Online Food Ordering System

Group 3 黃柏盛、羅翊瑄、陳昱銓、林柏廷

Motivation

Since COVID-19 ravages worldwide, the restaurant online ordering is growing and the global food delivery market is worth more than $150 billion, having more than tripled since 2017. Many companies want to occupy a place in the hottest market. There are a lot of opportunities to get profits from a slice of the market, so we want to build our own online ordering system to join the fierce competition. Hope we can play an independent role in this market.

Abstract

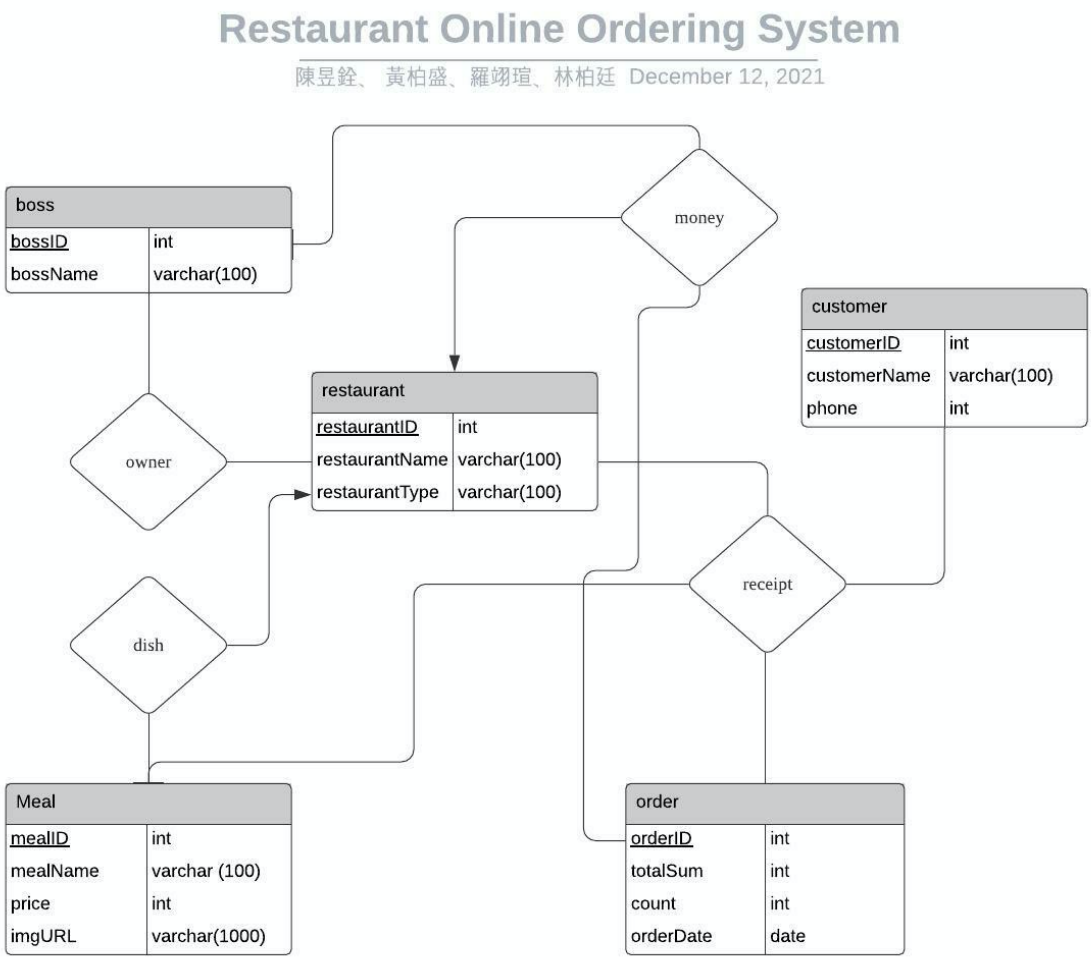
In the restaurant online ordering system database, we would usually query the following queries:

1. Categorize restaurants by restaurant type.
2. Categorize restaurants by price level.
3. Rank the popularity of each meal for a specified restaurant.
4. Get monthly average income for a restaurant in a specified month.
5. Search for a specific customer's ordering history from the database.

After thinking about what we need in the restaurant online ordering system, the most important relation is restaurant to customer, so the first step is to add the restaurant entity set and the customer entity set. When a customer wants to order some meals, we have to give a receipt. And the order entity set will be created to record what the customer ordered. In the restaurant entity set , owners have the right to edit their restaurant info, so we have to create the boss entity. In order to complete the details of the restaurant, we add the meal entity set to store meal menus instead of storing in the restaurant entity set because of the complexity of the detail (e.g. meal name, price, etc.). It would be lengthy and troublesome to add into the restaurant entity set.

The owner relation represents the relation between the restaurant entity set and the boss entity set. The dish relation represents the relation between the restaurant entity set and the meal entity set. The money relation represents the relation between the boss entity set, the restaurant entity set and the order entity set, which can help us get income from the restaurant per month. The receipt relation represents the relation between restaurant entity set, the customer entity set and the order entity set, which can be a proof of a purchase.

