**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**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту на тему:

**«Разработка системы электронного учета посещаемости занятий»**

По дисциплине «Веб-технологии и защита информации»

Выполнил ст. гр. ИСТ-731\_\_\_\_\_\_\_\_\_\_\_\_Маргун О.А.  
«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.   
Принял: ст.пр.каф. \_\_\_\_\_\_\_\_\_\_\_\_Евстигнеев В.А.  
«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Санкт-Петербург  
2019

РЕФЕРАТ

Отчет по курсовой работе содержит 28 страницы, 1 иллюстрацию, 3 таблицы, 7 использованных источников и 2 приложения. Оформлен согласно ГОСТ 7.32-2017[1].

Ключевые слова: СИСТЕМА, PYTHON, SQLITE, ПРИЛОЖЕНИЕ, УЧЕТ ПОСЕЩАЕМОСТИ.

Выполнен курсовой проект по разработке системы электронного учета посещаемости занятий. Проанализированы различные инструменты для разработки (языки программирования, СУБД). Рассмотрена актуальность применения данной системы в образовательных учреждениях.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc28035717)

[ПОСТАНОВКА ЗАДАЧИ 5](#_Toc28035718)

[ТРЕБОВАНИЯ К СИСТЕМЕ УЧЕТА ПОСЕЩАЕМОСТИ 6](#_Toc28035719)

[ВЫБОР МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 7](#_Toc28035720)

[1 Язык программирования 7](#_Toc28035721)

[2 Язык описания базы данных 8](#_Toc28035722)

[РЕШЕНИЕ ЗАДАЧИ 10](#_Toc28035723)

[1 Архитектура системы учета посещаемости 10](#_Toc28035724)

[1.1 Описание базы данных 10](#_Toc28035725)

[1.2 Интерфейс системы учета посещаемости 10](#_Toc28035726)

[1.3 Разработка кода системы 11](#_Toc28035727)

[1.4 Инсталляция системы 11](#_Toc28035728)

[2 Проектная документация 11](#_Toc28035729)

[2.1 Описание БД 11](#_Toc28035730)

[2.2 Файловая структура проекта 12](#_Toc28035731)

[2.3 Алгоритм работы с системой 13](#_Toc28035732)

[ЗАКЛЮЧЕНИЕ 15](#_Toc28035733)

[СПИСОК ИПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc28035734)

[ПРИЛОЖЕНИЯ 17](#_Toc28035735)

[Приложение А 17](#_Toc28035736)

[Приложение Б 20](#_Toc28035737)

# ВВЕДЕНИЕ

Контроль посещаемости студентами занятий является неотъемлемой частью учебного процесса и часто занимает много времени. Особенно это заметно на лекционных занятиях, которые проводятся для нескольких групп.

Рутинность процесса контроля посещаемости привела к необходимости сокращения временных затрат на его проведение или, иными словами, к автоматизации.

В данный момент большая часть преподавателей использует бумажные носители (ведомости, бланки, блокноты) для контроля посещаемости, что неудобно при наличии большого числа групп, а также повышает вероятность ошибки при выставлении посещения.

Идея системы электронного учета посещаемости не является новой и уже нашла свое применение в различных компаниях для учета времени, в которое приходят сотрудники и т.д. Подобные системы существуют и для учебных заведений, но основной их недостаток – необходимость предварительной установки дополнительного ПО (самой программы для учета посещаемости, например).

# ПОСТАНОВКА ЗАДАЧИ

Цель курсового проекта – создание системы электронного учета посещаемости студентов.

В рамках курсового проекта планируется реализовать следующие задачи:

- разработка системы ввода данных для системы электронного учета посещаемости;

- разработка системы вывода данных для системы электронного учета посещаемости;

- разработка шаблона документа для формирования журнала посещаемости группы в системе электронного учета посещаемости.

- построение базы данных для хранения информации о расписании и группах;

- анализ дальнейшей разработки проекта;

- написание руководства пользователя и руководства разработчика для удобной работы с системой и возможности ее расширения.

# ТРЕБОВАНИЯ К СИСТЕМЕ УЧЕТА ПОСЕЩАЕМОСТИ

Система электронного учета посещаемости должна отвечать следующим требованиям:

- наличие веб-интерфейса для возможности выбора типа взаимодействия с сиcтемой;

- возможность добавления в БД данных об учебных группах;

- возможность добавления в БД данных о дисциплинах и занятиях на семестр;

- возможность регистрации электронного студенческого пропуска;

- возможность отметки посещаемости студентов.

Более подробные требования к работе системы изложены в приложении А.

# ВЫБОР МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

Система является desktop-приложением с интегрированной базой данных. Она должна быть достаточно легкой, чтобы не загружать оперативную память и не мешать работе сторонних приложений на пользовательском компьютере.

## 1 Язык программирования

Для разработки desktop-приложений на момент написания курсового проекта наиболее популярны языки программирования Java, C++, Python.[2]

Сравнение языков программирования показывает, что выбор сильно зависит от масштаба и типа задачи [3][4].

