Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Пермский национальный исследовательский

политехнический университет»

(ПНИПУ)

Кафедра вычислительной математики и механики

**Лабораторная работа № 3**

по дисциплине:

«**Информационная безопасность и защита информации**»

по теме:

«Разработка программы разграничения полномочий пользователей на основе парольной аутентификации»

Выполнил:

Студент группы ИСТ-19-1бзу

Соболь Е. В.

Проверил:

доцент кафедры ВММБ

Труфанов А. Н.

Пермь

2022 г.

**Цель работы:**

Ознакомится с методами разграничения доступа, получить практические навыки создания программного модуля разграничения доступа на основании парольной аутентификации.

**Описание алгоритма соответствия введённого пользователем пароля установленным в программе ограничениям:**

За проверку пароля при аутентификации отвечает функция Authorization класса AuthFrame.

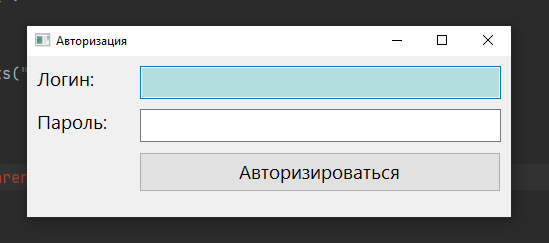


Рисунок 1 - Окно авторизации

Логин и пароль получаются из соответствующих полей, пароль преобразуется в md5-хеш. По логину запрашивается информация из БД, затем проверяется на наличие информации в ответе. Если её нет, значит, пользователь указал неверный логин, появляется уведомление об ошибке, а счётчик ошибочных вводов уменьшается.

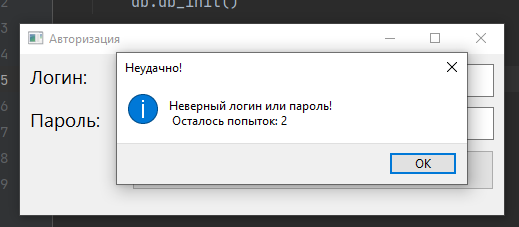


Рисунок 2 – Уведомление о неправильных учётных данных

Если данные в ответе есть, то они сравниваются с данными, полученными от пользователя: хеш пароля должен совпадать с хешем, находящимся в базе. При несовпадении появляется уведомление об ошибке, счётчик ошибочных вводов уменьшается. Если данные правильные и аутентификация прошла успешно, счётчик неудачных попыток обнуляется. После этого проверяется, не заблокирован ли пользователь.

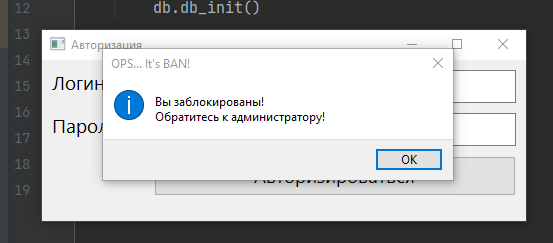


Рисунок 3 – Уведомление о блокировке пользователя

Если это так, то снова уведомление, и пользователь остаётся на окне авторизации. Дальше идёт проверка, новый ли это пользователь. Если нет, то он попадает в личный кабинет, в котором в зависимости от его привилегий появляется интерфейс программы.

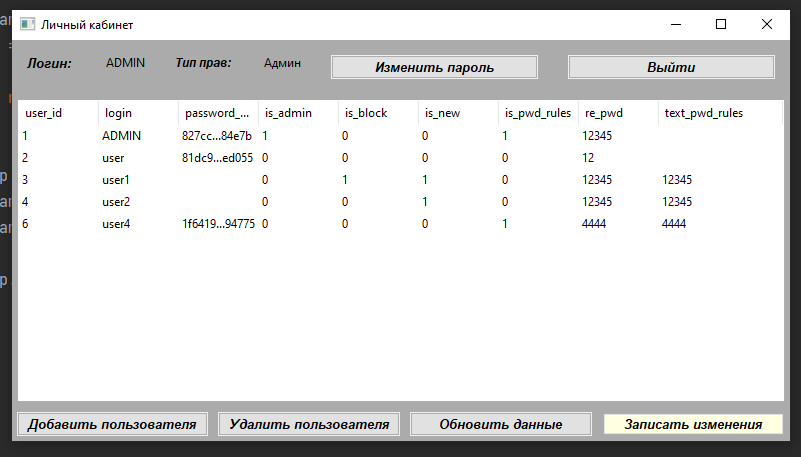


Рисунок 4 – Личный кабинет админа

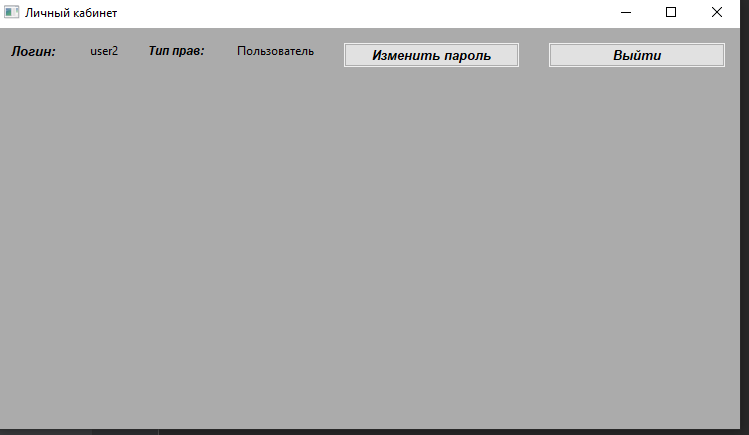


Рисунок 5 – Личный кабинет пользователя

Если же он новый, то попадает на форму смены пароля, за логику которой отвечает функция ChangePassword класса UpdatePasswordFrame.