E-R Diagram


Restaurant Online Ordering System
陳昱銓、 黃柏盛、羅翊瑄、林柏廷 December 12, 2021
boss
bossID int
bossName varchar(100)
money
customer
customerID int
customerName varchar(100)
phone int
restaurant
restaurantID int
restaurantName varchar(100)
restaurantType varchar(100)
owner
receipt
dish
Meal
mealID int
mealName varchar (100)
price int
imgURL varchar(1000)
order
orderID int
totalSum int
count int
orderDate date

Data Schema

Schemas Derived from Entity Sets:

Restaurant (<u>restaurantID</u>, restaurantName, restaurantType, imgURL)

```
create table Restaurant (
        restaurantID int
        restaurantName  varchar (100)
        ~~bossID int~~
        restaurantType varchar (100)
        imgURL varchar(100)
        primary key (restaurantID)
);
```

The Restaurant entity set includes a specific restaurantID, a restaurant name, a owner which means bossID, type of the restaurant (ex. Japanese, Chinese...etc).

Order (<u>orderID</u>, orderDate, count, totalSum)

```
create table Order (
        orderID int
        customerID int
        restaurantID int
        mealID int
        orderDate date
        count int
        totalSum int
        primary key (orderID, mealID)
        foreign key (customerID) references Customer
        foreign key (restaurantID) references Restaurant
        foreign key (mealID) references Meal
);
```

The Order entity set includes attributes of orderID, customerID, restaurantID, mealID, orderDate, count, and totalSum. We build orderDate attribute because we provide two functions. One is to get the average monthly income for a restaurant in a specified month, and the other is to search for a specific customer's ordering history from the database. Because we might order more than one meal at a time, we provide count attribute to record the number of the same meals on the same orderID. We create a totalSum attribute to record the total amount of each kind of meal in one order. We choose orderDate attribute and mealID attribute as our primary key because each meal has its own ID. Even though they have the same meal names in different restaurants, we still can distinguish them by mealID.

Customer (<u>customerID</u>, customerName, phone)

```
create table Customer (
        customerID int
        customerName varchar (100)
        phone int
        primary key (customerID)
);
```

The Customer entity set includes a specific customerID, a customer name, and a phone number, which the restaurant can contact with. The primary key is the customerID. The phone number might be unique to each real user, but sometimes it will go wrong. (e.g. Bob created a user account with his phone number in the past. Later on, he changed his phone number. The telecommunications company sold the phone number to another person.)

Meal (mealID, mealName, price, imgURL)

```
create table Meal (
        mealID int
        mealName varchar(100)
        restaurantID int
        price int
        imgURL varchar(1000)
        primary key (mealID)
        foreign key (restaurantID) references Restaurant
);
```

The Meal entity set includes attributes of specific mealID, mealName, restaurantID which represents a restaurant, price, imgURL linked to a picture of the meal.
(we tend to upload the image to Google cloud and save the URL to the database, so we can show the image by the URL)

Boss (bossID, bossName)

```
create table Boss (
        bossID int
        bossName varchar (100)
        ~~restaurantID int~~
        primary key (bossID)
);
```

The Boss entity set includes attributes of specific bossID, name, restaurantID which represents a restaurant. We build the Boss entity set to retain the capability to insert, update and delete their Restaurant entity.

Schema Derived from Relationship Sets:

The dish relationship between Restaurant and Meal entity set:
dish (restaurantID, mealID)

The owner relationship between Boss and Restaurant entity set:
owner (bossID, restaurantID)

The receipt relationship between Order, Restaurant, Customer and Meal entity set:
receipt (orderID, restaurantID, customerID, mealID)

The money relationship between Boss, order and Restaurant entity set:
money (restaurantID, bossID, orderID)

Non-trivial Functional Dependencies

Restaurant (<u>restaurantID</u>, restaurantName, restaurantType, imgURL)

Our functional dependency:

(1) restaurantID -> restaurantName, restaurantType, imgURL

(2) restaurantName -> restaurantType, imgURL

, so Restaurant entity set belongs to non-trivial functional dependency.

2NF holds because there is a transitive functional dependency, but non-prime attributes (restaurantName, restaurantType, imgURL) are fully functionally dependent on primary key (restaurantID).

Order (<u>orderID</u>, orderDate, count, totalSum)

Our functional dependency:

(1) orderID -> orderDate, count, totalSum

, so Order entity set belongs to non-trivial functional dependency.

4NF holds because orderID is a super-key, and there is no multi-valued dependency.

Customer (<u>customerID</u>, customerName, phone)

Our functional dependency:

(1) customerID -> customerName, phone

, so Customer entity set belongs to non-trivial functional dependency.

4NF holds because customerID is a super-key, and there is no multi-valued dependency.

Meal (<u>mealID</u>, mealName, price, imgURL)

Our functional dependency:

(1) mealID -> mealName, price, imgURL

, so Meal entity set belongs to non-trivial functional dependency.

4NF holds because mealID is a super-key, and there is no multi-valued dependency.

Boss (<u>bossID</u>, bossName)

Our functional dependency:

(1) bossID -> bossName

, so Boss entity set belongs to non-trivial functional dependency.

4NF holds because bossID is a super-key, and there is no multi-valued dependency.