Компилируемый язык программирования Java является достаточно мощным инструментом разработки с высокой скоростью работы, устоявшейся философией, большим количеством различных библиотек и мануалов от самых разных разработчиков. Требует установки JVM (Java Virtual Machine) и JDK (Java Development Kit) для исполнения программ. Хорошо подходит для разработки масштабных desktop-приложений.

Компилируемый язык программирования C++ является одним из наиболее популярных языков для разработки. Имеет богатую стандартную библиотеку, сочетает свойства высокоуровневых и низкоуровневых языков, большое внимание уделено поддержке ООП. Эффективно работает с памятью и выигрывает Java в скорости выполнения программ. Область применения включает в себя создание операционных систем, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов и игр [5].

Интерпретируемый язык программирования Python обладает легким синтаксисом, большим количеством библиотек, динамической типизацией и низким порогом вхождения. Python реализован под всеми распространенными операционными системами и на множестве архитектур. Так как используется интерпретатор, Python проигрывает в скорости исполнения программ, но компенсирует это более малым объемом кода (по сравнению с Java и C++) и большим количеством различных сред разработки и сервисов [6].

Так как система должна быть в первую очередь легкой и не требовать большое количество предустановленного ПО, для разработки был выбран язык программирования Python, как отвечающий требованиям выше.

## 2 Язык описания базы данных

Любое приложение может использовать, как локальную БД или сетевую БД, так и обе одновременно. Данная система должна хранить достаточно малое количество данных, не предназначенных для совместного использования. Таким образом, система будет использовать локальную БД для хранения данных.

На данный момент есть огромное количество систем управления БД, которые применяются для описания БД и создания запросов. Более популярны MySQL, PostgerSQL и SQLite [7].

MySQL является одной из самых распространенных СУБД, хорошо работает с различными сайтами и веб-приложениями, предлагает много инструментов для разработки. Однако, по сравнению с остальными СУБД уступает в надежности и скорости разработки, а также не всегда соответствует стандартам SQL.

PostgerSQL наиболее профессиональная СУБД, поддерживающая объектно-ориентированный или реляционный подход к БД. Является очень производительной, с возможностью расширения функционала за счет сохранения собственных процедур. Недостатком можно назвать низкую производительность.

SQLite является легко встраиваемой в приложения БД, с высокой скоростью исполнения. Имеет файловую структуру, что позволяет легко ее переносить на разные машины. Из недостатков - ограниченная производительность и слабая поддержка многопользовательских приложений.

Система учета посещаемости не требует большой производительности и масштабируемости БД, а также должна использовать локальную БД. Приведенным выше требованиям в полной мере соответствует SQLite.

# РЕШЕНИЕ ЗАДАЧИ

## 1 Архитектура системы учета посещаемости

### 1.1 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. Она также необходима для создания и сопровождения файла с посещаемостью студентов.

Полное описание таблиц базы данных приводится в разделе «Документация».

### 1.2 Интерфейс системы учета посещаемости

Внешний вид интерфейса выполнен в минималистическом стиле и содержит только необходимые кнопки и название системы. Внешний вид модуля представлен на рисунке 1.

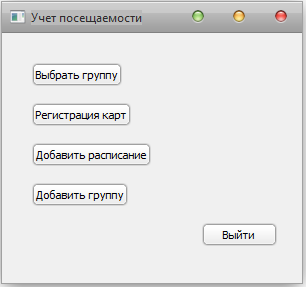


Рисунок 1. Интерфейс системы учета посещаемости

### 1.3 Разработка кода системы

Логика работы системы реализована на языке программирования Python. В качестве примера приводится код модуля choose\_group.py.

В данном модуле инициируется открытие стандартного окна файлового менеджера для открытия файла с посещаемостью студентов группы для проведения занятия и последующий запуск Excel для запуска файла.

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

### 1.4 Инсталляция системы

Для инсталляции системы требуется создание рабочей директории на пользовательском компьютере.

В рабочую директорию копируются файлы с программным кодом системы, документация, папка с программой «Place Card» и тестовые данные.

Запуск системы осуществляется с помощью запуска с помощью интерпретатора Python файла interface.py (указывается абсолютный путь к файлу) из-под командной строки.

## 2 Проектная документация

### 2.1 Описание БД

База данных системы учета посещаемости содержит 2 таблицы:

- Студент (в БД – СТУДЕНТ), хранящая данные о студентах:

- Дисциплина (в БД – ДИСЦИПЛИНА), хранящая данные о расписании преподавателя на семестр.

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 занятия (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 |

### 2.2 Файловая структура проекта

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 |

Продолжение таблицы 3

|  |  |
| --- | --- |
| interface.py | Скрипт, содержащий функцию вызова основного окна (интерфейса) и функцию регистрацию карты |
| Shape.xlsx | Шаблон таблицы для создания файла с посещаемостью студентов группы |

### 2.3 Алгоритм работы с системой

При первом входе в систему необходимо выполнить описанные ниже действия.

Нажать кнопку «Добавить расписание» и выбрать excel-файл, содержащий расписание, нажать «Открыть» и дождаться занесения данных в базу.