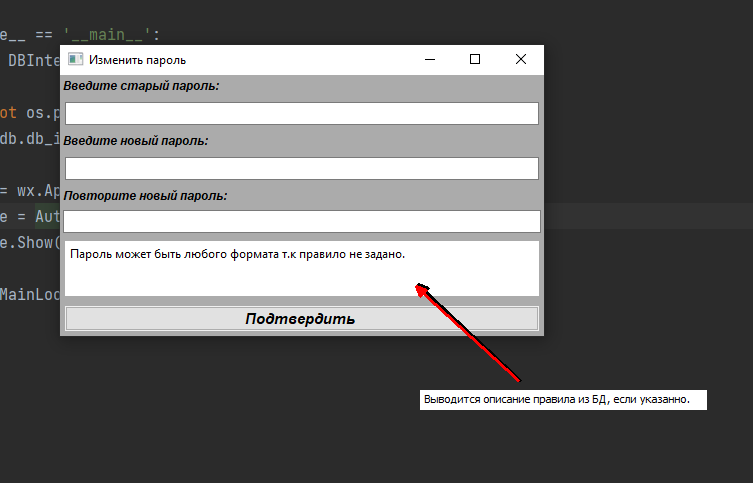


Рисунок 6 – Окно изменения пароля

В данной функции получаются значения 3 полей - старого пароля (может быть пустым), нового и повторения нового. Затем идёт проверка на ограничения для пароля. Если таких нет, значение флага соответствующего правила корректности принимает положительное значение, если нет, то отрицательное, и пароль будет дальше проверяться на соответствие правилам, заданным администратором в БД. Если поле для ввода старого пароля осталось пустым, то его значение устанавливается в None, иначе берётся md5 хеш и сравнивается с тем, что находится в базе. При несовпадении появляется уведомление об ошибке, при совпадении проверяется корректность нового пароля на соблюдение правил, указанных пользователю администратором (с помощью регулярного выражения, указанного в базе данных).

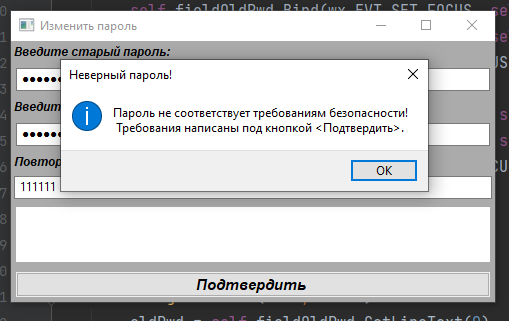


Рисунок 7 – Проверка соблюдения правил при смене пароля

Если проверка успешно пройдена, флаг корректности пароля устанавливается в положительное значение, проверяется идентичность в полях нового и подтверждения нового.

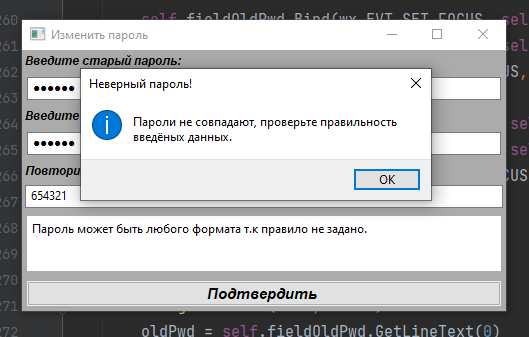


Рисунок 8 – Проверка совпадения указанных новых паролей

Если они не одинаковы - уведомление об ошибке, если всё успешно - берётся md5 хеш от нового пароля и записывается в базу взамен старого, пользователь получает уведомление об успешности смены пароля и предложение авторизоваться заново с помощью обновлённых учётных данных.

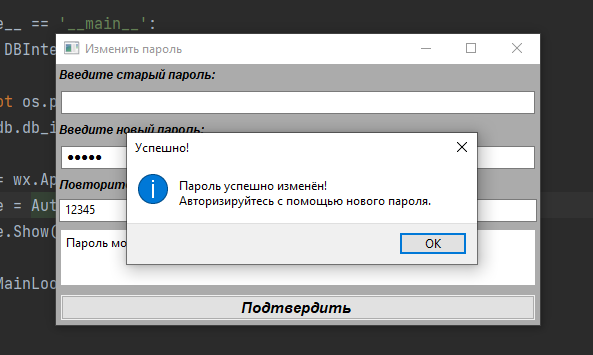


Рисунок 9 – Уведомление о успешной смене пароля

**Листинг программы:**

main.py

import wx  
import os  
  
from ui\_auth import AuthFrame  
from db\_func import DBInteraction  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 db = DBInteraction()  
  
 if not os.path.exists("users.db"):  
 db.db\_init()  
  
 app = wx.App()  
 frame = AuthFrame(parent=None)  
 frame.Show()  
  
 app.MainLoop()

db\_func.py – Процедуры для взаимодействия с БД

import sqlite3 as sq  
  
  
class DBInteraction:  
  
 def \_\_init\_\_(self, db\_path="users.db"):  
 self.con = sq.connect(db\_path)  
 self.con.row\_factory = sq.Row  
 self.cur = self.con.cursor()  
  
 def \_\_del\_\_(self):  
 if self.con:  
 self.con.close()  
  
 def get\_users(self):  
 self.cur.execute("SELECT \* FROM users")  
 return self.cur.fetchall()  
  
 def get\_all\_login(self):  
 self.cur.execute("SELECT login FROM users")  
 return self.cur.fetchall()  
  