Explanation for Dataset Generator

真實資料

真實資料為利用網路去搜尋真實餐廳得到的數據，請參考 csv 檔或 json 檔。

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
boss	16	37.7 B	603.0 B	1	36.9 KB	
customer	14	67.8 B	949.0 B	1	36.9 KB	
dish	91	68.0 B	6.2 KB	1	36.9 KB	
meal	91	249.8 B	22.7 KB	1	36.9 KB	
money	12	89.0 B	1.1 KB	1	36.9 KB	
order	12	74.0 B	888.0 B	1	36.9 KB	
owner	16	68.0 B	1.1 KB	1	36.9 KB	
reciept	12	113.0 B	1.4 KB	1	36.9 KB	
restaurant	16	690.9 B	11.1 KB	1	36.9 KB	

真實資料數量 documents

合成資料產生方法

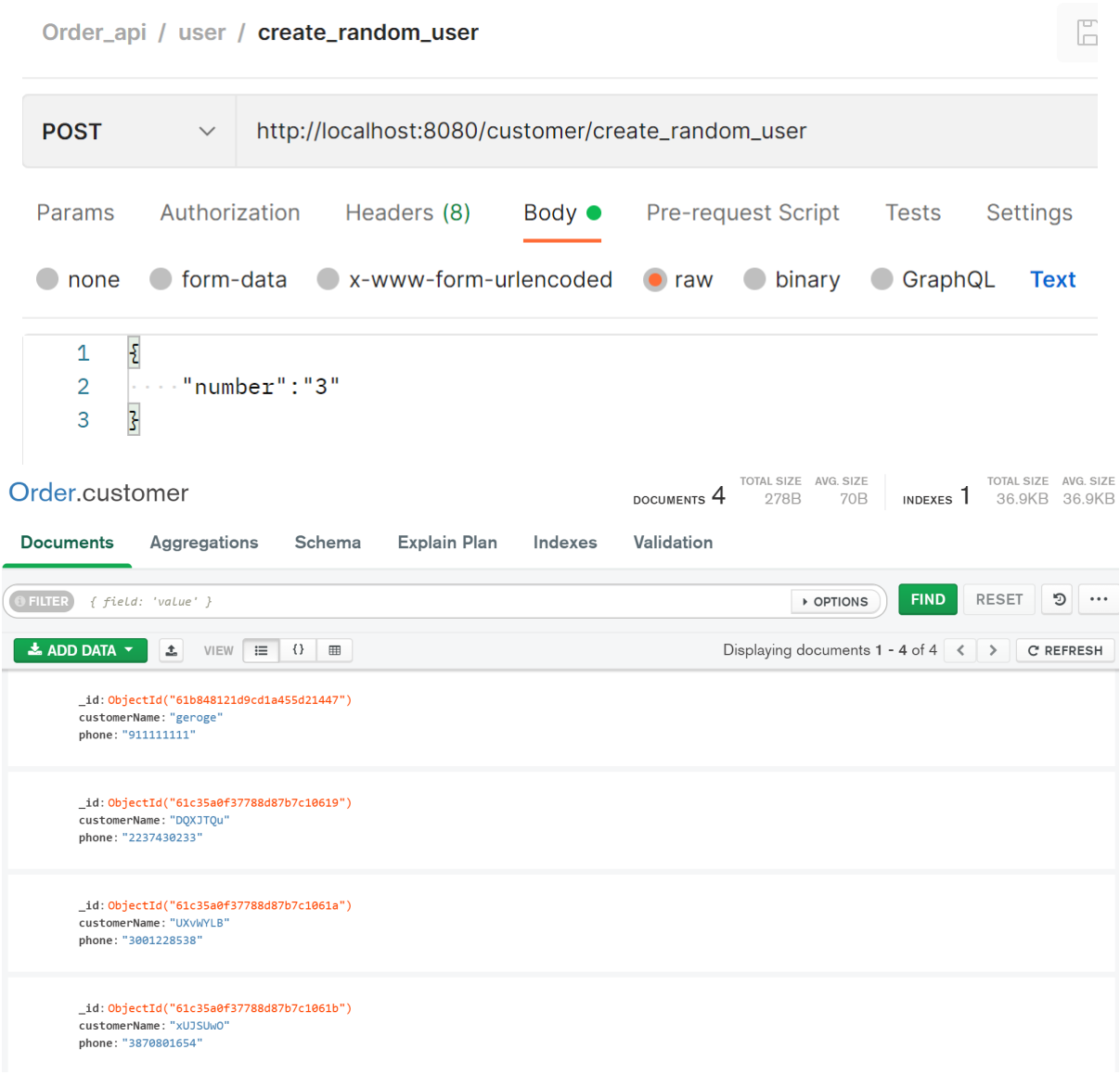
我們的組別使用的資料庫為 NoSQL 的 MongoDB 來做出仿 SQL 的資料庫架構。需要存取的 collection 為 9 個，分別為 5 個 entity set : **boss, restaurant, customer, order, meal，以及 4 個** relation : **owner, dish, money, receipt。使用** python 的 Django 來與 MongoDB 串接，利用 code 和 request 的方式直接生成資料。(資料庫裡的第一筆資料為手動輸入，方便做觀察)

