ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)
Кафедра безопасности информационных систем

Система электронного учета посещаемости занятий

РУКОВОДСТВО РАЗРАБОТЧИКА

Санкт-Петербург

2019

АННОТАЦИЯ

В данном руководстве приведено описание применения программного продукта «Система электронного учета посещаемости занятий» (далее – система).

Система предназначена для автоматизации учета и контроля посещаемости занятий студентами/учащимися образовательных учреждений.

В данном документе приведено подробное описание программного кода системы, а также основные требования к среде запуска программного продукта.

СОДЕРЖАНИЕ

1 НАЗНАЧЕНИЕ ПРОГРАММНОГО ПРОДУКТА

1.1 Назначение программного продукта

Назначением программного продукта «Система электронного учета посещаемости занятий» является автоматизация учета и контроля посещаемости занятий обучающимися.

Под обучающимися лица, посещающие учебные дисциплины образовательных учреждений среднего, среднего специального и высшего образования с целью получения свидетельства об окончании учебного заведения.

1.2 Возможности системы

Данный программный продукт позволяет осуществлять открытие, изменение, сохранение и хранение данных о посещаемости занятий различных групп. Также система способна создавать журналы посещаемости и хранить данные о расписании дисциплин преподавателя на семестр.

1.3 Основные характеристики системы

Система занятий позволяет осуществлять следующие операции:

- чтение файлов с расписанием, извлечение данных и запись их в базу данных;
- чтение файлов с составом группы, извлечение данных (ФИО студентов, номер группы) и запись их в базу данных;
- создание журнала посещаемости группы по запросу пользователя;
- регистрация студентов с помощью студенческих пропусков и считывателя карт;
- открытие, редактирование и сохранение журнала посещаемости группы.

1.4 Ограничения, накладываемые на область применения программы

Система предназначена для работы только в семействе операционных систем Microsoft Windows версии не ниже Windows XP.

2 УСЛОВИЯ ПРИМЕНЕНИЯ

2.1 Требования к техническим аппаратным средствам

Минимальный состав используемых технических (аппаратных) средств:

- оперативная память объемом не менее 2 Гб;

- место на диске не менее 10 Мб;

- как минимум, один USB-порт для подключения считывателя карт;

- считыватель карт типа Z-2 USB, Z-2 USB MF, Z-2 USB MFI, Z-2 EHR, Z-2 Base, RF-1996, Z-2MF CCID

2.2 Требования к программным средствам

Для функционирования программы необходимо следующее программное обеспечение:

- операционная система Windows XP/Vista/7/8/10 (32 или 64-битная версия);

- программа для работы с таблицами Microsoft Excel версии не ниже 5.0 (1995);

- драйвера для считывателя карт

- интерпретатор языка программирования Python версии не ниже 3.X.

2.3 Требования организационного характера

Для начала работы с программой необходимо предварительно подготовить файлы с расписанием и составом группы (формат файлов подробно описан в «Руководстве пользователя» глава 4 раздел 4.1).

3 ТРЕБОВАНИЯ К СИСТЕМЕ ЭЛЕКТРОННОГО УЧЕТА ПОСЕЩАЕМОСТИ

3.1 Требования к ПО

Система электронного учета посещаемости должна использовать программу для работы с электронными таблицами Microsoft Excel (далее – Excel) и программного пакета Microsoft Office версии не ниже 2010 года.

Для запуска программы с системой учета необходимо наличие на пользовательском компьютере интерпретатора языка Python версии не ниже 3.X.

3.2 Требования к интерфейсу

Интерфейс должен запускаться вместе с запуском программы и быть доступен на протяжении всего времени использования системы.

Интерфейс должен обеспечивать следующие типы взаимодействия с системой:

- выбор файла с данными о посещаемости группы для проведения занятия;
- выбор файла с посещаемостью группы для регистрации карт студентов;
- выбор файла с информацией о учебной группе для добавления данных в БД;
- выбор файла с информацией о расписании преподавателя для добавления данных в БД;
- возможность выхода из системы;

3.3 Описание работы интерфейса

Все приведенные выше типы взаимодействия должны происходить при событии «click», происходящем на соответствующих кнопках.