 def get\_user(self, login: str):  
 self.cur.execute("""SELECT \* FROM users WHERE login = ?""", (login, ))  
 return self.cur.fetchall()  
  
 def add\_user(self, value\_dict):  
 self.cur.execute("""  
 INSERT INTO users(login, password\_hash, is\_admin,   
 is\_block, is\_new, is\_pwd\_rules, re\_pwd, text\_pwd\_rules)   
 VALUES (?, ?, ?, ?, ?, ?, ?, ?)""",  
 (value\_dict["login"], value\_dict["password\_hash"],  
 value\_dict["is\_admin"], value\_dict["is\_block"],  
 value\_dict["is\_new"], value\_dict["is\_pwd\_rules"],  
 value\_dict["re\_pwd"], value\_dict["text\_pwd\_rules"]))  
 self.con.commit()  
  
 def update\_user\_by\_login(self, login, value\_dict):  
 self.cur.execute("""  
 UPDATE users SET   
 password\_hash = ?, is\_admin = ?,   
 is\_block = ?,is\_new = ?,is\_pwd\_rules = ?,   
 re\_pwd = ?, text\_pwd\_rules = ? WHERE login = ?""",  
 (value\_dict["password\_hash"],  
 value\_dict["is\_admin"], value\_dict["is\_block"],  
 value\_dict["is\_new"], value\_dict["is\_pwd\_rules"],  
 value\_dict["re\_pwd"], value\_dict["text\_pwd\_rules"],  
 login  
 ))  
 self.con.commit()  
  
 def update\_user\_is\_new\_by\_login(self, value\_status, login):  
 self.cur.execute("""UPDATE users SET is\_new = ? WHERE login = ?""",  
 (value\_status, login)  
 )  
 self.con.commit()  
  
 def get\_password\_hash\_by\_login(self, login):  
 self.cur.execute("""  
 SELECT password\_hash FROM users WHERE login = ?  
 """, login)  
 return self.cur.fetchall()  
  
 def update\_password\_hash\_by\_login(self, login, pwd\_hash):  
 self.cur.execute(  
 """UPDATE users SET password\_hash = ? WHERE login = ?""",  
 (pwd\_hash, login)  
 )  
 self.con.commit()  
  
 def delete\_user\_by\_login(self, login):  
 self.cur.execute(  
 """DELETE FROM users WHERE login = ?""", (login, )  
 )  
 self.con.commit()  
  
 def db\_init(self):  
 with open("db\_init.sql", "r") as f:  
 sql = f.read()  
 self.cur.executescript(sql)

db\_init.sql – скрипт для инициализации БД

BEGIN TRANSACTION;CREATE TABLE users (  
 user\_id INTEGER PRIMARY KEY,  
 login TEXT NOT NULL UNIQUE ,  
 password\_hash TEXT,  
 is\_admin INTEGER NOT NULL DEFAULT 0,  
 is\_block INTEGER NOT NULL DEFAULT 0,  
 is\_new INTEGER NOT NULL DEFAULT 1,  
 is\_pwd\_rools INTEGER NOT NULL DEFAULT 0,  
 re\_pwd TEXT,  
 text\_pwd\_rools TEXT  
 );INSERT INTO "users" VALUES(1,'ADMIN',NULL,1,0,1,0,NULL,NULL);COMMIT;

text\_values\_elements.py – все текстовые значения

# Названия окон  
NAME\_WINDOW\_AUTH = "Авторизация"  
NAME\_WINDOW\_ERROR\_BAN = "OPS... It's BAN!"  
NAME\_WINDOW\_ERROR\_AUTH = "Неудачно!"  
NAME\_WINDOW\_ERROR\_UPDATE\_PASSWORD = "Неверный пароль!"  
NAME\_WINDOW\_SUCCESSFULLY\_PASSWORD = "Успешно!"  
NAME\_WINDOW\_UPDATE\_PASSWORD = "Изменить пароль"  
NAME\_WINDOW\_PERSONAL\_AREA = u"Личный кабинет"  
NAME\_WINDOW\_ERROR\_UPDATE\_DATA = "Данные не корректны!"  
NAME\_WINDOW\_ERROR\_DELETE\_USER = "Ошибка при удалении!"  
  
# Названия кнопок  
BUTTON\_AUTH = "Авторизироваться"  
BUTTON\_CONFIRM\_UPDATE\_PASSWORD = u"Подтвердить "  
BUTTON\_UPDATE\_PASSWORD = u"Изменить пароль"  
BUTTON\_EXIT\_PERSONAL\_AREA = u"Выйти"  
BUTTON\_ADD\_USER = u"Добавить пользователя"  
BUTTON\_DELETE\_USER = u"Удалить пользователя"  
BUTTON\_UPDATE\_USER = u"Обновить данные"  
BUTTON\_SAVE\_UPDATE = u"Записать изменения"  
  
# Текст интерфейса  
TEXT\_LOGIN\_AUTH = "Логин: "  
TEXT\_PASSWORD\_AUTH = "Пароль: "  
TEXT\_OLD\_PASSWORD = u"Введите старый пароль: "  
TEXT\_NEW\_PASSWORD = u"Введите новый пароль:"  
TEXT\_CONFIRM\_NEW\_PASSWORD = u"Повторите новый пароль:"  
TEXT\_LOGIN\_PERSONAL\_AREA = u"Логин: "  
TEXT\_RIGHTS\_PERSONAL\_AREA = u"Тип прав: "  
TEXT\_RIGHTS\_USER = "Пользователь"  
TEXT\_RIGHTS\_ADMIN = "Админ"  
  