Вернуться к главному окну, нажать кнопку «Добавить группу» и выбрать excel-файл, содержащий состав группы, нажать «Открыть».

В открывшемся окне «Сохранение» выбрать путь сохранения файла с посещаемостью студентов только что загруженной группы и нажать «Сохранить».

Вернуться к главному окну, нажать кнопку «Регистрация карт» и в открывшемся окне выбрать excel-файл с посещаемостью студентов группы, нажать «Открыть» и дождаться запуска Excel.

После запуска журнала посещаемости дождаться запуска программы «Place Card». Подключить считыватель, произвести настройку и активацию считывателя с помощью «Place Card» (способы настройки и активации считывателя можно посмотреть в «Инструкции к работе с Place Card\_2.1.5.pdf»).

Регистрация карт осуществляется путем выделения ячейки напротив нужной фамилии и прикладывание карты студента к считывателю карт.

После завершения процедуры регистрации карт, закрыть Excel, вернуться к главному окну и нажать на кнопку «Выбрать группу». В открывшемся окне выбрать файл с посещаемостью группы, нажать «Открыть» и дождаться запуска Excel.

После окончания работы с файлами, вернуться к главному окну и нажать кнопку «Выход».

При повторном входе в систему шаги с добавлением расписания, добавлением групп и регистрацией карт выполнять при необходимости.

# ЗАКЛЮЧЕНИЕ

В ходе работы над курсовым проектом было составлено техническое задание на разработку системы электронного учета посещаемости занятий, как самостоятельного desktop-приложения. Проанализированы существующие инструменты разработки desktop-приложений, рассмотрены различные СУБД и выбран оптимальный стек технологий для разработки системы. Разработан функционал системы электронного учета посещаемости. Спроектирована база данных для хранения информации о студентах и расписании преподавателя на семестр. Разработано руководство пользователя с подробным описанием работы с системой и руководство разработчика с подробным описанием работы всех модулей кода системы.

# СПИСОК ИПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 7.32-2017 СИБИД. Отчет о научно-исследовательской работе. Структура и правила оформления [электронный ресурс], URL: http://www.tsu.ru/upload/medialibrary/235/gost\_7.32\_2017.pdf;
2. 10 языков программирования для изучения в 2018 году [электронный источник], URL: https://icanchoose.ru/blog/10-yazykov-programmirovaniya-dlya-izucheniya-v-2018-godu/;
3. Что лучше: Java, С++ или Python [электронный источник], URL: https://upread.ru/blog/articles-it/java-c-ili-python;
4. Python и другие языки программирования: сходства и отличия [электронный источник], URL: https://techrocks.ru/2017/09/18/python-and-other-programming-languages-similarities-and-differences/;
5. С++ – Википедия. Свободная энциклопедия [электронный источник], URL: https://ru.wikipedia.org/wiki/C%2B%2B;
6. Илья Бубнов. 5 причин, почему Python все еще крут [электронный источник], URL: https://geekbrains.ru/posts/why\_love\_python;
7. SQLite vs MySQL vs PostgreSQL: сравнение систем управления базами данных [электронный источник], URL: https://devacademy.ru/article/sqlite-vs-mysql-vs-postgresql/

# ПРИЛОЖЕНИЯ

## Приложение А

**Требования к системе электронного учета посещаемости**

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.

2 Требования к интерфейсу

Главное окно должно 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 об учебной группе для добавления данных в БД;

- выбор файла с информацией о расписании преподавателя для добавления данных в БД;

- возможность выхода из системы;

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 главного окна.

Событие «click» кнопки, соответствующей выбору файла с посещаемостью группы для проведения занятия, должно инициировать открытие стандартного окна файлового менеджера операционной системы для открытия файла. Выбранный файл должен открываться в программе Excel для дальнейшей работы с данными.

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

Событие «click» кнопки, соответствующей выбору файла с данными об учебных группах, должно инициировать открытие стандартного интерфейса файлового менеджера операционной системы для открытия файла. Выбранный файл должен обрабатываться системой. Данные, извлеченные из файла, должны проверяться на совпадения с данными БД и, при отсутствии совпадений, должны вноситься в базу. При наличии совпадений, данные должны инициировать вызов окна с сообщением о наличии таких данных в БД и отбрасываться. При внесении данных в базу должно инициироваться создание файла с посещаемостью группы и открытие стандартного окна файлового менеджера для сохранения файла вручную.

Событие «click» кнопки, соответствующей выбору файла с данными о расписании, должно инициировать открытие стандартного интерфейса файлового менеджера операционной системы для открытия файла. Выбранный файл должен обрабатываться системой. Данные, извлеченные из файла, должны проверяться на совпадения с данными БД и, при отсутствии совпадений, вноситься в базу. При наличии совпадений, данные должны инициировать вызов окна с сообщением о наличии таких данных в БД и отбрасываться.

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

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.

## Приложение Б

**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'**)

**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'**)

**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'**)

**Interface.py**

**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'**)

**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'**)

**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'**)