Кнопки типов взаимодействия (за исключением кнопки, инициирующей выход из системы) должны располагаться одна под другой в левой части окна системы.

Кнопка, инициирующая выход из системы должна располагаться в правом нижнем углу окна интерфейса.

Событие «click» кнопки, соответствующей выбору файла с посещаемостью группы для проведения занятия, инициирует открытие стандартного интерфейса файлового менеджера операционной системы для открытия файла. Выбранный файл открывается в программе Excel для дальнейшей работы с данными.

Событие «click» кнопки, соответствующей выбору файла с посещаемостью группы для регистрации карт, инициирует открытие стандартного интерфейса файлового менеджера операционной системы для открытия файла. Выбранный файл открывается в программе Excel для дальнейшей работы с данными.

Событие «click» кнопки, соответствующей выбору файла с данными об учебных группах, инициирует открытие стандартного интерфейса файлового менеджера операционной системы для открытия файла. Выбранный файл обрабатывается системой. Данные, извлеченные из файла, проверяются на совпадения с данными БД и, при отсутствии совпадений, вносятся в базу. При наличии совпадений, данные инициируют вызов окна с сообщением о наличии таких данных в БД и отбрасываются. При внесении данных в базу инициируется создания файла с посещаемостью группы и открытие стандартного окна файлового менеджера для сохранения файла вручную.

Событие «click» кнопки, соответствующей выбору файла с данными о расписании, инициирует открытие стандартного интерфейса файлового менеджера операционной системы для открытия файла. Выбранный файл обрабатывается системой. Данные, извлеченные из файла, проверяются на совпадения с данными БД и, при отсутствии совпадений, вносятся в базу. При наличии совпадений, данные инициируют вызов окна с сообщением о наличии таких данных в БД и отбрасываются.

Событие «click» кнопки, соответствующей выходу из системы, инициирует прекращение работы программы.

3.4 Описание стандартной работы с системой

Первая работа с системой должна включать в себя следующие этапы:

1. Внесение данных о расписании;
2. Внесение данных об учебной группе;
3. Сохранение файла с посещаемостью студентов учебной группы, данные о которой были внесены в БД на этапе выше;
4. Выбор файла с посещаемостью студентов учебной группы для регистрации карт;
5. Выбор файла с посещаемостью студентов учебной группы для проведения занятия.

При последующей работе с системой этапы 1-4 являются опциональными, т.е. выполняются при необходимости.

4 ВХОДНЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1 Входные данные

Входными данными являются:

- файл с расписанием на семестр;

- файл с составом группы;

- электронная карта студента.

4.2 Выходные данные

Выходными данными являются журналы с посещаемостью групп.

5 ПРОГРАММНЫЙ КОД

5.1 Описание базы данных

База данных системы учета посещаемости содержит 2 таблицы – СТУДЕНТ (далее – Студент) и ДИСЦИПЛИНА (далее – Дисциплина).

Таблица Студент содержит данные о фамилии, имени, отчестве и группе студента. Она необходима для создания файла с посещаемостью студентов.

Таблица Дисциплина содержит информацию о расписании преподавателя на семестр. Она также необходима для создания и сопровождения файла с посещаемостью студентов.

Описание структуры таблицы Студент представлено в таблице 1.

Описание структуры таблицы Дисциплина представлено в таблице 2.

Таблица 1 – Студент

Имя поля	Тип данных	Доп. характеристики	Назначение
Фамилия	Text	Первичный ключ	Фамилия студента
Имя	Text	Первичный ключ	Имя студента
Отчество	Text	not null	Отчество студента (если есть)
Группа	Text	Первичный ключ	Группа студента

Таблица 2 – Дисциплина

Имя поля	Тип данных	Доп. характеристики	Назначение
id_дис	Integer	Первичный ключ	Идентификационный номер занятия
Неделя	Integer	not null	Номер недели
Дата	Text	not null	Дата проведения занятия
Название	Text	not null	Название дисциплины
Тип	Text	not null	Тип дисциплины (лекция,

			практика, лабораторная работа)
Группа	Text	Первичный ключ	Номер группы, у которой проводится занятие
Время	Text	not null	Время проведения занятия (пара)
Аудитория	Text	not null	Аудитория, в которой проводится занятие

5.2 Модули

Все файлы (модули) системы учета посещаемости находятся в проекте Prog. Название и назначение модулей перечислены в таблице 3.