  
# Текст сообщений  
MSG\_BLOCK\_USER = "Вы заблокированы! \nОбратитесь к администратору!"  
MSG\_FAILED\_PASSWORD\_RULES = f"Пароль не соответствует требованиям безопасности!\n " \  
 f"Требования написаны под кнопкой <Подтвердить>."  
MSG\_FAILED\_PASSWORD\_CONFIRM = f"Пароли не совпадают, проверьте правильность\nвведёных данных."  
MSG\_FAILED\_OLD\_PASSWORD = f"Старый пароль введён некорректно,\n проверьте правильность указанных данных."  
MSG\_SUCCESSFULLY\_PASSWORD = f"Пароль успешно изменён!\nАвторизируйтесь с помощью нового пароля."  
MSG\_NO\_RULES\_PASSWORD = "Пароль может быть любого формата т.к правило не задано."  
MSG\_NO\_TEXT\_RULES\_PASSWORD = "Администратор не описал правило,\n но задал шаблон для проверки!" \  
 "\nОбратитесь за помощью к ему."  
MSG\_EMPTY\_FIELD = "Столбцы 2, 4, 5, 6, 7 - не должны быть пустыми!"  
MSG\_NON\_UNIQUE\_LOGIN = "Логин должен быть уникальным!"  
MSG\_NO\_SELECT\_ROW = "Не выбран ни один пользователь!"  
MSG\_NO\_DELETE\_SELF = "Нельзя удалить себя!"  
  
  
def error\_auth(count\_error):  
 return f"Неверный логин или пароль!\n " \  
 f"Осталось попыток: {count\_error}"

ui\_auth.py – Основная логика и UI

import wx  
import wx.xrc  
import wx.richtext  
import wx.dataview  
  
import re  
import collections  
import hashlib  
  
from db\_func import DBInteraction  
import text\_values\_elements as tve  
  
  
class AuthFrame(wx.Frame):  
 CODE\_AUTH = 100  
  
 def \_\_init\_\_(  
 self, parent, title=tve.NAME\_WINDOW\_AUTH,  
 size=(500, 200), pos=wx.DefaultPosition  
 ):  
 super().\_\_init\_\_(  
 parent, title=title, size=size, pos=pos  
 )  
 self.db = DBInteraction()  
 self.count\_error = 3  
 self.response = None  
 panel = wx.Panel(self)  
 vbox = wx.BoxSizer(wx.VERTICAL)  
  
 font = wx.SystemSettings.GetFont(wx.SYS\_DEFAULT\_GUI\_FONT)  
 font.SetPointSize(14)  
 panel.SetFont(font)  
  
 h\_box\_login = wx.BoxSizer(wx.HORIZONTAL)  
 text\_login = wx.StaticText(panel, label=tve.TEXT\_LOGIN\_AUTH)  
 self.field\_login = wx.TextCtrl(panel)  
 h\_box\_login.Add(text\_login, flag=wx.RIGHT, border=41)  
 h\_box\_login.Add(self.field\_login, proportion=1)  
 vbox.Add(  
 h\_box\_login,  
 flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP,  
 border=10  
 )  
  
 h\_box\_pwd = wx.BoxSizer(wx.HORIZONTAL)  
 text\_pwd = wx.StaticText(panel, label=tve.TEXT\_PASSWORD\_AUTH)  
 self.field\_pwd = wx.TextCtrl(panel, style=wx.TE\_PASSWORD)  
 h\_box\_pwd.Add(text\_pwd, flag=wx.RIGHT, border=8)  
 h\_box\_pwd.Add(self.field\_pwd, proportion=1)  
 vbox.Add(  
 h\_box\_pwd,  
 flag=wx.EXPAND | wx.LEFT | wx.RIGHT | wx.TOP,  
 border=10  
 )  
  
 text\_empty = wx.StaticText(panel)  
 btn\_login = wx.Button(  
 panel, id=self.CODE\_AUTH, label=tve.BUTTON\_AUTH, size=(300, 40)  
 )  
 h\_box\_button\_login = wx.BoxSizer(wx.HORIZONTAL)  
 h\_box\_button\_login.Add(text\_empty, flag=wx.RIGHT, border=102)  
 h\_box\_button\_login.Add(btn\_login, proportion=1)  
  
 vbox.Add(  
 h\_box\_button\_login,  
 flag=wx.EXPAND | wx.TOP | wx.LEFT | wx.RIGHT,  
 border=10  
 )  
  
 panel.SetSizer(vbox)  
  
 self.Centre(wx.BOTH)  
  
 btn\_login.Bind(wx.EVT\_BUTTON, self.Authorization, id=self.CODE\_AUTH)  
  
 self.field\_login.Bind(wx.EVT\_SET\_FOCUS, self.onSetFocus)  
 self.field\_pwd.Bind(wx.EVT\_SET\_FOCUS, self.onSetFocus)  
  
 self.field\_login.Bind(wx.EVT\_KILL\_FOCUS, self.onKillFocus)  
 self.field\_pwd.Bind(wx.EVT\_KILL\_FOCUS, self.onKillFocus)  
  
 @staticmethod  
 def hash\_in\_md5(pwd):  
 return hashlib.md5(pwd.encode('utf-8')).hexdigest()  
  
 def Authorization(self, event):  
 login = self.field\_login.GetLineText(0)  
 pwd = self.field\_pwd.GetLineText(0)  
 if pwd:  
 pwd\_hash = self.hash\_in\_md5(pwd)  
 else:  
 pwd\_hash = None  
 self.response = self.db.get\_user(login)  
 if self.response:  
 self.response = self.response[0]  
 if (self.response['login'] == login) and \  
 (self.response["password\_hash"] == pwd\_hash):  
 self.count\_error = 3  
 if not self.response['is\_block']:  
 if self.response['is\_new']:  
 self.OpenNewPwdWindow(self.response, self.db)  
 else:  
 self.OpenPersonalAreaWindow(  
 self.response, self.db  
 )  
 else:  
 self.msgBan()  
 else:  
 self.failedAuth()  
 else:  
 self.failedAuth()  
  
