МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
Заключительный этап
Продуктовый сектор
Раздел “ИТ”
Практика заключительного этапа

Условие

Задание доступно в электронном виде по ссылке: dt.miet.ru/ppo_it

Модест - студент третьего курса Международного Института Эстетических Технологий. Модест с детства играет на терменвоксе. Он небезосновательно считает, что его талант способен украсить любое творческое студенческое объединение. К сожалению, не все однокурсники разделяют это мнение.

Модест знает, что традиционно, в преддверии праздников, студенты в общежитии собираются на репетиции перед концертами студенческой самодеятельности. Стоя перед общежитием, Модест видит, в каких окнах горит свет, а значит полным ходом идет репетиция.

Помогите Модесту определить количество и номера комнат, в которых проходят репетиции, для того, чтобы он смог к ним присоединиться со своим терменвоксом.

Техническое задание

Необходимо разработать программный продукт, позволяющий демонстрировать, в каких комнатах в общежитии репетируют студенты. Входные данные, получаемые посредством API, представляют собой информацию:

- количество комнат на этаже;
- количество окон в комнатах;
- информация о том, в каких окнах горит свет;
- дата получения информации.

Также API предоставляет возможность проверки, правильно ли определены комнаты.

Необходимо разработать алгоритм, позволяющий найти номера комнат, в которых горит свет.

Пользовательский интерфейс должен предоставлять следующие возможности:

- графическое отображение информации о том, в каких окнах горит свет;
- графическое отображение информации о том, в каких окнах горит свет, по отдельности по каждой дате;
- отображение по запросу пользователя номеров комнат, в которых репетируют студенты;
- отображение того, правильно ли определены искомые комнаты;

- позволять запрашивать данные по заданной пользователем датой с последующим отображением;
- введение пользователем информации о количестве комнат на этаже, количестве окон в комнате, и информации о том, в каких окнах горит свет с последующим графическим отображением и поиском комнат, в которых репетируют студенты.

Входные данные

Получаемые данные разделены по датам и включают в себя информацию о количестве комнат на этаже, количестве окон в комнатах и информации о том, в каких окнах горит свет.

Данные доступны по адресу: https://olimp.miet.ru/ppo_it_final. В заголовках запроса необходимо указывать логин для Яндекс.контест одного (любого) из участников команды, формата: *ppo_00_00000*. Логин участника можно получить в ECP.

Формат указания заголовка: `{X-Auth-Token: <login>}` , например:

```
{X-Auth-Token: ppo_12_000000}
```

Описание API:

`GET /date` - получение списка дат, по которым есть данные.

Общий формат ответа сервера:

```
{
    "message": [
        "<date_1:string>",
        "<date_2:string>",
        "...",
        "<date_n:stting>"
    ]
}
```

`GET ?day=<day>&month=<month>&year=<year>` - получение данных по определенной дате.

Общий формат ответа сервера:

```
{
    "message": {
        "date": {
            "data": "<timestamp:int>",
            "description": "<description:string>"
```

```
},
"rooms_count": {
        "data": "<rooms_count:int>",
        "description": "<description:string>"
},
"windows_for_room": {
        "data": [
                "<windows_for_room_1:int>",
                "<windows_for_room_2:int>",
                "..."
                "<windows_for_room_n:int>"
        ],
        "description": "<description:string>"
},
"windows": {
        "data": {
                "floor_1": [
                        "<light_1:bool>",
                        "<light_2:bool>",
                        "..."
                        "<light_m:bool>"
                ],
                "floor_2": [
                        "<light_1:bool>",
                        "<light_2:bool>",
                        "..."
                        "<light_m:bool>"
                ],
                "floor_3": [
                        "<light_1:bool>",
                        "<light_2:bool>",
                        "..."
                        "<light_m:bool>"
                ],
                "floor_4": [
                        "<light_1:bool>",
                        "<light_2:bool>",
```

```
                                        "..."
                                        "<light_m:bool>"
                                ]
                        },
                        "description":  "<description:string>"
                }
        }
}
```

`POST` запрос с полезной нагрузкой (payload) в виде количества комнат и номеров комнат в формате:

```
{
        "data":  {
                "count":  "<count:int>",
                "rooms":  [
                        "<room_number_1:int>",
                        "<room_number_2:int>",
                        "..."
                        "<room_number_count:int>"
                ]
        },
        "date":  "<date:str>"
}
```

Пример запроса:

```
{
        "data":  {
                "count":  4,
                "rooms":  [
                        3,
                        5,
                        9,
                        10
                ]
        },
        "date":  "23-05-21"
```

```
}
```

В случае правильного формата запроса, ответ может принимать два значения, а код ответа 200:

```
{
    "message": "correct answer"
}
```

или

```
{
    "message": "wrong answer"
}
```

В случае неверного формата данных или заголовка POST запроса, код ответа примет значение 400.

При отправке POST запроса необходимо указывать в заголовке в поле `Content-Type` значение `application/json`.

Получаемые данные необходимо хранить в системе управления базами данных (СУБД). Выбор СУБД не регламентируется.

Рекомендации к выполнению

Программный продукт должен работать либо на двух и более платформах, либо иметь веб-интерфейс.

Рекомендуется разделить программный продукт на модули: back-end и front-end.

Разработку рекомендуется вести с помощью системы контроля версий git.

Рекомендуется использовать unit-тестирование при разработке продукта.

Схематичное изображение примера пользовательского интерфейса для набора данных:

```
{
    "date": {
        "data": 1674594000,
        "description": "Татьянин день"
    },
    "rooms_count": {
        "data": 3,
```

```
        "description":  "Количество комнат на этаже"
    },
    "windows_for_room":  {
        "data":  [
            3,
            2,
            1
        ],
        "description":  "Количество окон в каждой из комнат на этаже слева направо"
    },
    "windows":  {
        "data":  {
            "floor_1":  [
                false,
                true,
                false,
                true,
                false,
                false
            ],
            "floor_2":  [
                true,
                false,
                true,
                false,
                false,
                true
            ],
            "floor_3":  [
                false,
                false,
                true,
                false,
                true,
                false
            ],
            "floor_4":  [
```

```
                    false,
                    false,
                    false,
                    true,
                    false,
                    true
                ]
            },
            "description":  "Окна по этажам, в которых горит свет"
        }
}
```

представлено на рисунке ниже:

25 января 2023 года								
10	10	10	**11**	**11**	**12**	Входные данные:	Количество комнат на этаже:	3
7	7	**7**	8	**8**	9		Окна на этаже:	3 2 1
4	**4**	**4**	5	5	**6**	Ответ:	Количество комнат:	8
1	**1**	**1**	**2**	**2**	3		Номера комнат:	1, 2, 4, 6, 7, 8, 11, 12

Регламент испытаний

1. Запуск программного продукта.
2. Отображение окон по одному из дней.
3. Отображение всех дней, по которым получена информация от API.
4. Отображение окон с отметкой, какие горят, какие нет, по одному дню.
5. Отображение окон с отметкой, какие горят, какие нет, по всем дням, по которым получена информация от API.
6. Отображение количества комнат и их номеров по всем дням, с отметкой о том, корректно ли найдены занятые комнате.
7. Введение входных данных жюри с последующим отображением информации о горящих окнах, количества комнат и их номеров;

8. Ведение даты, по которой ранее не получались данные и отображение информации о горящих окнах, количестве комнат и их номерах;
9. Демонстрация работы СУБД, в том числе, включающая в себя проверку персистентого хранения данных.
10. Участники демонстрируют работу unit-тестов.