Таблица 3 – Файловая структура системы учета посещаемости

Название файла	Назначение файла
BD.db	База данных системы
choose_group.py	Скрипт, содержащий функцию открытия файла с группой для проведения занятия
create_group_file.py	Скрипт, содержащий функцию для создания журнала посещаемости
excel_group_parser.py	Скрипт, содержащий парсер excel-файла со списком группы и запрос с загрузкой данных в БД
excel_rasp_parser.py	Скрипт, содержащий парсер excel-файла с расписание преподавателя на семест и запрос с загрузкой данных в БД
fill_group_file.py	Скрипт, содержащий функцию для заполнения расписания занятий в журнале посещаемости
interface.py	Скрипт, содержащий функцию вызова основного окна (интерфейса) и функцию регистрацию карты
Shape.xlsx	Шаблон таблицы для создания файла с посещаемостью студентов группы

5.2.1 interface.py

Данный модуль отвечает за решение 2-х задач:

- вывод основного интерфейса программы;

- регистрация карт

```
import datetime, sys
from os import path, system
from tkinter import filedialog as fd
from PyQt5.QtCore import QCoreApplication
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton
from win32com.client import Dispatch
import choose_group, create_group_file, excel_group_parser, excel_rasp_parser


# Регистрация карт
def registration():
    reg_file = fd.askopenfilename()

    abs_path = path.abspath(reg_file)
```

```
    reg_excel = Dispatch("Excel.Application")
    reg_excel.Visible = True

    wb = reg_excel.Workbooks.Open(abs_path)
    appl = QApplication(sys.argv)
    sheet = wb.ActiveSheet

    system('PlaceCard.exe')

    def check_text():
        date = datetime.date.today().strftime('%d.%m')
        c = 5

        while sheet.Cells(c, 1).Value:
            c += 1

        for x in range(5, c):
            if int(appl.clipboard().text()[:3] + appl.clipboard().text()[4:])
== (sheet.Cells(x, 2).Value):
                sheet.Cells(x, date).Interior.ColorIndex = 6
                sheet.Cells(x, date).Value = 1
                wb.Save()

    def reg():
        print(appl.clipboard().text())

        reg_excel.ActiveCell.Value = (appl.clipboard().text()[:3] +
appl.clipboard().text()[4:])
        wb.Save()
        check_text()

    appl.clipboard().changed.connect(reg)

    print(appl.clipboard().text())


app = QApplication(sys.argv)

# Основное окно
start_window = QWidget()
start_window.resize(300, 250)
start_window.move(500, 200)
start_window.setWindowTitle('Учет посещаемости')

# Функциональные кнопки
group_choose_btn = QPushButton('Выбрать группу', start_window)
group_choose_btn.resize(group_choose_btn.sizeHint())
group_choose_btn.move(30, 10)
group_choose_btn.clicked.connect(choose_group.choose_group_file)

group_reg_btn = QPushButton('Регистрация карт', start_window)
group_reg_btn.resize(group_reg_btn.sizeHint())
group_reg_btn.move(30, 50)
group_reg_btn.clicked.connect(registration)

rasp_add_btn = QPushButton('Добавить расписание', start_window)
rasp_add_btn.resize(rasp_add_btn.sizeHint())
rasp_add_btn.move(30, 90)
rasp_add_btn.clicked.connect(excel_rasp_parser.rasp_excel_reader)

group_add_btn = QPushButton('Добавить группу', start_window)
group_add_btn.resize(group_add_btn.sizeHint())
```

```
group_add_btn.move(30, 130)
group_add_btn.clicked.connect(excel_group_parser.group_excel_reader)

create_group_file_btn = QPushButton('Создать журнал посещений', start_window)
create_group_file_btn.resize(create_group_file_btn.sizeHint())
create_group_file_btn.move(30, 170)
create_group_file_btn.clicked.connect(create_group_file.creation_group_file)

exit_btn = QPushButton('Выйти', start_window)
exit_btn.resize(exit_btn.sizeHint())
exit_btn.move(200, 190)
exit_btn.clicked.connect(QCoreApplication.instance().quit)

start_window.show()
sys.exit(app.exec_())
system('taskkill /IM EXCEL.EXE /F')
```

Для регистрации карты на вход функции подается файл, который пользователь открывает вручную с помощью стандартного окна открытия файлов файлового менеджера.