 @staticmethod  
 def msgBan():  
 return wx.MessageBox(  
 tve.MSG\_BLOCK\_USER,  
 tve.NAME\_WINDOW\_ERROR\_BAN, wx.OK | wx.OK\_DEFAULT  
 )  
  
 def failedAuth(self):  
 self.count\_error -= 1  
 if self.count\_error <= 0:  
 self.Close()  
 exit(1)  
 return wx.MessageBox(  
 tve.error\_auth(self.count\_error),  
 tve.NAME\_WINDOW\_ERROR\_AUTH, wx.OK | wx.OK\_DEFAULT  
 )  
  
 def onSetFocus(self, event):  
 event.GetEventObject().SetBackgroundColour("#b2dfdb")  
 event.Skip()  
  
 def onKillFocus(self, event):  
 event.GetEventObject().SetBackgroundColour("#FFF")  
 event.Skip()  
  
 def OpenNewPwdWindow(self, response, db: DBInteraction):  
 frame = UpdatePasswordFrame(response, db)  
 self.Close()  
  
 def OpenPersonalAreaWindow(self, response, db: DBInteraction):  
 frame = PersonalAreaFrame(response, db)  
 self.Close()  
  
  
class UpdatePasswordFrame(AuthFrame):  
 CODE\_BUTTON\_CONFIRM = 200  
  
 def \_\_init\_\_(self, response, db: DBInteraction):  
 wx.Frame.\_\_init\_\_(  
 self, parent=None, id=wx.ID\_ANY, title=tve.NAME\_WINDOW\_UPDATE\_PASSWORD, pos=wx.DefaultPosition,  
 size=wx.Size(500, 300), style=wx.DEFAULT\_FRAME\_STYLE | wx.TAB\_TRAVERSAL  
 )  
  
 self.Show()  
 self.response = response  
 self.db = db  
  
 self.SetSizeHints(wx.DefaultSize, wx.DefaultSize)  
  
 bSizerMain = wx.BoxSizer(wx.VERTICAL)  
 bSizerOldPwd = wx.BoxSizer(wx.VERTICAL)  
  
 self.TextOldPwd = wx.StaticText(  
 self, wx.ID\_ANY, tve.TEXT\_OLD\_PASSWORD,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.TextOldPwd.Wrap(-1)  
 self.TextOldPwd.SetFont(  
 wx.Font(  
 wx.NORMAL\_FONT.GetPointSize(), wx.FONTFAMILY\_DEFAULT,  
 wx.FONTSTYLE\_ITALIC, wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerOldPwd.Add(self.TextOldPwd, 0, wx.ALL | wx.EXPAND, 3)  
  
 self.fieldOldPwd = wx.TextCtrl(  
 self, wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition,  
 wx.DefaultSize, wx.TE\_PASSWORD  
 )  
 bSizerOldPwd.Add(self.fieldOldPwd, 0, wx.ALL | wx.EXPAND, 5)  
  
 bSizerMain.Add(bSizerOldPwd, 0, wx.EXPAND, 5)  
  
 bSizerNewPwd = wx.BoxSizer(wx.VERTICAL)  
  
 self.TextNewPwd = wx.StaticText(  
 self, wx.ID\_ANY, tve.TEXT\_NEW\_PASSWORD,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.TextNewPwd.Wrap(-1)  
 self.TextNewPwd.SetFont(  
 wx.Font(  
 wx.NORMAL\_FONT.GetPointSize(), wx.FONTFAMILY\_DEFAULT,  
 wx.FONTSTYLE\_ITALIC, wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerNewPwd.Add(self.TextNewPwd, 0, wx.ALL, 3)  
  
 self.fieldNewPwd = wx.TextCtrl(  
 self, wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition,  
 wx.DefaultSize, wx.TE\_PASSWORD  
 )  
 bSizerNewPwd.Add(self.fieldNewPwd, 0, wx.ALL | wx.EXPAND, 5)  
  
 bSizerMain.Add(bSizerNewPwd, 0, wx.EXPAND, 5)  
  
 bSizerConfirmPwd = wx.BoxSizer(wx.VERTICAL)  
  
 self.TextConfirmPwd = wx.StaticText(  
 self, wx.ID\_ANY, tve.TEXT\_CONFIRM\_NEW\_PASSWORD,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.TextConfirmPwd.Wrap(-1)  
 self.TextConfirmPwd.SetFont(  
 wx.Font(  
 wx.NORMAL\_FONT.GetPointSize(), wx.FONTFAMILY\_DEFAULT,  
 wx.FONTSTYLE\_ITALIC, wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerConfirmPwd.Add(self.TextConfirmPwd, 0, wx.ALL, 3)  
  
 self.fieldConfirmPwd = wx.TextCtrl(  
 self, wx.ID\_ANY, wx.EmptyString, wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 bSizerConfirmPwd.Add(self.fieldConfirmPwd, 0, wx.ALL | wx.EXPAND, 3)  
  
 bSizerMain.Add(bSizerConfirmPwd, 0, wx.EXPAND, 5)  
  
 bSizerRulesAndButton = wx.BoxSizer(wx.VERTICAL)  
  