Загрузка решений

По окончании защиты необходимо загрузить все исходные файлы проекта:

- архив с репозиторием/программный код;
- файлы с данными;
- файлы БД (дамп БД);
- снимки экрана

в облачную папку в соответствии с наименованием вашей команды.

Адрес облачной папки:
https://cloud.predprof.olimpiada.ru/index.php/s/dfPJPW73PJYr2M3

Этапы решения

Этап «Работа с данными»

Перед началом разработки алгоритма поиска необходимых комнат, требуется изучить протокол получения данных и реализовать клиентскую часть, работающую по протоколу HTTP для получения данных от API. Результатом станут полученные данные по всем дням в формате JSON. Данные необходимо обработать и сохранить в системе управления базами данных в виде, подходящем для дальней работы. Отсутствие в решении СУБД не даст получить полный балл за этап.

Результаты выполнения данного этапа необходимо использовать при выполнении этапа «Алгоритм». При этом, этапы «Пользовательский интерфейс» и «Программный код» могут реализовываться параллельно.

Для успешного выполнения данного этапа необходимо

Знать: основы работы с протоколом HTTP, структуру хранения данных в JSON, форматы хранения дат.

Уметь: использовать модули, библиотеки, программы позволяющие осуществлять взаимодействие по протоколу HTTP, проектировать базы данных.

Этап «Пользовательский интерфейс»

Разработка пользовательского интерфейса важный этап. С помощью него пользователь может взаимодействовать с данными и результатами их обработки. Разработка его может вестись параллельно остальным этапам, на основе данных из примера. Результатом выполнения станет возможность просматривать информацию о свободных комнатах по всем дням, просматривать правильно ли определены комнаты, возможность вводить пользовательские данные, для проверки правильности алгоритма поиска. Полный балл при оценке возможно получить при разработке графического пользовательского интерфейса. Также, допустимо реализовать консольный интерфейс. Это даст возможность получить не полный балл за данный этап.

Полное отсутствие интерфейса не даст возможность продемонстрировать работоспособность всего решения.

Для успешного выполнения данного этапа необходимо

Знать: основы проектирования графических пользовательских интерфейсов.

Уметь: применять модули, библиотеки, программы позволяющие реализовывать графический интерфейс, проектировать модульные приложения и осуществлять связь между модулями.

Этап «Алгоритм»

После получения данных от API требуется разработать алгоритм поиска необходимых по условию комнат. Обработанные данные, сохраненные в СУБД, представляют собой массив булевых значений, обозначающих горящие или темные окна в комнате, информацию о количестве комнат на этаже и количестве окон в каждой из комнат. При переборе окон, необходимо определить, есть ли хотя бы одно горящее окно в комнате. Если есть, то учесть номер этой комнаты для дальнейшего отображения и подсчета количества. При оценке отдельно оценивается набор данных по каждому дню.

Пользовательский интерфейс может реализовываться параллельно, с использованием данных из примера. Этап «Программный код» также не зависит от этапа «Алгоритм».

Для успешного выполнения данного этапа необходимо

Знать: основы работы со структурами данных.

Уметь: разрабатывать алгоритмы поиска.

Этап Программный код

При оценке решения на данном этапе оценивается соблюдение единой стилистики написания программного кода. Также, оценивается наличие системы управления версиями, например git. Также, при оценке учитывается наличие unit-тестов программного продукта.

Для успешного выполнения данного этапа необходимо

Знать: основы оформления программного кода.

Уметь: применять системы управления версиями программного кода.

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
Заключительный этап
Продуктовый сектор
Раздел “ИТ”
Практика заключительного этапа

Оценка

Данные для проверки Жюри:

Количество комнат на этаже: 2

Количество окон в каждой из комнат на этаже слева направо: **1 1**

Окна по этажам, в которых горит свет:

True False

False True

Работа с данными (накопительно)

Уровень	Критерии	Максимальный балл 50
1	Данные от API не разобраны	0
2	Начата работа с данными, но не правильно интерпретированы данные	20
3	Данные получены, разобраны, хранятся не в СУБД	30
4	Данные получены, разобраны, хранятся в СУБД	50

Пользовательский интерфейс (накопительно)

Уровень	Критерии	Максимальный балл 120
1	Интерфейс пользователя отсутствует (отсутствуют возможность посмотреть на окна).	0
2	Интерфейс реализован в командной строке (для инициализации, перехода по дням используются отдельные исполняемые файлы). Присутствуют не все возможности по просмотру информации о	20

Уровень	Критерии	Максимальный балл 120
	состоянии окон	
3	Интерфейс реализован в командной строке (для инициализации, перехода по дням используются отдельные исполняемые файлы) Присутствуют возможности по просмотру состояний окон.	40
4	Интерфейс реализован в командной строке (для инициализации, перехода по дням используются отдельные исполняемые файлы) Присутствуют возможности по просмотру состояний окон, а также по вводу пользовательских данных, отображения выходных данных (количество комнат, их номера)	60
5	Разработан графический интерфейс на выбранной платформе. Присутствуют не все возможности по просмотру состояния окон или нельзя переключаться между днями	80
6	Разработан графический интерфейс на выбранной платформе. Присутствуют все возможности по просмотру состояния окон или нельзя переключаться между днями, но нет возможности ввода пользовательских данных	100
7	Разработан графический интерфейс на выбранной платформе. Присутствуют все возможности по просмотру состояния окон или нельзя переключаться между днями в том числе возможность ввода пользовательских данных	120

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
Заключительный этап
Продуктовый сектор
Раздел “ИТ”
Практика заключительного этапа

Алгоритм (суммарно)

Уровень	Критерии	Максимальный балл 90	Комментарий
1	Алгоритм неработоспособен/программный код отсутствует.	0	
2	Алгоритм работает, проверяется каждый тест Если не реализован пользовательский интерфейс, то можно оценить без него	90	по 10 баллов за каждый тест

Программный код (суммарно)

Уровень	Критерии	Максимальный балл 40	Комментарий
1	Код написан без соблюдения стилистики, имена переменных не несут смысловой нагрузки, код в целом трудно читаем.	0	
2	Код читаем, разработчики в целом придерживаются одного стиля.	20	
3	Разработка велась с помощью git	20	

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
Заключительный этап
Продуктовый сектор
Раздел “ИТ”
Практика заключительного этапа

Выполнение технического задания(суммарно)

Уровень	Критерии	Максимальный балл 300	Комментарий
1	Произведен запуск программного продукта	20	
2	Отображение окон по одному из дней.	20	
3	Отображение всех дней, по которым получена информация от API.	20	
4	Отображение окон с отметкой, какие горят, какие нет, по одному дню.	20	
5	Отображение окон с отметкой, какие горят, какие нет, по всем дням, по которым получена информация от API.	40	20 если окна некорректно “подсвечиваются”
6	Отображение количества комнат и их номеров по всем дням, по которым получена информация от API.	40	20 данные отображаются не по воздействую пользователя
7	“Подсветка корректности полученного ответа”	40	
8	Введение входных данных жюри с последующим отображением информации о горящих окнах, количество комнат и их номера;	40	20 если данные принимаются, но по ним нет отображения