Сначала открывается журнал посещаемости группы, и затем программа «Place Card»для настройки и активации считывателя карт (процесс настройки и активации карты с помощью программы описан в документе «Инструкция по работе с Place Card_2.1.5.pdf»).

В открывшемся файле пользователь выбирает ячейку в столбце с номерами карт напротив ФИО студента. Студент прикладывает карту к считывателю и в выделенной ячейке отражается считываемый номер.

Для создания интерфейса приложения и работы с ним используются графические библиотеки PyQt5 (главное окно приложения) и Tkinter (стандартные окна открытия и сохранения файлов, системные сообщения о завершении загрузки файлов, сообщения об ошибке загрузки данных).

Для создания главного окна приложения вводится переменная start_window. Функции resize, move и setWindowTitle класса QWidget отвечают за размер окна, его расположение на экране и заголовок окна соответственно.

С помощью подкласса QPushButton класса QWidgets создаются функциональные кнопки для взаимодействия с приложением:

- group_choose_btn – кнопка для выбора журнала посещаемости группы из уже имеющихся;

- group_reg_btn – кнопка для выбора журнала посещаемости группы и регистрации карты студентов;

- rasp_add_btn – кнопка для добавления данных о расписании преподавателя на семестр;

- group_add_btn – кнопка для добавления данных о группе;

- create_group_file_btn – кнопка для создания журнала посещаемости;

- exit_btn – кнопка для выхода из приложения.

Функции resize и move отвечают за размер и расположение кнопки относительно окна приложения соответственно.

Метод clicked.connect нужен для привязывания действия, которое будет происходит по нажатию кнопки.