 self.TextRulesPwd = wx.richtext.RichTextCtrl(  
 self, wx.ID\_ANY, self.GetTextRules(), wx.DefaultPosition, wx.DefaultSize,  
 0 | wx.VSCROLL | wx.HSCROLL | wx.NO\_BORDER | wx.WANTS\_CHARS  
 )  
 bSizerRulesAndButton.Add(self.TextRulesPwd, 1, wx.EXPAND | wx.ALL, 5)  
  
 self.buttonConfirm = wx.Button(  
 self, id=self.CODE\_BUTTON\_CONFIRM, label=tve.BUTTON\_CONFIRM\_UPDATE\_PASSWORD,  
 pos=wx.DefaultPosition, size=wx.DefaultSize, style=0  
 )  
 self.buttonConfirm.SetFont(  
 wx.Font(  
 11, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerRulesAndButton.Add(self.buttonConfirm, 0, wx.ALL | wx.EXPAND, 5)  
  
 bSizerMain.Add(bSizerRulesAndButton, 1, wx.EXPAND, 5)  
  
 self.SetSizer(bSizerMain)  
 self.Layout()  
  
 self.Centre(wx.BOTH)  
  
 self.Bind(wx.EVT\_BUTTON, self.ChangePassword, id=self.CODE\_BUTTON\_CONFIRM)  
  
 self.fieldOldPwd.Bind(wx.EVT\_SET\_FOCUS, self.onSetFocus)  
 self.fieldNewPwd.Bind(wx.EVT\_SET\_FOCUS, self.onSetFocus)  
 self.fieldConfirmPwd.Bind(wx.EVT\_SET\_FOCUS, self.onSetFocus)  
  
 self.fieldOldPwd.Bind(wx.EVT\_KILL\_FOCUS, self.onKillFocus)  
 self.fieldNewPwd.Bind(wx.EVT\_KILL\_FOCUS, self.onKillFocus)  
 self.fieldConfirmPwd.Bind(wx.EVT\_KILL\_FOCUS, self.onKillFocus)  
  
 def \_\_del\_\_(self):  
 pass  
  
 def ChangePassword(self, event):  
 oldPwd = self.fieldOldPwd.GetLineText(0)  
 newPwd = self.fieldNewPwd.GetLineText(0)  
 confirmNewPwd = self.fieldConfirmPwd.GetLineText(0)  
 correctPwd = 1 if self.response['is\_pwd\_rules'] == 0 else 0  
 oldPwdHash = None if (oldPwd == "") or (oldPwd is None) \  
 else self.hash\_in\_md5(oldPwd)  
 if oldPwdHash == self.response["password\_hash"]:  
 if not correctPwd:  
 if re.search(self.response["re\_pwd"], newPwd) or (self.response["re\_pwd"] is None):  
 correctPwd = 1  
 else:  
 correctPwd = 0  
 self.PwdMsg(  
 tve.MSG\_FAILED\_PASSWORD\_RULES,  
 tve.NAME\_WINDOW\_ERROR\_UPDATE\_PASSWORD  
 )  
 else:  
 correctPwd = 0  
 self.PwdMsg(  
 tve.MSG\_FAILED\_OLD\_PASSWORD,  
 tve.NAME\_WINDOW\_ERROR\_UPDATE\_PASSWORD  
 )  
 if correctPwd:  
 print(confirmNewPwd, " == ", newPwd)  
 if confirmNewPwd == newPwd:  
 newPwd = self.hash\_in\_md5(newPwd)  
 self.db.update\_password\_hash\_by\_login(self.response['login'], newPwd)  
 self.PwdMsg(  
 tve.MSG\_SUCCESSFULLY\_PASSWORD,  
 tve.NAME\_WINDOW\_SUCCESSFULLY\_PASSWORD  
 )  
 self.UpdateStatusIsNew()  
 self.GoToOpenAuthPwd()  
 else:  
 self.PwdMsg(  
 tve.MSG\_FAILED\_PASSWORD\_CONFIRM,  
 tve.NAME\_WINDOW\_ERROR\_UPDATE\_PASSWORD  
 )  
  
 def GetTextRules(self):  
 if (self.response['text\_pwd\_rules'] is None) and \  
 (self.response['is\_pwd\_rules'] == 1):  
 if (self.response["re\_pwd"] is None) or \  
 (self.response["re\_pwd"] == ""):  
 return tve.MSG\_NO\_RULES\_PASSWORD  
 else:  
 return tve.MSG\_NO\_TEXT\_RULES\_PASSWORD  
 else:  
 if self.response['is\_pwd\_rules'] == 0:  
 return tve.MSG\_NO\_RULES\_PASSWORD  
 else:  
 return self.response['text\_pwd\_rules']  
  
 @staticmethod  
 def PwdMsg(msg, window\_name):  
 return wx.MessageBox(  
 msg, window\_name, wx.OK | wx.OK\_DEFAULT  
 )  
  
 def UpdateStatusIsNew(self):  
 self.db.update\_user\_is\_new\_by\_login(  
 0, self.response['login']  
 )  
  
 def GoToOpenAuthPwd(self):  
 frame = AuthFrame(None)  
 frame.Show()  
 self.Close()  
  
  
class PersonalAreaFrame(UpdatePasswordFrame):  
 CODE\_EXIT\_TO\_AUTH = 300  
 CODE\_CHANGE\_PWD = 301  
 CODE\_UPDATE\_DATA = 302  
 CODE\_GET\_UPDATE\_TABLE = 303  
 CODE\_ADD\_USER = 304  
 CODE\_DELETE\_USER = 305  
  