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
Заключительный этап
Продуктовый сектор
Раздел “ИТ”
Практика заключительного этапа

Уровень	Критерии	Максимальный балл 300	Комментарий
9	Введение жюри даты, по которой ранее не получилось данные, отображение информации о горящих окнах, количества комнат и их номеров;	40	20 если дата принимается, но по ней нет отображения
10	Предоставлены unit-тесты.	20	

МОСКОВСКАЯ ПРЕДПРОФЕССИОНАЛЬНАЯ ОЛИМПИАДА ШКОЛЬНИКОВ
Заключительный этап
Продуктовый сектор
Раздел “ИТ”
Практика заключительного этапа

Решение задачи

File: predprof2024-main

```
predprof2024-main/
|-- .gitignore
|-- LICENSE
|-- README.md
|-- app/
|   |-- app.py
|   |-- data.db
|   |-- static/
|   |   `-- css/
|   |       `-- main.css
|   `-- templates/
|       |-- index.html
|       |-- list.html
|       `-- vvod.html
|-- requirements.txt
`-- src/
    |-- data.db
    |-- database.py
    |-- logic.py
    |-- main.py
    `-- test.py

5 directories, 15 files
```

Рисунок 1: Структура решения

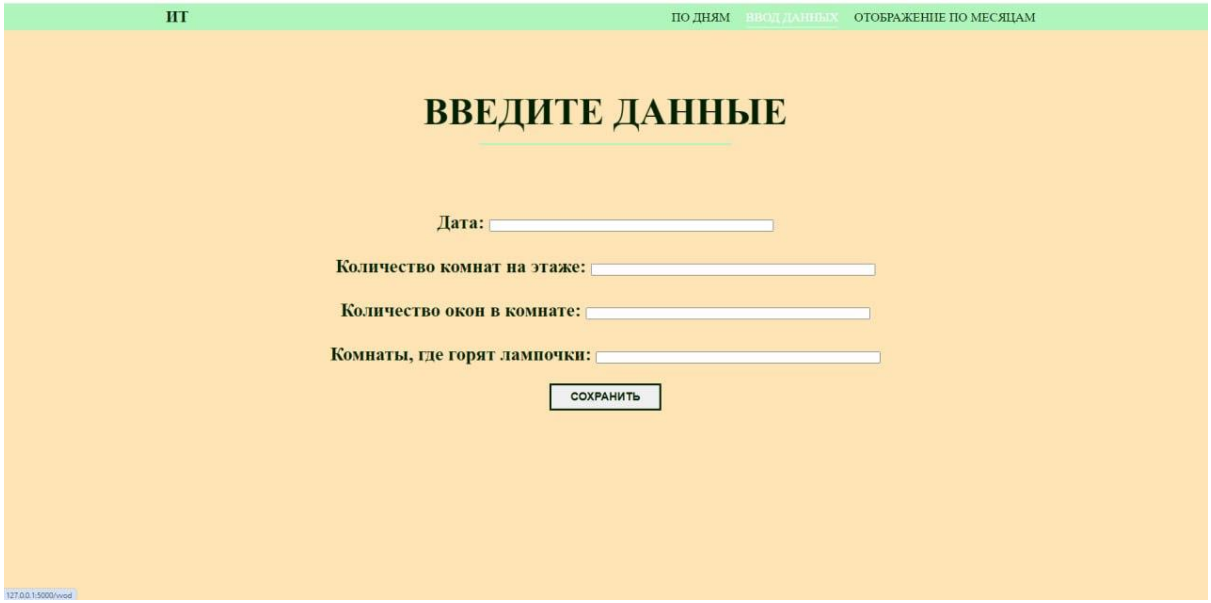


Рисунок 2: Снимок экрана разработанного программного продукта

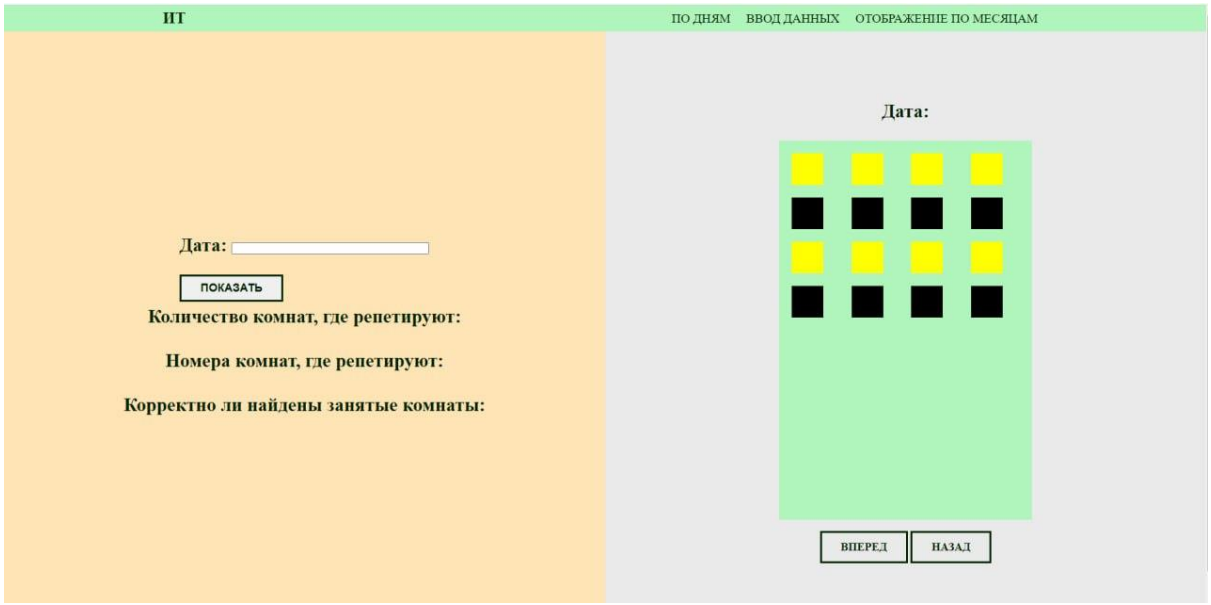
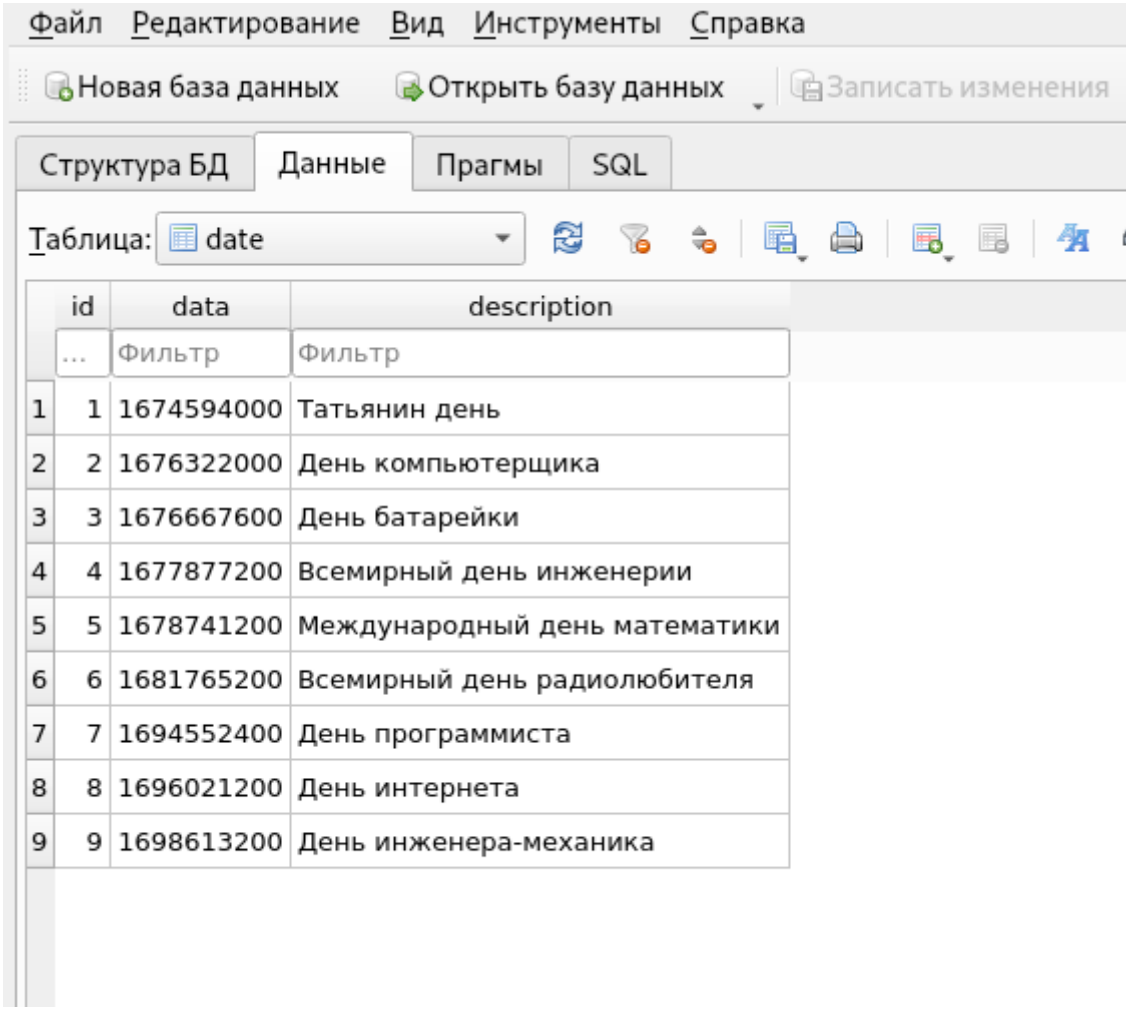
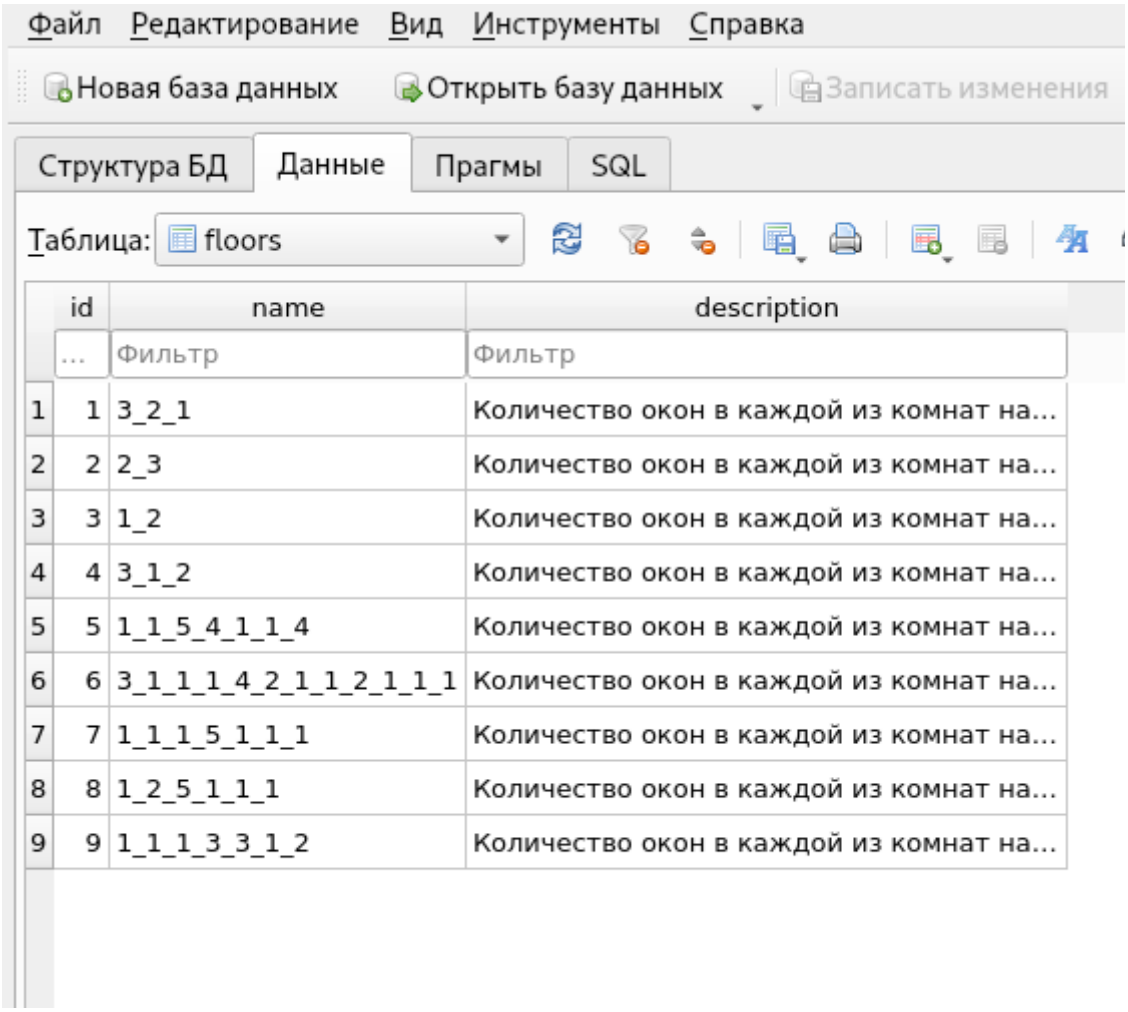