5.2.2 excel_rasp_parser.py

```
import sqlite3
from openpyxl import load_workbook
from tkinter import filedialog as fd, messagebox as mb
from os import system


def rasp_excel_reader():
    conn = sqlite3.connect('BD.db')
    cursor = conn.cursor()

    system('taskkill /IM EXCEL.EXE /F')

    timetable = list()  # расписание всех дисциплин

    loc_rasp = fd.askopenfilename()

    wb = load_workbook(loc_rasp)
    sheets_list = wb.sheetnames

    # проход по всем листам excel-файла
    for month in sheets_list:
        num_sheet_rasp = wb[month]
        max_row = num_sheet_rasp.max_row

        weeks = dict()  # словарь номеров недель (ключ) и их расположения по строкам в документе (нужно для парсинга пар)
        row_week = dict()  # словарь соответствия номера недели и максимального кол-ва строк в ней

        dis = []  # параметры дисциплины (id, неделя, дата, название, тип, и т.д.)
        id_dis = ''  # уникальный идентификатор пары
        lesson_num = ''  # номер пары (1-я, 2-я, и т.д.)
```

```
        # подсчет количества учебных недель в месяце
        for num_week in range(6, max_row + 1):
            if num_sheet_rasp.cell(row=num_week, column=1).value is not None:
                weeks[num_sheet_rasp.cell(row=num_week, column=1).value] =
num_week

        # вычисление максимального кол-ва строк для каждой недели
        for week in range(min(weeks.keys()), max(weeks.keys())+1):
            if week in range(1, 5):
                num_rows_in_week = weeks[week + 1] - weeks[week]
            elif month is 'Сентябрь' and week is 5:
                num_rows_in_week = max_row - weeks[week]

            row_week[week] = num_rows_in_week

        # Парсим расписание
        for num_week in weeks:
            for day in range(2, 9):
                # print(weeks[num week])
                for row in range(weeks[num_week], weeks[num_week] +
row_week[num_week]+1):
                    if type(num_sheet_rasp.cell(row=row, column=day).value)
is int:
                        date = str(num_sheet_rasp.cell(row=row,
column=day).value)
                    else:
                        if num_sheet_rasp.cell(row=row, column=day).value is
not None:
                            dis.append(num_sheet_rasp.cell(row=row,
column=day).value)
                        else:
                            if len(dis) != 0:
                                # Определение номера пары (1-я, 2-я, и т.д.)
                                if dis[3] == '09.00-10.35':
                                    lesson_num = '1'
                                elif dis[3] == '10.45-12.20':
                                    lesson_num = '2'
                                elif dis[3] == '13.00-14.35':
                                    lesson_num = '3'
                                elif dis[3] == '14.45-16.20':
                                    lesson_num = '4'
                                elif dis[3] == '16.30-18.05':
                                    lesson_num = '5'
                                elif dis[3] == '18.15-19.50':
                                    lesson_num = '6'
                                elif dis[3] == '20.00-21.35':
                                    lesson_num = '7'

                                id_dis = int(str(num_week) + str(date) +
lesson_num)
                                lesson_num = ''

                                dis[4].format('\\xa0', '')

                                dis.insert(0, id dis)
                                dis.insert(1, num_week)

                                if month == 'Сентябрь':
                                    date_month = str(date) + '.09'
                                    dis.insert(2, date_month)
                                elif month == 'Октябрь':
                                    date_month = str(date) + '.10'
```

```
                                    dis.insert(2, date_month)
                                elif month == 'Ноябрь':
                                    date_month = str(date) + '.11'
                                    dis.insert(2, date_month)
                                elif month == 'Декабрь':
                                    date_month = str(date) + '.12'
                                    dis.insert(2, date_month)
                                elif month == 'Февраль':
                                    date_month = str(date) + '.02'
                                    dis.insert(2, date_month)
                                elif month == 'Март':
                                    date_month = str(date) + '.03'
                                    dis.insert(2, date_month)
                                elif month == 'Апрель':
                                    date_month = str(date) + '.04'
                                    dis.insert(2, date_month)
                                elif month == 'Май':
                                    date_month = str(date) + '.05'
                                    dis.insert(2, date_month)
                                elif month == 'Июнь':
                                    date_month = str(date) + '.06'
                                    dis.insert(2, date_month)

                                timetable.append(dis)

                            id_dis = ''
                            dis = []

    # загрузка данных в БД
    try:
        for i in range(0, len(timetable)):
            cursor.execute("""INSERT INTO ДИСЦИПЛИНА VALUES
(?,?,?,?,?,?,?,?)""", timetable[i])
        conn.commit()
    except sqlite3.IntegrityError:
        mb.showerror('Внимание', 'Такое расписание уже есть в базе')

    mb.askokcancel('Загрузка расписания', 'Загрузка расписания завершена.')

    system('taskkill /IM EXCEL.EXE /F')
```

Осуществляется подключение к базе данных приложения (conn) и запуск «курсора» для возможности чтения и записи (cursor).

С помощью системной команды *taskkill* завершаются все процессы, запущенные Microsoft Excel.

Создается список timetable, в который будет заноситься вся информация о найденных занятиях.

На вход функции подается excel-файл (loc_rasp), который пользователь выбирает вручную с помощью стандартного окна открытия файлов.

Далее считывается название всех листов в файле и они заносятся в список sheets_list.

Организуется цикл, который проходит по все листам, указанным в списке sheets_list.

Выбирается лист с информацией о расписании на месяц (номер недели, день недели, дата, название дисциплины, тип, время, группы, аудитория) – num_sheet_rasp. Подсчитывается общее количество непустых строк в листе (max_row).

Создаются следующие словари:

1. weeks для записи номера недели (ключ) и его расположения в файле (значение);
2. row_week для записи номера недели (ключ) и максимального количества строк, относящихся к этой неделе (значение);

Создаются список dis для записи найденного занятия и его параметров (id, неделя, дата, название и т.д.), переменная id_dis для записи идентификатора конкретного занятия и переменная lesson_num для определения номера пары по времени начала и окончания занятия.

Далее подсчитывается количество недель на рассматриваемый месяц и записывается в словарь weeks в виде:

Номер недели : строка excel,

где *строка excel* – значение строки в excel-файле, с которой начинается неделя.

В следующем шаге, вычисляется, сколько строк приходится на каждую неделю (т.к. количество строк зависит от максимального количества пар в день на данной неделе) и записывается в словарь row_week в виде:

Номер недели : количество строк

Следующий шаг – обработка файла с расписанием.

