(self.msg)


def show_window():
    app = QApplication([])
    window = EmailWindow()
    window.show()
    app.exec_()


if __name__ == "__main__":
    show_window()
```

Cap.4. Concluzii

4.1. Concluzii generale

La început proiectul nu părea ușor de realizat dar după divizarea problemelor în subcategorii de probleme am reușit să abordăm o strategie în direcția îndeplinirii totului unitar. Prin lucrarea această am învățat să ne expunem ideile și să relaționam totodată și cu un client.

În urmă lucrării date am deprins cunoștințe în Python, QT și am aprofundat lucrul cu bazele de date.

4.2. Elemente de originalitate

Că limbaj de programare ne-am axat pe Python pentru a îndeplinii viziunea proiectului într-un mod cât mai ușor de folosit și original.

Un alt aspect care a fost folosit original în această lucrare a fost creearea interfețelor folosind QT.

4.3. Direcții viitoare de dezvoltare

Baza de date trebuie încărcată pe serverul companiei iar algoritmul de căutare în baza de date trebuie definitizat.

4.4. Impresii

Proiectul prezinta un grad de dificultate ridicat si necesita foarte mult timp. Acest proiect ne-a ajutat sa descoperim si alte limbaje de programare si ne-a format spiritul de echipa.

Cap.5. Bibliografie

https://stackoverflow.com/questions/34398797/bold-font-in-label-with-setbold-method/34399490
https://programtalk.com/python-examples/PyQt5.QtGui.QFont.Bold/
https://www.programcreek.com/python/example/92655/PyQt5.QtGui.QFont.Bold
https://www.geeksforgeeks.org/pyqt5-qspinbox-making-text-italic/
https://www.geeksforgeeks.org/pyqt5-qspinbox-checking-if-text-is-bold/
https://www.binpress.com/building-text-editor-pyqt-1/
https://pythonpyqt.com/pyqt-label/
https://www.daniweb.com/programming/software-development/code/460444/make-e-mail-text-more-secure-pyqt-pyside
https://www.youtube.com/watch?v=RL9nGmv3uSU&ab_channel=CodeFirst
http://thecodeinn.blogspot.com/2013/08/tutorial-pyqt-email-app.html
https://github.com/cutelyst/simple-mail/
tps://doc.qt.io/qt-5/qplaintextedit.html
https://doc.qt.io/qt-5/qtextedit.html
https://doc.qt.io/qt-5/qtextedit.html
https://www.w3schools.com/
https://pythonbasics.org/qcheckbox/
https://pythonbasics.org/