1. customer

生成 customer。

```
@api_view(['POST'])
def create_random_user(request):
    stringNumber = JSONParser().parse(request)
    number = eval(stringNumber['number'])
    for i in range(number):
        phone = customer_generator.phone_generator()
        customerName = customer_generator.customer_name_generator()
        customer = {
            'customerName':customerName,
            'phone':phone,
        }
        customer_serializer = serializers.CustomerSerializer(data = customer)
        if customer_serializer.is_valid():
            customer_serializer.save()
    return JsonResponse({'message':'create'},status=status.HTTP_201_CREATED)
import random
import string
def customer_name_generator():
    name = ''.join(random.choice(string.ascii_letters) for _ in range(7))
    return name
def phone_generator():
    phone = ''.join(random.choice(string.digits) for _ in range(10))
    return phone
```

Request 指定要生成的數量，用 random 產生隨機的字符來建立名字，隨機的數字來建立手機號碼，最後存進資料庫裡。



得到三筆合成資料

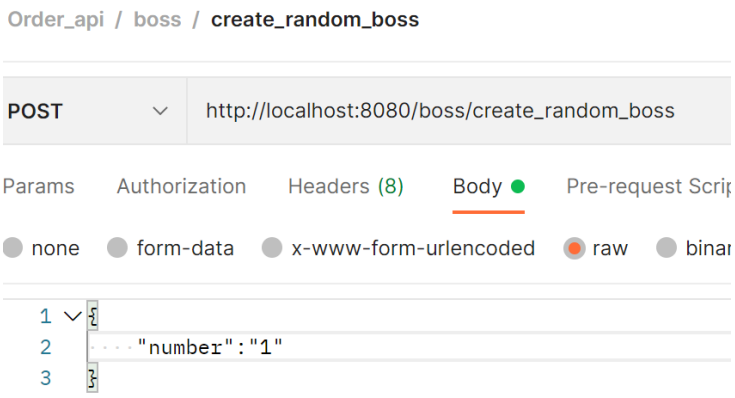
2. boss

生成 boss 時需要附帶的條件為給他一間餐廳，並建立 owner 的關係。

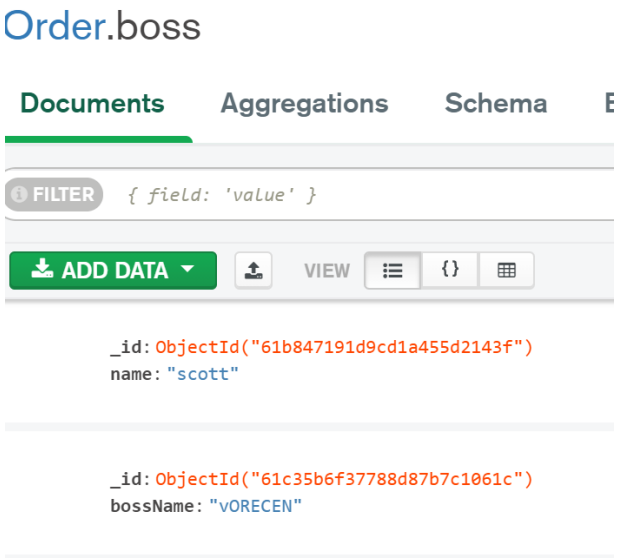
```python
from rest_framework.decorators import api_view
from django.http.response import JsonResponse
from rest_framework.parsers import JSONParser
from rest_framework import status
from OrderBackend import serializers
from utils import restaurant_generator,customer_generator


@api_view(['POST'])
def create_random_boss(request):
    stringNumber = JSONParser().parse(request)
    number = eval(stringNumber['number'])
    for i in range(number):
        boss = {
            "bossName":customer_generator.customer_name_generator()#因為跟顧客名字沒有差別所以直接用這個func
        }
        boss_serializers = serializers.BossSerializer(data=boss)
        if boss_serializers.is_valid():
            boss_serializers.save()
        bossID = boss_serializers.data.get('id')
        restaurant_name = restaurant_generator.restaurant_name_generator()
        restaurant_type = restaurant_generator.restaurant_type_generator()
        restaurant = {
            'restaurantName':restaurant_name,
            'restaurantType':restaurant_type
        }
        restaurant_serializer = serializers.RestaurantSerializer(data = restaurant)
        if restaurant_serializer.is_valid():
            restaurant_serializer.save()
        restaurantID = restaurant_serializer.data.get('id')

        owner = {
            "bossID":bossID,
            "restaurantID":restaurantID
        }
        owner_serializers = serializers.OwnerSerializer(data=owner)
        if owner_serializers.is_valid():
            owner_serializers.save()
    return JsonResponse({'message':'create'},status=status.HTTP_201_CREATED)
```

```python
import random
import string
def restaurant_name_generator():
    name = ''.join(random.choice(string.ascii_letters) for _ in range(5))
    return name
def restaurant_type_generator():
    data = ['Chinese','American','Japanese']
    type = data[random.randrange(3)]
    return type
```

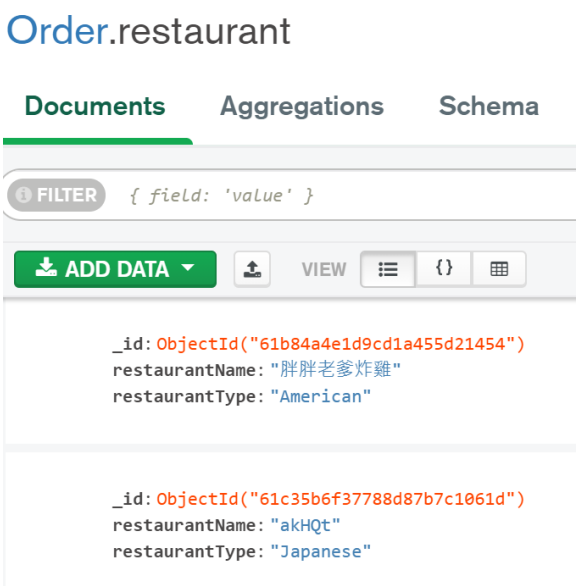
Boss 名稱的取名方式與 customer 名稱一樣，restaurant 名稱方式也一樣，不過長度為 5，餐廳種類為 3 取 1。Request 生成指定數量的 boss。