Создается цикл, который проходит по всем значениям ключа в словаре weeks.

Создается вложенный цикл, проходящий по столбцам документа в диапазоне от 2 до 9 (дни недели).

Далее в цикле с днями недели создается еще вложенный цикл, который проходит по значениям в диапазоне от строки со значением недели, до максимальной строки, принадлежащей этой неделе. Если ячейка содержит значение типа int, то оно определяется как дата, если же значение типа str, то оно относится к занятиям.

Пока цикл не доходит до пустой строки, он считывает данные о занятии и заносит их в список dis.

Если цикл дошел до пустой строки, то переходим к следующему шагу – форматирование полученных данных о занятии и занесение их в список timetable.

По времени начала и окончания занятия определяется номер пары и значение записывается в переменную lesson_num.

Затем составляется идентификационные номер пары. Для этого значение даты и номера недели преобразуется в строковый тип данных и они складываются с переменной lesson_num. Полученная строка преобразуется в тип данных int и записывается в переменную id_dis.

Значение идентификационного номера и номера недели заносятся в список dis.

Дату проведения занятия необходимо привести к формату «*день.месяц*». Для этого с помощью названия просматриваемого листа определяется месяц. Далее значение переменной date преобразуется в строковый тип данных, складывается со значением месяца и в таком виде добавляется в список dis.

Таким образом список dis содержит идентификационные номер занятия, номер недели, дату проведения, название дисциплины, тип занятия (лекция, практические занятия, лабораторная работа), номер группы, у которой проводится занятие, время проведения занятия и аудитория, в которой проводится занятие.

Список dis добавляется в список timetable (т.е. переменная timetable является списком списков). Затем переменным id_dis и dis присваиваются пустые значения.

После выхода из всех циклов происходит попытка занесения данных переменной timetable в базу данных. Если таких данных в базе нет, то они загружаются, и по окончании загрузки выводится сообщение с соответствующим текстом.

Если же такие данные уже есть в базе данных, то выводится соответствующее сообщение.

Затем с помощью системной команды *taskkill* завершается процесс Excel и работа модуля заканчивается.

5.2.3 excel_group_parser.py

```
from pathlib import Path
import sqlite3
from openpyxl import load_workbook
from tkinter import filedialog as fd, messagebox as mb
from os import system


def group_excel_reader():
    conn = sqlite3.connect('BD.db')
    cursor = conn.cursor()

    system('taskkill /IM EXCEL.EXE /F')

    loc_group = fd.askopenfilename()

    wb = load_workbook(loc_group)

    num_sheet = wb['Лист1']
    max_row = num_sheet.max_row

    group_num = Path(loc_group).stem

    global students_list
    students_list = list()

    for row in range(1, max_row + 1):
        if num_sheet.cell(row=row, column=2) is not None:
            student = num_sheet.cell(row=row, column=2).value
            a = student.split(' ')

            # проверка на наличие отчества у студента
            if len(a) == 2:
                a.append('0')

            a.append(group_num)

            # Проверка на наличие записи в БД
```

```
            if cursor.execute("""SELECT * FROM СТУДЕНТ WHERE ФАМИЛИЯ==? AND
ИМЯ==? AND ОТЧЕСТВО==? AND ГРУППА==?""", a).fetchall() == []:
                cursor.execute("""INSERT INTO 'СТУДЕНТ' VALUES (?,?,?,?)""",
a)
                conn.commit()

            else:
                print('Такая запись уже существует')
                mb.showerror('Внимание', 'Группа уже есть в базе. Переход к
созданию файла посещений')

            students_list.append(a)
        else:
            None

    mb.askokcancel('Загрузка групп', 'Загрузка завершена. Следующий шаг -
создание файла посещений.')
    system('taskkill /IM EXCEL.EXE /F')
```

Осуществляется подключение к базе данных приложения (conn) и запуск «курсора» для возможности чтения и записи (cursor).

С помощью системной команды *taskkill* завершаются все процессы, запущенные Microsoft Excel.

На вход функции подается excel-файл (loc_group), который пользователь выбирает вручную с помощью стандартного окна открытия файлов.