Рисунок 3: Снимок экрана разработанного программного продукта



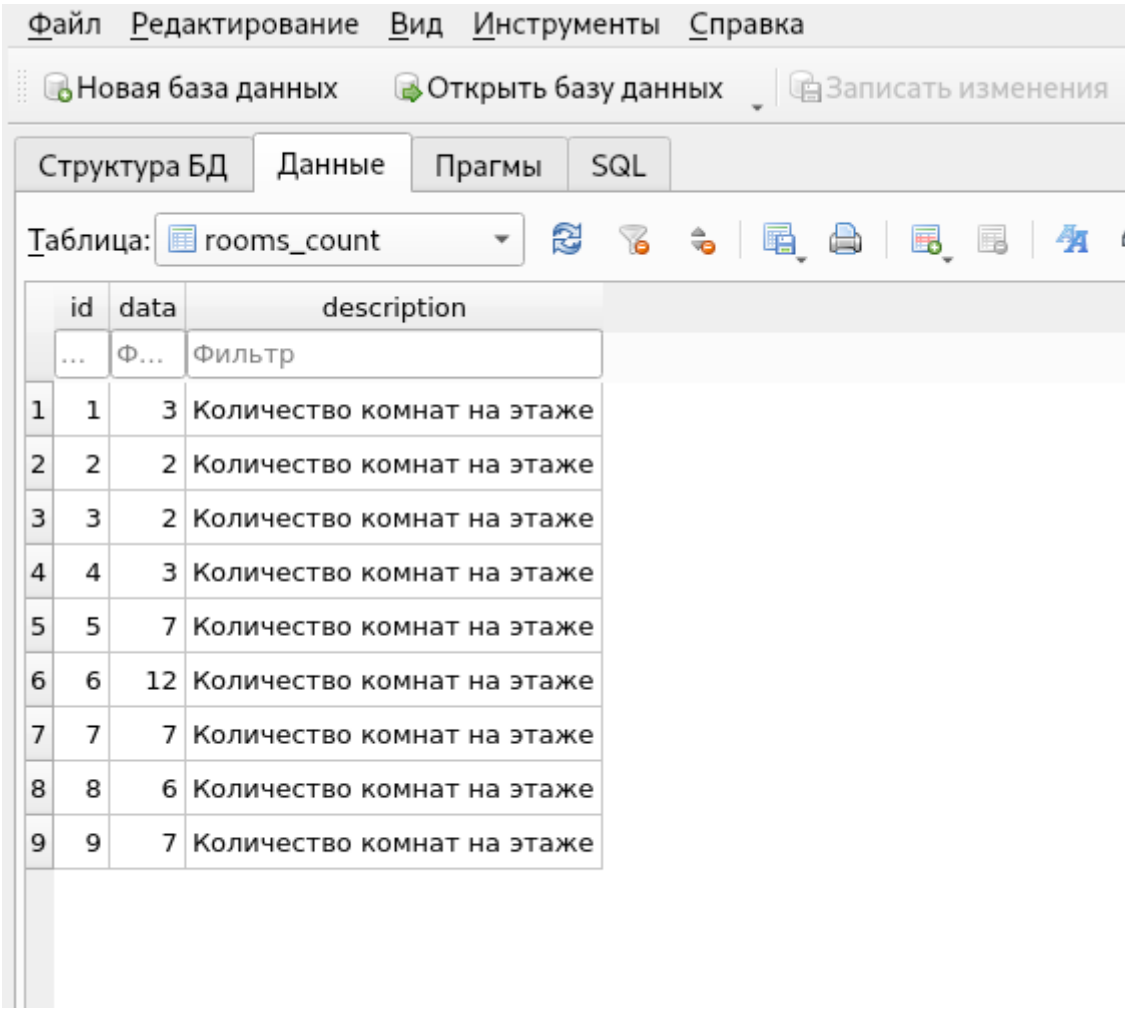
	id	data	description
	...	Фильтр	Фильтр
1	1	1674594000	Татьянин день
2	2	1676322000	День компьютерщика
3	3	1676667600	День батарейки
4	4	1677877200	Всемирный день инженерии
5	5	1678741200	Международный день математики
6	6	1681765200	Всемирный день радиолюбителя
7	7	1694552400	День программиста
8	8	1696021200	День интернета
9	9	1698613200	День инженера-механика

Рисунок 4: Снимок экрана СУБД 1



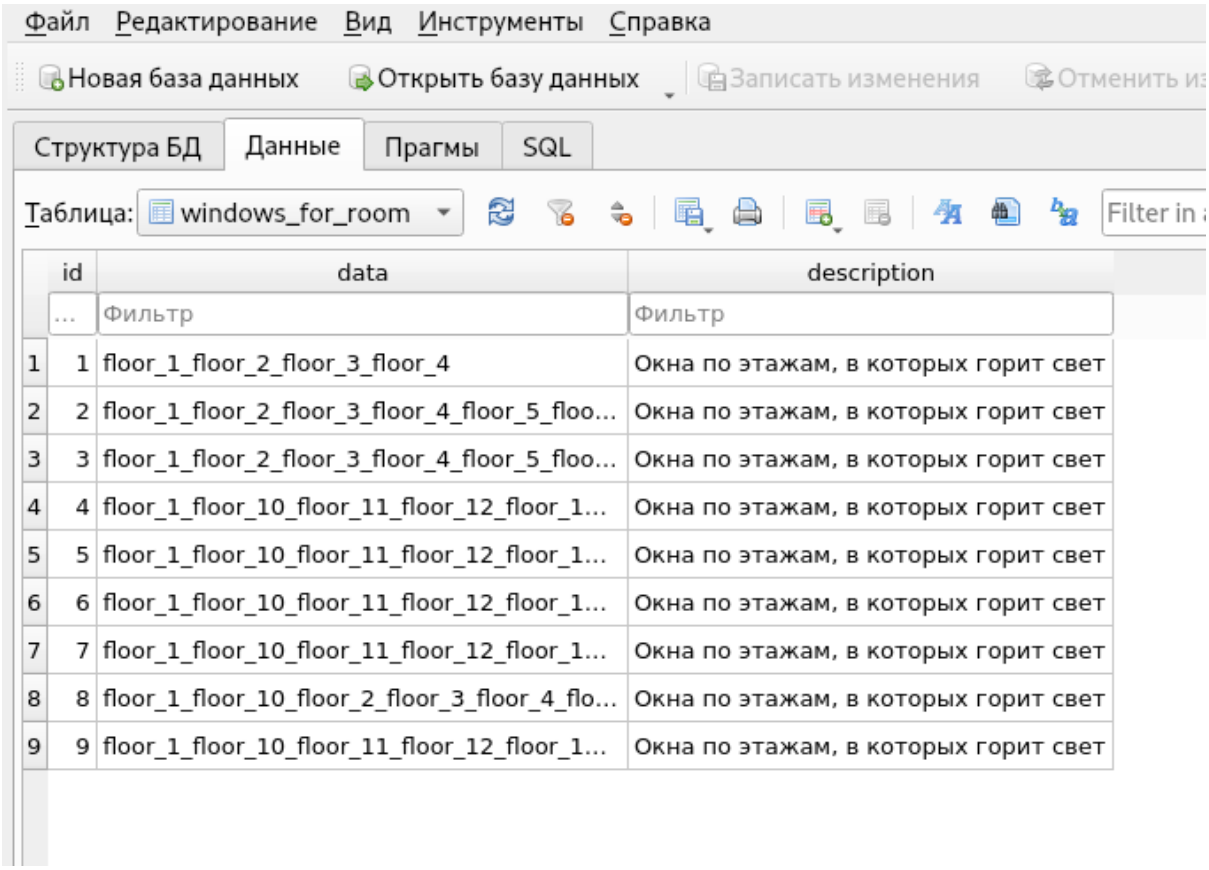
	id	name	description
	...	Фильтр	Фильтр
1	1	3_2_1	Количество окон в каждой из комнат на...
2	2	2_3	Количество окон в каждой из комнат на...
3	3	1_2	Количество окон в каждой из комнат на...
4	4	3_1_2	Количество окон в каждой из комнат на...
5	5	1_1_5_4_1_1_4	Количество окон в каждой из комнат на...
6	6	3_1_1_1_4_2_1_1_2_1_1_1	Количество окон в каждой из комнат на...
7	7	1_1_1_5_1_1_1	Количество окон в каждой из комнат на...
8	8	1_2_5_1_1_1	Количество окон в каждой из комнат на...
9	9	1_1_1_3_3_1_2	Количество окон в каждой из комнат на...