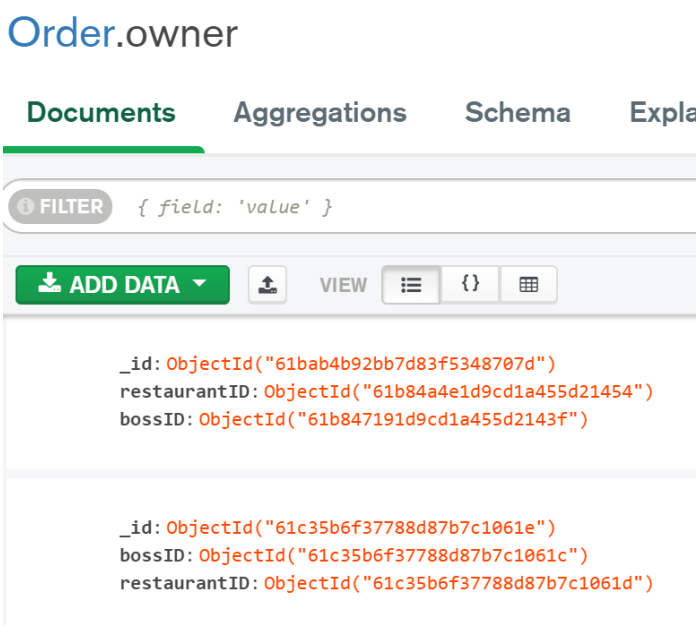
發送數量 1



Boss



restaurant



能發現有好好的儲存到 id

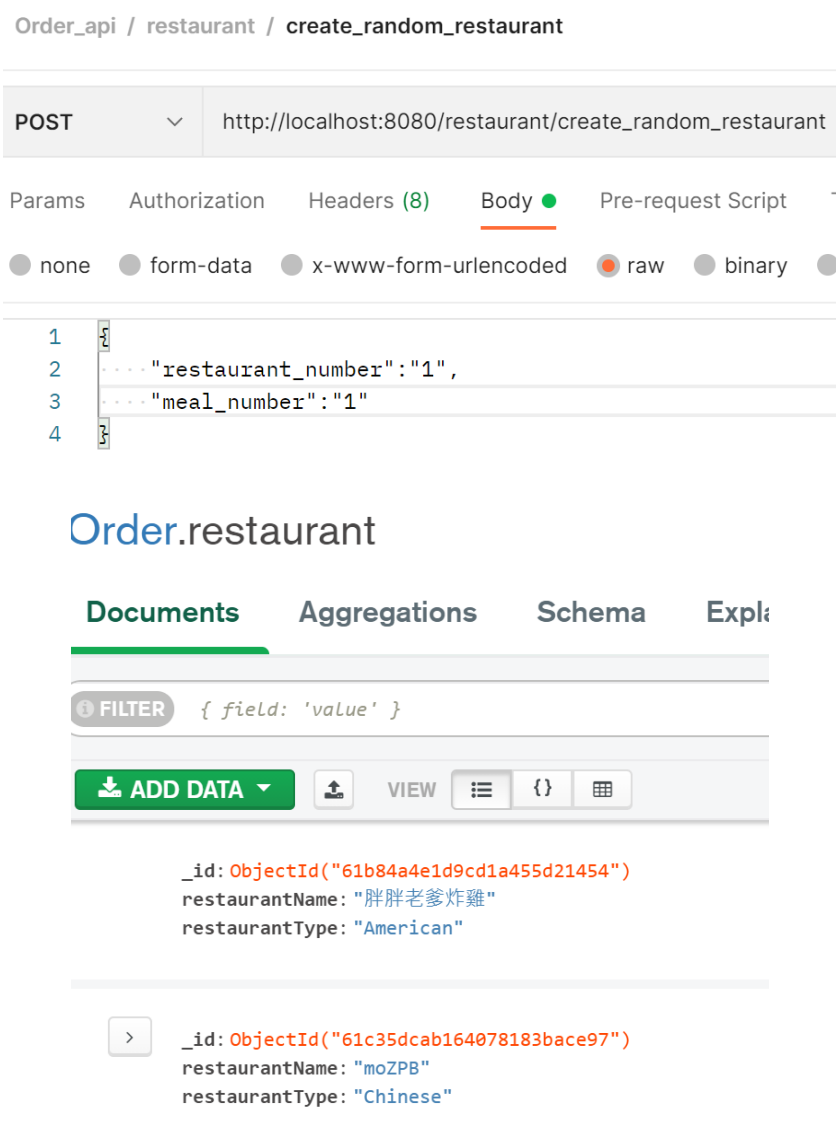
3. restaurant

生成每間餐廳會同時生成一個 boss，生成指定 meal 數量，建立 owner, dish 的關係

```python
# data set, restaurant_number指定餐廳數量, mael_number指定每間餐廳餐點數量
@api_view(['POST'])
def create_random_restaurant(request):
    body = JSONParser().parse(request)
    meal_number = eval(body['meal_number'])
    restaurant_number = eval(body['restaurant_number'])
    for _ in range(restaurant_number):
        restaurant_name = restaurant_generator.restaurant_name_generator()
        restaurant_type = restaurant_generator.restaurant_type_generator()
        restaurant = {
            'restaurantName':restaurant_name,
            'restaurantType':restaurant_type
        }
        restaurant_serializer = serializers.RestaurantSerializer(data = restaurant)
        if restaurant_serializer.is_valid():
            restaurant_serializer.save()
        restaurantID = restaurant_serializer.data.get('id')
        boss = {
            "bossName":customer_generator.customer_name_generator()#因為跟顧客名字沒有
        }
        boss_serializers = serializers.BossSerializer(data=boss)
        if boss_serializers.is_valid():
            boss_serializers.save()
        bossID = boss_serializers.data.get('id')
        owner = {
            "bossID":bossID,
            "restaurantID":restaurantID
        }
        owner_serializers = serializers.OwnerSerializer(data=owner)
        if owner_serializers.is_valid():
            owner_serializers.save()
        #新增 meal
        for _ in range(meal_number):
            meal_name = meal_generator.food_catcher()
            price = random.randrange(501)
            imgURL = meal_generator.rd_img()
            meal = {
                'mealName': meal_name,
                'price': price,