 def \_\_init\_\_(self, response, db: DBInteraction):  
 wx.Frame.\_\_init\_\_(  
 self, parent=None, id=wx.ID\_ANY, title=tve.NAME\_WINDOW\_PERSONAL\_AREA,  
 pos=wx.DefaultPosition, size=wx.Size(760, 440),  
 style=wx.DEFAULT\_FRAME\_STYLE | wx.TAB\_TRAVERSAL  
 )  
  
 self.Show()  
 self.response = response  
 self.db = db  
 self.all\_users = self.db.get\_users()  
  
 self.SetSizeHints(wx.DefaultSize, wx.DefaultSize)  
  
 bSizerMain = wx.BoxSizer(wx.VERTICAL)  
  
 bSizerMenu = wx.BoxSizer(wx.HORIZONTAL)  
  
 self.textLoginName = wx.StaticText(  
 self, wx.ID\_ANY, tve.TEXT\_LOGIN\_PERSONAL\_AREA,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.textLoginName.Wrap(-1)  
 self.textLoginName.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerMenu.Add(self.textLoginName, 0, wx.ALL | wx.RIGHT, 15)  
  
 self.textLoginValue = wx.StaticText(  
 self, wx.ID\_ANY, self.response['login'],  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.textLoginValue.Wrap(-1)  
 bSizerMenu.Add(self.textLoginValue, 0, wx.ALL, 15)  
  
 self.textRightsName = wx.StaticText(  
 self, wx.ID\_ANY, tve.TEXT\_RIGHTS\_PERSONAL\_AREA,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.textRightsName.Wrap(-1)  
 self.textRightsName.SetFont(  
 wx.Font(  
 wx.NORMAL\_FONT.GetPointSize(), wx.FONTFAMILY\_DEFAULT,  
 wx.FONTSTYLE\_ITALIC, wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerMenu.Add(self.textRightsName, 0, wx.ALL, 15)  
  
 self.textRightsValue = wx.StaticText(  
 self, wx.ID\_ANY, self.define\_rights(),  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.textRightsValue.Wrap(-1)  
 bSizerMenu.Add(self.textRightsValue, 0, wx.ALL, 15)  
  
 self.buttonChangePwd = wx.Button(  
 self, self.CODE\_CHANGE\_PWD, tve.BUTTON\_UPDATE\_PASSWORD,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.buttonChangePwd.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerMenu.Add(self.buttonChangePwd, 1, wx.ALL, 15)  
  
 self.buttonExitAuth = wx.Button(  
 self, self.CODE\_EXIT\_TO\_AUTH, tve.BUTTON\_EXIT\_PERSONAL\_AREA,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.buttonExitAuth.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerMenu.Add(self.buttonExitAuth, 1, wx.ALL, 15)  
  
 bSizerMain.Add(bSizerMenu, 0, wx.EXPAND, 5)  
  
 if self.response['is\_admin']:  
 bSizerTableUsers = wx.BoxSizer(wx.VERTICAL)  
  
 self.dataViewList = wx.dataview.DataViewListCtrl(  
 self, wx.ID\_ANY, wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.fillInTable(is\_primary\_filling=True)  
  
 bSizerTableUsers.Add(self.dataViewList, 1, wx.ALL | wx.EXPAND, 5)  
  
 bSizerMain.Add(bSizerTableUsers, 1, wx.EXPAND, 5)  
  
 bSizerButtonUser = wx.BoxSizer(wx.HORIZONTAL)  
  
 self.buttonCreateUser = wx.Button(  
 self, self.CODE\_ADD\_USER, tve.BUTTON\_ADD\_USER,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.buttonCreateUser.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerButtonUser.Add(self.buttonCreateUser, 1, wx.ALL | wx.EXPAND, 5)  
  
 self.buttonDeleteUser = wx.Button(  
 self, self.CODE\_DELETE\_USER, tve.BUTTON\_DELETE\_USER,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.buttonDeleteUser.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerButtonUser.Add(self.buttonDeleteUser, 1, wx.ALL | wx.EXPAND, 5)  
  
 self.buttonUpdateTable = wx.Button(  
 self, self.CODE\_GET\_UPDATE\_TABLE, tve.BUTTON\_UPDATE\_USER,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.buttonUpdateTable.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 bSizerButtonUser.Add(  
 self.buttonUpdateTable, 1, wx.ALL | wx.EXPAND, 5  
 )  
  
 self.buttonUpdateCompleted = wx.Button(  
 self, self.CODE\_UPDATE\_DATA, tve.BUTTON\_SAVE\_UPDATE,  
 wx.DefaultPosition, wx.DefaultSize, 0  
 )  
 self.buttonUpdateCompleted.SetFont(  
 wx.Font(  
 10, wx.FONTFAMILY\_DEFAULT, wx.FONTSTYLE\_ITALIC,  
 wx.FONTWEIGHT\_BOLD, False, wx.EmptyString  
 )  
 )  
 self.buttonUpdateCompleted.SetForegroundColour(  
 wx.SystemSettings.GetColour(wx.SYS\_COLOUR\_INFOTEXT)  
 )  
 self.buttonUpdateCompleted.SetBackgroundColour(  
 wx.SystemSettings.GetColour(wx.SYS\_COLOUR\_INFOBK)  
 )  
  
 bSizerButtonUser.Add(  
 self.buttonUpdateCompleted, 1, wx.ALL | wx.EXPAND, 5  
 )  
  
 bSizerMain.Add(bSizerButtonUser, 0, wx.EXPAND, 5)  
  
 self.SetSizer(bSizerMain)  
 self.Layout()  
  
 self.Centre(wx.BOTH)  
  