Рисунок 5 Снимок экрана СУБД 2



	id	data	description
	...	Ф...	Фильтр
1	1	3	Количество комнат на этаже
2	2	2	Количество комнат на этаже
3	3	2	Количество комнат на этаже
4	4	3	Количество комнат на этаже
5	5	7	Количество комнат на этаже
6	6	12	Количество комнат на этаже
7	7	7	Количество комнат на этаже
8	8	6	Количество комнат на этаже
9	9	7	Количество комнат на этаже

Рисунок 6: Снимок экрана СУБД 3



	id	data	description
	...	Фильтр	Фильтр
1	1	floor_1_floor_2_floor_3_floor_4	Окна по этажам, в которых горит свет
2	2	floor_1_floor_2_floor_3_floor_4_floor_5_floo...	Окна по этажам, в которых горит свет
3	3	floor_1_floor_2_floor_3_floor_4_floor_5_floo...	Окна по этажам, в которых горит свет
4	4	floor_1_floor_10_floor_11_floor_12_floor_1...	Окна по этажам, в которых горит свет
5	5	floor_1_floor_10_floor_11_floor_12_floor_1...	Окна по этажам, в которых горит свет
6	6	floor_1_floor_10_floor_11_floor_12_floor_1...	Окна по этажам, в которых горит свет
7	7	floor_1_floor_10_floor_11_floor_12_floor_1...	Окна по этажам, в которых горит свет
8	8	floor_1_floor_10_floor_2_floor_3_floor_4_flo...	Окна по этажам, в которых горит свет
9	9	floor_1_floor_10_floor_11_floor_12_floor_1...	Окна по этажам, в которых горит свет

Рисунок 7: Снимок экрана СУБД 4

Модуль app

Подмодуль app.py

```
import sys
sys.path.append("../")

from src.database import *

from flask import Flask, render_template,request

app = Flask(__name__)
```

```
createDatabase()

@app.route('/')
@app.route('/index', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        print(getAallData())
        date = request.form['date']
        datas = getAallData()[int(date)]
        print(datas)
        render_template('index.html', answer="Данные получены корректно",
numbers=datas[3], count_room=str(datas[1]), newdata=[i[0] for i in datas])

    return render_template('index.html')

@app.route('/vvod', methods=['GET', 'POST'])
def vvod():
    if request.method == 'POST':
        date = request.form['date']
        number = request.form['number']
        window = request.form['window']
        light = request.form['light']

        print(date, number, window, light)

        if date and number and window and light:
            addDataDate(date, "")
            addDataFloors(number, "")
            addDataWindows(window, "")
            addDataRooms(light, "")
        else:
            print("Не довведенны необходимые данные")
    return render_template('vvod.html')

@app.route('/list')
```

```
def list():
    return render_template('list.html')


if __name__ == '__main__':
    app.run(debug=True)
```

static/css

main.css

```
html {
    scroll-behavior: smooth;
    }

    body{
        margin: 0;
        font-family: 'Rokkitt', serif;
        font-size: 18px;
        line-height: 1.6;
        color: #020;
        background-size: cover;
        background: #FFE4B5;
    }

    *,
    *:before,
    *:after {
        box-sizing: border-box;
    }
    h1, h2, h3, h4, h5, h6 {
        margin: 0;
    }

    /*Container*/
    .container {
        width: 100%;
        max-width: 1430px;
```

```
    margin: 0 auto;
    padding:0 15px;
}

/*intro*/

.intro {
    display: flex;
    flex-direction: column;
    justify-content:center;
    width: 100%;
    height: auto;
    background: #FFE4B5;
    -webkit-background-size: cover;
    background-size: cover;
}

.intro__inner {
    width: 100%;
    max-width: 900px;
    margin: 0 auto;
    text-align: center;



}

/*заголовок*/
.intro__title {
    color: #020;
    font-size: 60px;
    font-weight: 700;
    text-transform: uppercase;
    line-height: 1;
    margin-top: 100px;
}
/*полоска под заголовком*/
.intro__title:after {
```

```
    content: "";
    display: block;
    width: 400px;
    height: 3px;
    margin: 20px auto 100px;
    background-color:  #B0F5BC;
}

.for__what {
    color: #020;
    font-size: 30px;
    font-weight: 700;
    line-height: 1.5;
    font-family: "Raleway";
    margin-bottom: 25px;

}
/*header*/
.header, .footer {
    display: flex;
    background-color: #B0F5BC;
    color: #fff;
    padding: 0px 15px 0px 15px;
    width: 100%;
    position: sticky;
    left: 0;
    right: 0;
    z-index: 1000;
}

.header {
    top: 0;
}

.footer {
    bottom: 0;
}
```

```
.header__inner {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
/*логотип*/
.header__logo{
    font-size: 25px;
    font-weight: 700;
    color: #020;
}

/*nav*/
.nav {
    font-size: 20px;
    padding: 5px;
    text-transform: uppercase;
}
/*ссылки*/
.nav__link {
    display: inline-block;
    vertical-align: top;
    margin: 0 10px;
    position: relative;
    color:#020;
    text-decoration:solid;
    transition: color 0.2s linear;
}

/*полоски под ссылками в правом верхнем углу*/
.nav__link:after {
    content: "";
    display: block;
    width: 100%;
    height: 1px;
    display: none;
```

```
    background-color: #fff;
    position: absolute;
    top: 100%;
    left: 0;
    z-index: 1;
}
/*изменение цвета слова и полоски снизу при наводе мышкой*/
.nav__link:hover {
    color: #fff;
}
.nav__link:hover:after {
    display: block;
}
/*курсор при наводе на ссылки в хэдере*/
.headerButton {
    cursor: pointer;
}

.btn {
    display: inline-block;
    padding: 8px 30px;
    border: 3px solid #020;
    font-size: 18px;
    font-weight: 700;
    color: #020;
    text-transform: uppercase;
    text-decoration: none;
    transition: background .1s linear, color .1s linear;
}
/*изменение цвета слова при наводе мышкой*/
.btn:hover {
    background-color: #B0F5BC;
    color: #020;
}

* {
  box-sizing: borde-box;
```

```
  margin: 0;
  padding: 0;
}

.block__left,
.block__right {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  flex-basis: 0;
  flex-grow: 1;
  padding: 3em;
}

.block__left {
  background-color: #FFE4B5;
  width: 100%;
  height: 100vh;
}

.block__right {
  background-color: #EAEAEA;
  width: 100%;
  height: 100vh;
}

@media (min-width:1000px) {
  .block {
    display: flex;
  }
  .block__left,
  .block__right {
    flex-basis: 0;
    flex-grow: 1;
  }
}
```

```
.box {
  width: 400px;
  height: 600px;
  background: #B0F5BC;
  margin-bottom: 10px;
}


.box1 {
    width: 50px;
    height: 50px;
    background: rgb(255, 238, 0);
    /* margin-bottom: 10px; */
}


.btnss {
    display: inline-block;
    flexDirection: row;
    justifyContent: space-between;
    padding: 8px 30px;

    font-size: 18px;
    font-weight: 700;
    color: #020;
    text-transform: uppercase;
    text-decoration: none;
    transition: background .1s linear, color .1s linear;
}

.box1 {
  width: 50px;
  height: 50px;
  background: yellow;

  margin: 20px;
```

```
    }
    .box2 {
      width: 50px;
      height: 50px;
      background: black;

      margin: 20px;
    }

    .row {
      flexDirection: row;
      white-space: nowrap;
      display:inline-block;
    }
```

templates

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta
      name="viewport"
      content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0"
    />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link
      rel="stylesheet"
      href="{{ url_for('static', filename='css/main.css') }}"
    />
  </head>
  <body>
    <div class="header">
      <div class="container">
```