                'imgURL':imgURL
            }
            meal_serializer = serializers.MealSerializer(data = meal)
            if meal_serializer.is_valid():
                meal_serializer.save()
            mealID = meal_serializer.data.get('id')
            dish = {
                'restaurantID':restaurantID,
                'mealID':mealID
            }
            dish_serializer = serializers.DishSerializer(data = dish)
            if dish_serializer.is_valid():
                dish_serializer.save()
    #response可能需要更動
    return JsonResponse({'message':'create'},status=status.HTTP_201_CREATED)
```

```python
import random

data = ["拉麵","生魚片","握壽司","散壽司","涼麵","咖哩","手捲","味噌湯",
        "大福","土瓶蒸","涮涮鍋","沙拉","可樂餅","生蠔","烤魚","羊小排",
        "牛小排","串燒","一夜干","壽喜燒","麻辣鍋","炒飯","烏龍麵","餃子",
        "義大利麵","披薩","燉飯","火腿","罐頭","醃漬品","湯","咖哩","滷肉",
        "羹","蛋","麻糬","豆腐","卷餅","春捲","熱炒","包子","饅頭","燒賣",
        "蒸餃","乳酪","蛋糕","冰淇淋","提拉米蘇","奶酪","布丁","果凍","餅乾",
        "粥","雪花冰","大魯麵","飯","甜食","鹽水雞","鹽酥雞","糕餅"]
def food_catcher():
    number = len(data)
    index = random.randrange(number)
    return data[index]
def rd_img():
    url = "https://picsum.photos/id/"+ str(random.randrange(1001)) +"200"
    return url
```

Meal 生成器利用 random 隨機選取食物，imgURL 隨機生成圖片網址，price 為範圍 1~501 的隨機數字。



restaurant

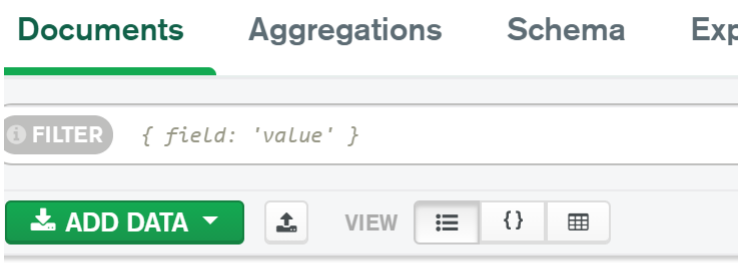
Order.boss


```
_id: ObjectId("61b847191d9cd1a455d2143f")
name: "scott"
```

```
_id: ObjectId("61c35dcab164078183bace98")
bossName: "crbqMBP"
```

Boss

Order.owner

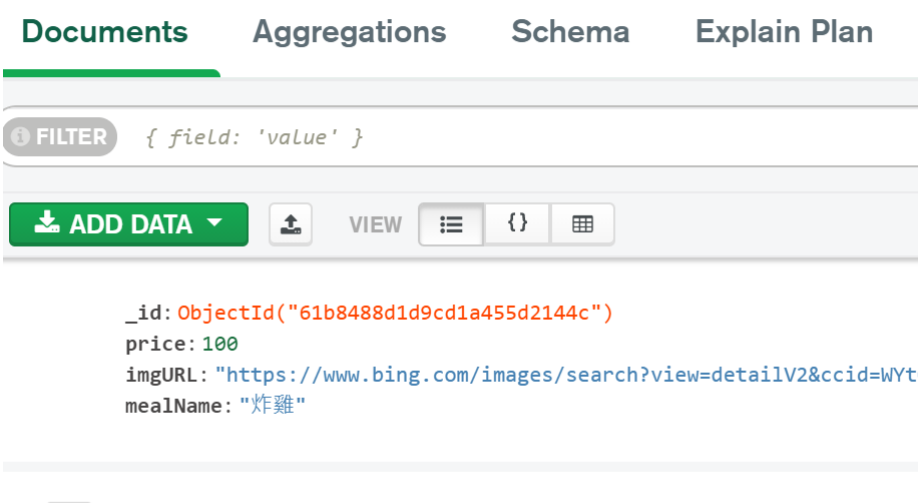



```
_id: ObjectId("61bab4b92bb7d83f5348707d")
restaurantID: ObjectId("61b84a4e1d9cd1a455d21454")
bossID: ObjectId("61b847191d9cd1a455d2143f")
```

```
_id: ObjectId("61c35dcbb164078183bace99")
bossID: ObjectId("61c35dcab164078183bace98")
restaurantID: ObjectId("61c35dcab164078183bace97")
```