Выбирается лист с информацией о составе группы (номер по порядку, ФИО студента) – num_sheet. Из названия файла выгружается номер группы для последующей загрузки в БД (group_num).

Подсчитывается общее количество непустых строк для определения количества студентов в группе (max_row). Затем создается цикл, где, проходя по каждой ячейке в столбце, содержащий ФИО, выгружается значение ячейки (student) и разбивается на фамилию, имя и отчество (a).

Проводится проверка на наличие отчества у студента. Если отчество нет, то добавляется значение ‘0’.

К списку ‘a’ добавляется значение group_num.

Перед занесением данных производится проверка на их наличие в БД. Если такие данные уже есть в базе, то выводится соответствующее сообщение. Если записи нет, то данные о студенте добавляются в базу.

По завершении загрузки выводится соответствующее сообщение.

С помощью системной команды *taskkill* завершается процесс Excel и работа заканчивается.

5.2.4 create_group_file.py

```
from os import system, path
from win32com.client import Dispatch
import win32com
from pathlib import Path
from tkinter import filedialog as fd, messagebox as mb
import sqlite3
import fill_group_file


def creation_group_file():
    conn = sqlite3.connect('BD.db')
    cursor = conn.cursor()

    # Прерывание всех процессов Micrisoft Excel
    system('taskkill /IM EXCEL.EXE /F')

    # Создание файла с посещаемостью группы
    file_path = fd.asksaveasfilename(defaultextension='.xls')

    system('copy Shape.xls {}'.format(path.abspath(file_path)))

    excel = win32com.client.Dispatch("Excel.Application")
    excel.Visible = 0

    wb = excel.Workbooks.Open(file_path)
    sheet = wb.ActiveSheet

    group_num = Path(file_path).stem

    sheet.Cells(1, 3).Value = group_num

    students_bd_list = cursor.execute("""SELECT * FROM СТУДЕНТ WHERE
ГРУППА==?""", (group_num, )).fetchall()
    i = 1

    # Заполнения столбца ФИО студента в таблице с посещаемостью
    for student in students_bd_list:
        sheet.Cells(i + 6, 3).Value = student[0] + ' ' + student[1] + ' ' +
student[2]
        i += 1
    wb.Save()

    mb.askokcancel('Создание файла посещений', 'Создание файла посещений
группы {group_num} завершено.')

    fill_group_file.fill_file_group()

    system('taskkill /IM EXCEL.EXE /F')
```

Осуществляется подключение к базе данных приложения (conn) и запуск «курсора» для возможности чтения и записи (cursor).

С помощью системной команды *taskkill* завершаются все процессы, запущенные Microsoft Excel.

Пользователь вручную выбирает путь сохранения файла и название (file_path). Далее помощью системной команды *copy* файл с шаблоном журнала посещаемости копируется в указанную директорию и переименовывается в соответствии с указанным названием.

Из указанного пути с помощью функции *stem* извлекается имя файла group_num (в имени файла должен быть указан номер группы).

Созданный журнал посещаемости открывается в фоновом режиме (excel).

В ячейку C3 записывается значение переменной group_num.

Из БД извлекаются данные о студентах указанной группы и помещаются в переменную students_bd_list в виде списка кортежей.

Организуется цикл который проходит по этому списку и заносит значения фамилии, имени и отчества студента в ячейки, начиная с C7.

После завершения записи данных в ячейки выводится соответствующее сообщение, а журнал посещаемости сохраняется.

Далее вызывается функция модуля fill_group_file.

После завершения работы функции с помощью системной команды *taskkill* завершаются все процессы, связанные с Excel и модуль заканчивает работу.