```
        <div class="header__inner">
          <div class="header__logo">ИТ</div>

          <nav class="nav headerButton">
            <a class="nav__link" href="/index">ПО ДНЯМ</a>
            <a class="nav__link" href="/vvod">ВВОД ДАННЫХ</a>
            <a class="nav__link" href="/list">ОТОБРАЖЕНИЕ ПО МЕСЯЦАМ</a>
          </nav>
        </div>
      </div>
    </div>
</div>
<div class="block">
  <div class="block__left">
      <form action="/vvod" method="post">
     <h3 class="for__what">Дата:
         <input type="text" id="3" name="datee" required minlength="1"
maxlength="100000" size="40" />
     </h3>
          <input class="btn" type="submit" value="Показать">
      </form>
      <h3 class="for__what">Количество комнат, где репетируют: </h3>
      <h3 class="for__what">Номера комнат, где репетируют: </h3>
      <h3 class="for__what">Корректно ли найдены занятые комнаты: </h3>
  </div>
  <div class="block__right">
      <h3 class="for__what">Дата: </h3>
    <div class="box">
        {% for i in range(4) %}
        <div class="row">
            {% for j in range(4) %}
            {% if j % 2 == 0 %}
                <div class="box1"></div>
            {% endif %}
            {% if j % 2 != 0 %}
                <div class="box2"></div>
            {% endif %}
```

```
            {% endfor %}
        </div>
        {% endfor %}
    </div>
      <div class="btnss">
    <div class="block">
      <div class="block__left">
        <h3 class="for__what">
          Дата:
          <input
            type="text"
            id="1"
            name="date"
            required
            minlength="1"
            maxlength="100000"
            size="40"
          />
        </h3>
        <h3 class="for__what">Количество комнат, где репетируют:
{{count_room}}</h3>
        <h3 class="for__what">Номера комнат, где репетируют: {{numbers}}</h3>
        <h3 class="for__what">Корректно ли найдены занятые комнаты:
{{answer}}</h3>
      </div>
      <div class="block__right">
        <h3 class="for__what">Дата:</h3>
        <div class="box">
          {% for keys,keys2 in result1 %}
          <tr class="box1">
            <td>{{keys}}</td>
            <td>{{keys2}}</td>
          </tr>

          {% endfor %}
        </div>
```

```
        <div class="btnss">
          <a class="btn">вперед</a>
          <a class="btn">назад</a>
        </div>
      </div>
    </div>
  </body>
</html>
```

list.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv='X-UA-Compatible' content='ie=edge'>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/main.css') }}">
</head>
<body>
<div class="header">
    <div class="container">
        <div class="header__inner">
            <div class="header__logo">ИТ</div>

            <nav class="nav headerButton">
                <a class="nav__link" href="/index">ПО ДНЯМ</a>
                <a class="nav__link" href="/vvod">ВВОД ДАННЫХ</a>
                <a class="nav__link" href="/list">ОТОБРАЖЕНИЕ ПО МЕСЯЦАМ</a>
            </nav>
        </div>
    </div>
</div>

</body>
```

```
</html>
```

vvod.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
        content="width=device-width, user-scalable=no, initial-scale=1.0,
maximum-scale=1.0, minimum-scale=1.0">
    <meta http-equiv='X-UA-Compatible' content='ie=edge'>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/main.css') }}">
</head>
<body>
<div class="header">
    <div class="container">
        <div class="header__inner">
            <div class="header__logo">ИТ</div>

            <nav class="nav headerButton">
                <a class="nav__link" href="/index">ПО ДНЯМ</a>
                <a class="nav__link" href="/vvod">ВВОД ДАННЫХ</a>
                <a class="nav__link" href="/list">ОТОБРАЖЕНИЕ ПО МЕСЯЦАМ</a>
            </nav>
        </div>
    </div>
</div>

<div id="vvod" class="intro">
    <div class="container">
        <div class="intro__inner">
            <form action="/vvod" method="post">

                <h1 class="intro__title">Введите данные</h1>
```

```
                <h3 class="for__what">Дата:
                    <input type="text" id="1" name="date" required minlength="1"
maxlength="100000" size="60" />
                </h3>

                <h3 class="for__what">Количество комнат на этаже:
                    <input type="text" id="2" name="number" required
minlength="1" maxlength="100000" size="60" />
                </h3>

                <h3 class="for__what">Количество окон в комнате:
                    <input type="text" id="3" name="window" required
minlength="1" maxlength="100000" size="60" />
                </h3>

                <h3 class="for__what">Комнаты, где горят лампочки:
                    <input type="text" id="4" name="light" required
minlength="1" maxlength="100000" size="60" />
                </h3>

                <input class="btn" type="submit" value="Сохранить">
            </form>
        </div>
    </div>

    <div class="line">
</div>
</div>

</body>
</html>
```

db.sql

```
BEGIN TRANSACTION;
CREATE TABLE IF NOT EXISTS "date" (
      "id"  INTEGER,
```

```
	"data"	INTEGER NOT NULL,
	"description"	TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "rooms_count" (
	"id"	INTEGER,
	"data"	INTEGER NOT NULL,
	"description"	TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "windows_for_room" (
	"id"	INTEGER,
	"data"	TEXT NOT NULL,
	"description"	TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
CREATE TABLE IF NOT EXISTS "floors" (
	"id"	INTEGER,
	"name"	TEXT NOT NULL,
	"description"	TEXT NOT NULL,
	PRIMARY KEY("id" AUTOINCREMENT)
);
INSERT INTO "date" VALUES (1,1,'');
INSERT INTO "date" VALUES (2,1,'');
INSERT INTO "date" VALUES (3,1,'');
INSERT INTO "rooms_count" VALUES (1,4,'');
INSERT INTO "rooms_count" VALUES (2,4,'');
INSERT INTO "rooms_count" VALUES (3,4,'');
INSERT INTO "windows_for_room" VALUES (1,'3','');
INSERT INTO "windows_for_room" VALUES (2,'3','');
INSERT INTO "windows_for_room" VALUES (3,'3','');
INSERT INTO "floors" VALUES (1,'2','');
INSERT INTO "floors" VALUES (2,'2','');
INSERT INTO "floors" VALUES (3,'2','');
COMMIT;
```

Модуль src

Подмодуль database.py

```
#
# Файл, содеражщий в себе исходный код всех операций с БД
# База данных используется в формате SQLite со встроенное в python библиотекой sqilte3
# База данных хранит в себе информацию в долгострочной перспективе на локальном устройстве
#


import sqlite3 as sql

# переменная для хранения пути до файла БД в формате .db
FILE: str = "data.db"


def createDatabase() -> None:
    '''
    Функция создания новой таблицы в базе данных _._._

    База данных хранит в себе информацию:

    - 1
    - 2
    - 3

    '''

    db = sql.connect(FILE)
    cursor = db.cursor()

    # делаем запрос БД
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS date (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
            data INTEGER NOT NULL,
            description TEXT NOT NULL
        )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS rooms_count (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            data INTEGER NOT NULL,
            description TEXT NOT NULL
        )
    """)

    # data будет разделена _
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS windows_for_room(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            data TEXT NOT NULL,
            description TEXT NOT NULL
        )
    """)

    # Положения окон формируется по _
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS floors(
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT NOT NULL,
            description TEXT NOT NULL
        )
    """)

    # сохраняем и закрываем
    db.commit()
    db.close()


def addDataDate(data, description):
```

```
    db = sql.connect(FILE)
    cursor = db.cursor()

    cursor.execute(f"""
        INSERT INTO date(data, description) VALUES('{data}', '{description}')
    """)

    db.commit()
    db.close()

    return "Success"


def addDataRooms(data, description):
    db = sql.connect(FILE)
    cursor = db.cursor()

    cursor.execute(f"""
        INSERT INTO rooms_count(data, description) VALUES('{data}',
'{description}')
    """)

    db.commit()
    db.close()

    return "Success"


def addDataWindows(data: list, description: str):
    db = sql.connect(FILE)
    cursor = db.cursor()

    data = '_'.join([str(i) for i in data])
    cursor.execute(f"""
        INSERT INTO windows_for_room(data, description) VALUES('{data}',
'{description}')
    """)
```

```
    db.commit()
    db.close()

    return "Success"


def addDataFloors(data: list, description: str):
    db = sql.connect(FILE)
    cursor = db.cursor()

    data = '_'.join([str(i) for i in data])
    cursor.execute(f"""
        INSERT INTO floors(name, description) VALUES('{data}', '{description}')
    """)

    db.commit()
    db.close()

    return "Success"

def getAallData():
    db = sql.connect(FILE)
    cursor = db.cursor()

    res = dict()

    cursor.execute(f"""
        SELECT * FROM date
    """)

    fetch = cursor.fetchall()

    for elem in fetch:
        if elem[0] in res.keys():
            res[elem[0]].append(elem[1:])
        else:
```

```
            res[elem[0]] = [elem[1:]]

    cursor.execute(f"""
        SELECT * FROM floors
    """)

    fetch = cursor.fetchall()
    for elem in fetch:
        if elem[0] in res.keys():
            res[elem[0]].append(elem[1:])
        else:
            res[elem[0]] = [elem[1:]]

    cursor.execute(f"""
        SELECT * FROM rooms_count
    """)

    fetch = cursor.fetchall()
    for elem in fetch:
        if elem[0] in res.keys():
            res[elem[0]].append(elem[1:])
        else:
            res[elem[0]] = [elem[1:]]



    cursor.execute(f"""
        SELECT * FROM windows_for_room
    """)

    fetch = cursor.fetchall()
    for elem in fetch:
        if elem[0] in res.keys():
            res[elem[0]].append(elem[1:])
        else:
            res[elem[0]] = [elem[1:]]

    return res
```

```
# запуск тестирующей части (при неоходимости)
if __name__ == "__main__":
    createDatabase()
    # тестим получение данных
    getAallData()
```

Подмодуль logic.py

```
#
# Файл, содержащий в себе основную логику проекта
# По сути здесь располагается вся backend часть, необходимая для обработки данных
# Обработанные данные возвращаются по обратно зависимому признаку, т.е пришел запрос - получен ответ
#


import requests
import json

URL = "https://olimp.miet.ru/ppo_it_final"
HEADER = {"X-Auth-Token": 'ppo_11_11573'}


session = requests.Session()


def getData(url: str = "/date") -> dict:
    '''
    Функция парсинга данных.

    Функция принимает в себя url страницы для парсинга и возвращает словарь данных.

    '''
```

```
    p = session.get(URL+url, headers=HEADER)
    return json.loads(p.text)


def getAllData():
    res = []
    ls = getData()["message"]
    for elem in ls:
        data = elem.split("-")

        day = data[0]
        month = data[1]
        year = data[2]

        p = session.get(
            URL+f"?day={day}&month={month}&year={year}", headers=HEADER)
        res.append(json.loads(p.text)["message"])
    return res


def postData():
    data = {
        "payload": {
            "data": {
                "count": 4,
                "rooms": [
                    3,
                    5,
                    9,
                    10
                ]
            },
            "date": "01-05-21"}
    }
    HEADER = {"X-Auth-Token": 'ppo_11_11573'}
    p = session.post(URL, params=data, headers=HEADER)
    return True
```

```
# запуск тестирующих модулей по отдельности
if __name__ == "__main__":
    # print(getAllData())
    postData()
```

Подмодуль main.py

```
#
# Файл, содержащий основные надстройки проекта
# Запускает и подгружает основные компоненты всего приложения
# Импорт библиотек и их связь в единую схему
#

from database import *
from logic import *


def main():
    createDatabase()

    for data in getAllData():
        data1 = data['date']
        data2 = data['flats_count']
        data3 = data['windows']
        data4 = data['windows_for_flat']

        addDataDate(data1['data'], data1['description'])
        addDataRooms(data2['data'], data2['description'])
        addDataWindows(data3['data'], data3['description'])
        addDataFloors(data4['data'], data4['description'])


# запустить только при исходном файле
if __name__=="__main__":
    main()
```

```
    print(getAallData())
```

Подмодуль test.py

```
#
# Unit тесты
# Тестирование компонентов и функций отдельного выделенных в backend и frontend
части
# К примеру тестирование на ошибки БД и статус коды для REST API запросов
#

from colorama import init, Fore


from database import *
from logic import *


def test(func, data, result):
    try:
        if type(func(*data)) == result:
            return Fore.GREEN + 'THE TEST IS PASSED!'
        else:
            return Fore.RED + 'THE TEST IS FAILED'
    except:
        return Fore.GREEN + 'THE TEST IS PASSED!'


print(test(getData, (), dict))
print(test(getData, (), dict))
print(test(getData, (), dict))
print(test(getData, (), dict))
print(test(getData, (), dict))
print(test(getData, (), dict))
print(test(getData, (), dict))
print(test(getData, (), dict))
```

```
print(test(addDataWindows, ([1], "wnciwnv"), str))
print(test(addDataWindows, (["svdnwv"], "sfwef"), str))
print(test(addDataWindows, ([], "wncwdvweveiwnv"), str))
print(test(addDataWindows, ([1, 2, "svjowdo", 3], "rewverv"), str))
print(test(addDataWindows, ([1, 2, "wnovew", 3], "wnciwnv"), str))
print(test(addDataWindows, ([1, 2, 3, "wonewo"], "12434"), str))
print(test(addDataWindows, ([1, 2, 3, "wjndcvw"], "dvewrv r"), str))
print(test(addDataWindows, ([1, 2, 3, 2032], "wcwev"), str))
print(test(addDataWindows, ([1, 2, 36, 4, 2], "rwvrwvvfve"), str))
print(test(addDataWindows, ([1, 2, 3, 3, 3], "1343tbrb"), str))
print(test(addDataWindows, ([1, 22, 3], "2efwevr"), str))
print(test(addDataWindows, ([1, 4, 3], "wdvwrv"), str))
print(test(addDataWindows, ([1, 2, 3, 4], "wdvwrv"), str))
```