 self.Bind(wx.EVT\_BUTTON, self.exitToAuth, id=self.CODE\_EXIT\_TO\_AUTH)  
 self.Bind(wx.EVT\_BUTTON, self.changePwd, id=self.CODE\_CHANGE\_PWD)  
 if self.response['is\_admin']:  
 self.Bind(wx.EVT\_BUTTON, self.GetDataFromTable, id=self.CODE\_UPDATE\_DATA)  
 self.Bind(wx.EVT\_BUTTON, self.updateDataFromTable, id=self.CODE\_GET\_UPDATE\_TABLE)  
 self.Bind(wx.EVT\_BUTTON, self.addNewUser, id=self.CODE\_ADD\_USER)  
 self.Bind(wx.EVT\_BUTTON, self.deleteUser, id=self.CODE\_DELETE\_USER)  
  
 def \_\_del\_\_(self):  
 pass  
  
 def define\_rights(self):  
 if self.response['is\_admin'] == 1:  
 return tve.TEXT\_RIGHTS\_ADMIN  
 return tve.TEXT\_RIGHTS\_USER  
  
 def exitToAuth(self, event):  
 self.GoToOpenAuthPwd()  
  
 def changePwd(self, event):  
 self.OpenNewPwdWindow(self.response, self.db)  
  
 def updateDataFromTable(self, event):  
 self.dataViewList.DeleteAllItems()  
 self.all\_users = self.db.get\_users()  
 self.fillInTable()  
 self.response = self.db.get\_user(  
 self.response['login']  
 )[0]  
  
 def fillInTable(self, is\_primary\_filling=False):  
 if is\_primary\_filling:  
 for column\_name in self.response.keys():  
 mode = wx.dataview.DATAVIEW\_CELL\_EDITABLE if (column\_name != "user\_id") \  
 else wx.dataview.DATAVIEW\_CELL\_INERT  
 self.dataViewList.AppendTextColumn(  
 column\_name, mode=mode  
 )  
 for i in self.all\_users:  
 item = list(tuple(i))  
 item = self.ColumnIntToStr(item)  
 self.dataViewList.AppendItem(item)  
  
 def addNewUser(self, event):  
 defaultDataNewUser = [  
 None, "<new user>", None,  
 "0", "0", "1", "0", None, None  
 ]  
 self.dataViewList.AppendItem(defaultDataNewUser)  
  
 def GetDataFromTable(self, event):  
 for row in range(len(self.all\_users)):  
 row\_item = self.get\_in\_the\_line(row)  
 if collections.Counter(row\_item) != self.all\_users[row]:  
 value\_dict = dict(zip(self.response.keys(), row\_item))  
 value\_dict.pop('user\_id')  
 login = value\_dict.pop('login')  
 self.db.update\_user\_by\_login(login, value\_dict)  
 if len(self.all\_users) != self.dataViewList.GetItemCount():  
 for row in range(len(self.all\_users), self.dataViewList.GetItemCount()):  
 row\_item = self.get\_in\_the\_line(row)  
 value\_dict = dict(zip(self.response.keys(), row\_item))  
 if self.CheckNewUserData(value\_dict):  
 value\_dict.pop('user\_id')  
 self.db.add\_user(value\_dict)  
 self.updateDataFromTable(None)  
  
 def get\_in\_the\_line(self, row):  
 row\_item = []  
 for column in range(len(self.response.keys())):  
 row\_item.append(self.dataViewList.GetTextValue(row, column))  
 row\_item = self.ColumnIntToStr(row\_item, True)  
 return row\_item  
  
 def getAllLoginUsers(self):  
 list\_login = []  
 all\_login = self.db.get\_all\_login()  
 for login in all\_login:  
 list\_login.append(login['login'])  
 return list\_login  
  
 def CheckNewUserData(self, value\_dict):  
 if not (value\_dict['login'] in self.getAllLoginUsers()):  
 if value\_dict["is\_admin"] is None or value\_dict["is\_admin"] == "" or \  
 value\_dict["is\_block"] is None or value\_dict["is\_block"] == "" or \  
 value\_dict["is\_new"] is None or value\_dict["is\_new"] == "" or \  
 value\_dict["is\_pwd\_rules"] is None or value\_dict["is\_pwd\_rules"] == "" or \  
 value\_dict["login"] is None or value\_dict["login"] == "":  
 self.PwdMsg(tve.MSG\_EMPTY\_FIELD, tve.NAME\_WINDOW\_ERROR\_UPDATE\_DATA)  
 return False  
 else:  
 return True  
 else:  
 self.PwdMsg(tve.MSG\_NON\_UNIQUE\_LOGIN, tve.NAME\_WINDOW\_ERROR\_UPDATE\_DATA)  
 return False  
  
 def ColumnIntToStr(self, list\_item: list, reverse=False):  
 columns\_numerical = [0, 3, 4, 5, 6]  
 if reverse:  
 if list\_item[2] == "":  
 list\_item[2] = None  
 for i in columns\_numerical[-1:]:  
 list\_item[i] = int(list\_item[i])  
 else:  
 for i in columns\_numerical:  
 list\_item[i] = str(list\_item[i])  
 return list\_item  
  
 def deleteUser(self, event):  
 row = self.dataViewList.GetSelectedRow()  
 if row is wx.NOT\_FOUND:  
 self.PwdMsg(tve.MSG\_NO\_SELECT\_ROW, tve.NAME\_WINDOW\_ERROR\_DELETE\_USER)  
 else:  
 login = self.dataViewList.GetTextValue(row, 1)  
 if login == self.response['login']:  
 self.PwdMsg(tve.MSG\_NO\_DELETE\_SELF, tve.NAME\_WINDOW\_ERROR\_DELETE\_USER)  
 else:  
 self.db.delete\_user\_by\_login(login)  
 self.updateDataFromTable(None)