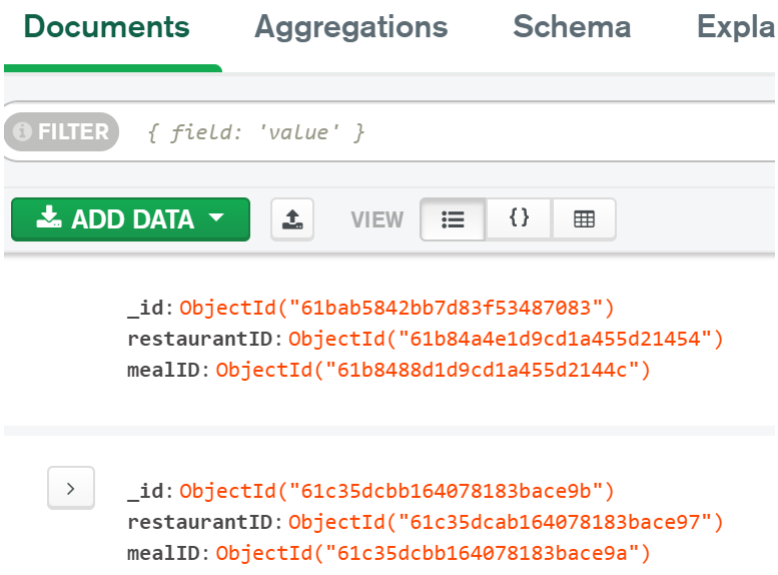
Owner

Order.meal



Meal

Order.dish

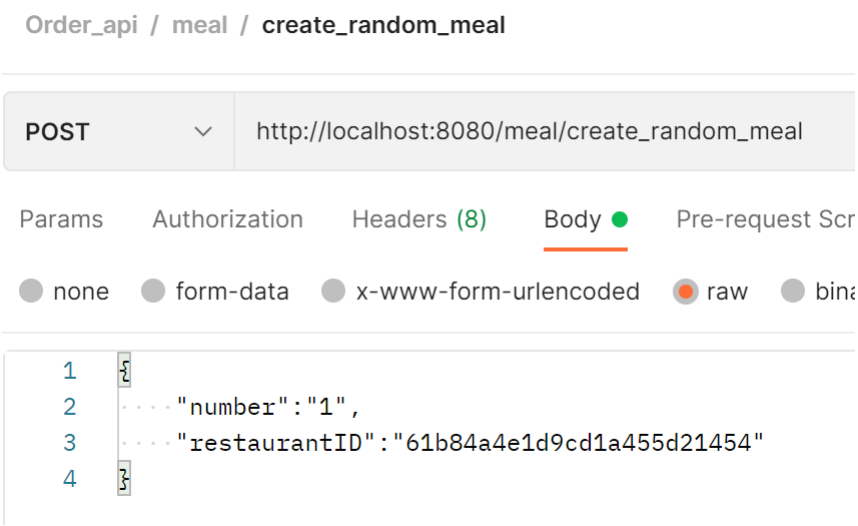


Dish

4. meal

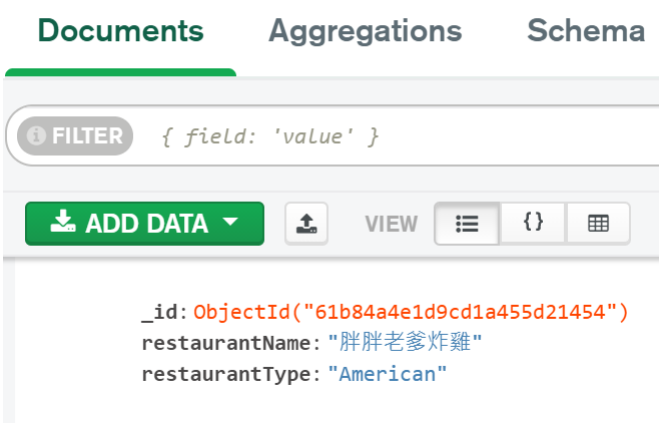
指定一間餐廳，生成指定數量的 meal

```python
# data set 給定一個餐廳, 隨機產生 meal
@api_view(['POST'])
def create_random_meal(request):
    body = JSONParser().parse(request)
    number = eval(body['number'])
    for _ in range(number):
        #新增 meal
        name = meal_generator.food_catcher()
        price = random.randrange(501)
        imgURL = meal_generator.rd_img()
        meal = {
            'mealName': name,
            'price': price,
            'imgURL':imgURL
        }
        meal_serializer = serializers.MealSerializer(data = meal)
        if meal_serializer.is_valid():
            meal_serializer.save()
        restaurantID = body['restaurantID']
        mealID = meal_serializer.data.get('id')
        dish = {
            'restaurantID':restaurantID,
            'mealID':mealID
        }
        dish_serializer = serializers.DishSerializer(data = dish)
        if dish_serializer.is_valid():
            dish_serializer.save()
    #response可能需要更動
    return JsonResponse({'message':'create'},status=status.HTTP_201_CREATED)
```



