ivanovbroo@gmai…

Цифровая кафедра. Веб-разработка 2022-2023

29 ноя 2022, 21:38:09
старт: 10 окт 2022, 11:03:58
начало: 25 сен 2022, 14:59:00

20. Исключения

Ограничение времени	10 секунд
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Напишем часть сервиса, который будет помогать бронировать переговорки в офисе. Для этого опишем класс Booking - его объекты будут содержать информацию о конкретном бронировании, а также вспомогательную функцию create_booking, которая будет создавать новый объект бронирования и записывать информацию о бронировании в базу данных бронирований через предоставляемое API. Возвращать она должна будет статус создания бронирования (получилось или переговорка уже занята) и информацию о брони в формате JSON. Ниже - подробности.

Класс `Booking` должен обладать следующим функционалом.

- конструктор должен принимать три аргумента в следующем порядке: название переговорки, datetime начала брони и datetime конца брони
- внутри конструктора, если datetime конца брони оказался раньше, чем datetime начала, нужно вызвать исключение ValueError

Также у объектов этого класса должны быть следующие поля (рекомендую сделать часть из них в виде проперти):

- `room_name`
  - название переговорки, полученное из конструктора
- `start`
  - datetime начала брони. Должна быть возможность назначить новое время начала уже созданной брони
- `end`
  - datetime конца брони. Должна быть возможность назначить новое время конца уже созданной брони
- `duration`
  - длительность бронирования в минутах (гарантируется, что длительность любой встречи кратна одной минуте, поэтому это должно быть целое число)
- `start_date`
  - дата начала брони в формате YYYY-MM-DD (строка)
- `end_date`
  - дата конца брони в формате YYYY-MM-DD (строка)
- `start_time`
  - время начала брони в формате HH-MM (строка)
- `end_time`
  - время конца брони в формате HH-MM (строка)

Функция create_booking должна обладать следующей сигнатурой:

`create_booking(room_name, start, end) -> str`,

где аргументы - это те же аргументы, которые принимает конструктор Booking, а выходная строка - это json определенного формата, который описан чуть ниже по тексту.

Будем считать, что взаимодействие с базой данных у нас уже описано нашим коллегой в соседнем файле `api.py`. В нем есть уже готовая функция `register_booking`, которая:

- принимает на вход объект класса Booking
- возвращает True, если бронирование получилось создать
- возвращает False, если мы пытаемся забронировать уже занятую в это время переговорку
- если такой переговорки не существует, вызывается KeyError

Таким образом, в том же файле, что и класс Booking, вам нужно описать фукнцию `create_booking`, которая:

- обладает сигнатурой `create_booking(room_name, start, end) -> str`, где аргументы - те же, что и в конструкторе Booking

- в самом начале своей работы выводит на экран текст: `Начинаем создание бронирования`
- внутри функции создается объект класса Booking, а также вызывается функция register_booking, которая принимает на вход созданный объект. Должны быть обработаны все случаи работы register_booking: True, False и KeyError. Сделать это поможет конструкция try-except
- перед выходом из функции должно выводиться на экран сообщение `Заканчиваем создание бронирования`. Это должно происходить в любом случае, даже если мы попытались создать бронирование с неверными датами и получили ValueError (см. описание класса Booking). Для этого рекомендую использовать блок finally, в котором описать этот print

Функция должна возвращать json-строку с ответом, в котором будут содержаться следующие поля:

- `created`: true/false, получилось ли забронировать комнату. Если возникло KeyError, то нужно здесь записать false
- `msg`: сообщение с пояснениями. Сообщение должно быть одним из следующих: `Бронирование создано`, `Комната занята`, `Комната не найдена`. Сообщение выбирается на основе того, что вернет функция register_booking
- `booking`
  - это бронирование в виде json-строки. Должны содержаться поля: room_name, duration, start_date, end_date, start_time, end_time.

Формат ввода

```
result = create_booking(
    "Вагнер",
    datetime.datetime(2022, 9, 1, 14),
    datetime.datetime(2022, 9, 1, 15, 15)
)
print(result)
```

Формат вывода

Вывод на экран:
```
Начинаем создание бронирования
Заканчиваем создание бронирования
```

Функция возвращает:
```
{
  "created": false,
  "msg": "Комната занята",
  "booking": {
    "room_name": "Вагнер",
    "start_date": "2022-09-01",
    "start_time": "14:00",
    "end_date": "2022-09-01",
    "end_time": "15:15",
    "duration": 75
  }
}
```

Примечания

Пример написания функции create_booking:
```
from api import register_booking

def create_booking(room_name, start, end):
    booking = Booking(........)
    try:
        result = register_booking(booking)
    except ....:
        ....

    return json.dumps(......)
```

Набрать здесь | Отправить файл

```
1
```

Отправить

Предыдущая

Следующая