5.2.5 fill_group_file.py

```
from os import system, path
from win32com.client import Dispatch
import win32com
from pathlib import Path
from tkinter import filedialog as fd, messagebox as mb
import sqlite3


def fill_file_group():
    conn = sqlite3.connect('BD.db')
    cursor = conn.cursor()

    # Прерывание всех процессов Micrisoft Excel
    system('taskkill /IM EXCEL.EXE /F')

    mb.askokcancel('Журнал посещений', 'Выберите журнал посещений для
загрузки расписания группы.')

    open_group_file = fd.askopenfilename()
```

```
    group_num = Path(open_group_file).stem

    excel = win32com.client.Dispatch("Excel.Application")
    excel.Visible = 0

    wb = excel.Workbooks.Open(open_group_file)
    sheet = wb.ActiveSheet

    rasp_bd_list = cursor.execute("""SELECT НЕДЕЛЯ,ДАТА,НАЗВАНИЕ,ТИП,ВРЕМЯ
FROM ДИСЦИПЛИНА WHERE ГРУППА==?""",(group_num.replace('_', '-'),)).fetchall()
    i = 1
    num_dis = len(rasp_bd_list)

    name_discip = cursor.execute("""SELECT НАЗВАНИЕ FROM ДИСЦИПЛИНА WHERE
ГРУППА=='ИСТ-731' GROUP BY ГРУППА""").fetchall()

    sheet.Cells(1, 12).Value = name_discip

    for dis in rasp_bd_list:
        # номер учебной недели
        sheet.Cells(3, i + 3).Value = dis[0]

        # Дата проведения занятия
        sheet.Cells(4, i + 3).Value = dis[1]

        # Тип занятия (практика, лабораторная работа)
        if dis[3] == 'Практические занятия':
            sheet.Cells(5, i + 3).Value = 'пр'
        elif dis[3] == 'Лабораторная работа':
            sheet.Cells(5, i + 3).Value = 'лр'

        # Время проведения занятия (пара)
        sheet.Cells(6, i + 3).Value = dis[4]

        i += 1
    wb.Save()

    mb.askokcancel('Журнал посещений', 'Создание журнала посещений
завершено.')
    system('taskkill /IM EXCEL.EXE /F')
```

Осуществляется подключение к базе данных приложения (conn) и запуск «курсора» для возможности чтения и записи (cursor).

С помощью системной команды *taskkill* завершаются все процессы, запущенные Microsoft Excel.

Выводится сообщение о необходимости выбрать журнал загрузки.

Пользователь вручную выбирает журнал посещаемости (open_group_file) с помощью стандартного окна открытия файла.

Из директории указанного файла с помощью функции *stem* извлекается имя файла group_num (в имени файла должен быть указан номер группы).

Выбранный журнал посещаемости открывается в фоновом режиме (excel).

Из БД извлекаются данные о расписании указанной группы и помещаются в переменную rasp_bd_list в виде списка кортежей.

В переменную name_discip с помощью запроса в БД записывается название дисциплины, которую преподают указанной пользователем группе.

В ячейку L1 вводится значение переменной name_discip.

Далее организуется цикл, который проходится по всем кортежам списка rasp_bd_list.

В ячейки 3-ой строки, начиная со столбца C вводятся значения номера учебной недели.

В ячейки 4-ой строки, начиная со столбца C вводится дата проведения занятия.

В ячейки 5-ой строки, начиная со столбца C вводится сокращение типа занятия. Если занятие является лабораторной работой, то вводится «лр», если практическим занятием – «пр».

В ячейки 6-ой строки, начиная со столбца C вводится время проведения занятия.

После окончания цикла изменения в журнале сохраняется и выводится сообщение об окончании создания журнала посещения.

С помощью системной команды *taskkill* завершаются все процессы, связанные с Excel и модуль заканчивает работу.

5.2.6 choose_group.py

```
from os import path, system
from win32com.client import Dispatch
from tkinter import filedialog as fd


# выбор файла с посещаемостью группы
def choose_group_file():
    system('taskkill /IM EXCEL.EXE /F')
    file = fd.askopenfilename()

    global absPath
    absPath = path.abspath(file)

    global excel
```

```
    excel = Dispatch("Excel.Application")
    excel.Visible = True

    wb = excel.Workbooks.Open(absPath)
    sheet = excel.ActiveSheet

    system('PlaceCard.exe')
```

С помощью системной команды taskkill завершаются все процессы, запущенные Microsoft Excel.

На вход функции подается журнал посещаемости (file), который пользователь выбирает вручную с помощью стандартного окна открытия файлов из уже существующих.

Сначала открывается журнал посещаемости группы, и затем программа «Place Card»для настройки и активации считывателя карт (процесс настройки и активации карты с помощью программы описан в документе «Инструкция по работе с Place Card_2.1.5.pdf»).