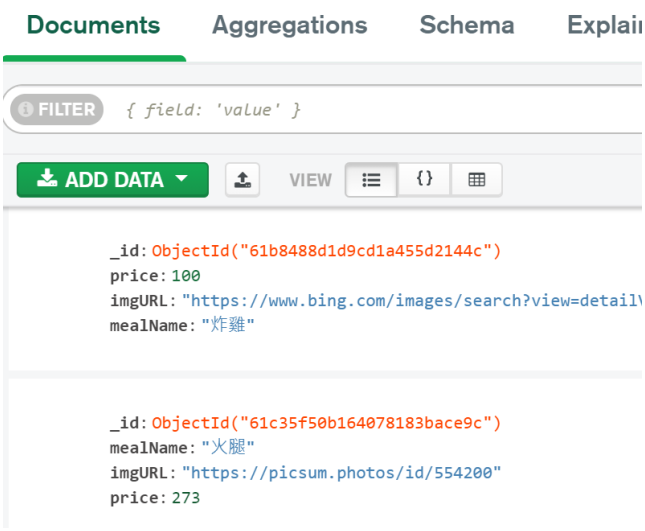
指定的餐廳為這間，生成一個隨機餐點

Order.restaurant



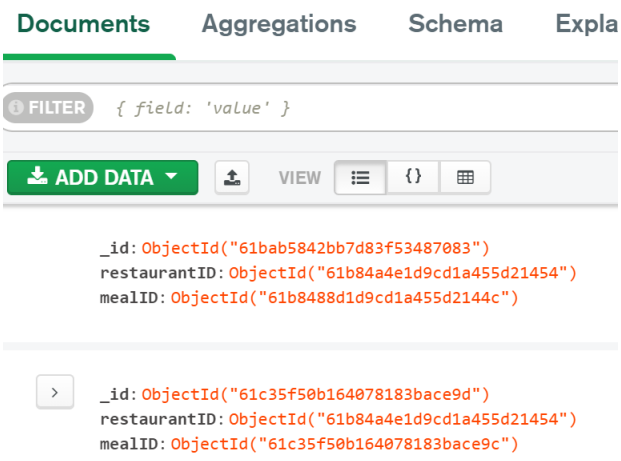
胖胖老爹炸雞

Order.meal



生成一個火腿

Order.dish



Dish

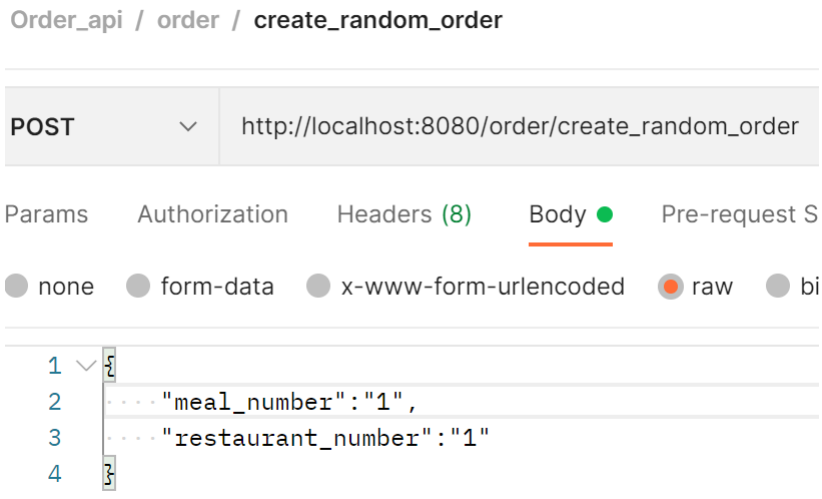
5. order

生成 order 會先生成一間 restaurant，接著生成一個 boss，生成 owner 的關係，接著生成指定數量的 meal，生成 dish 關係，每個 meal 生成 1~2 個 customer 建立 order，每個 order 生成 receipt, money。

```
12  @api_view(['POST'])
13  def create_random_order(request):
14      body = JSONParser().parse(request)
15      meal_number = eval(body['meal_number'])
16      restaurant_number = eval(body['restaurant_number'])
17      #新增 restaurant
18      for _ in range(restaurant_number):
19          restaurant_name = restaurant_generator.restaurant_name_generator()
20          restaurant_type = restaurant_generator.restaurant_type_generator()
21          restaurant = {
22              'restaurantName':restaurant_name,
23              'restaurantType':restaurant_type
24          }
25          restaurant_serializer = serializers.RestaurantSerializer(data = restaurant)
26          if restaurant_serializer.is_valid():
27              restaurant_serializer.save()
28          restaurantID = restaurant_serializer.data.get('id')
29          #新增 boss
30          boss = {
31              "bossName":customer_generator.customer_name_generator()#因為跟顧客名字沒有
32          }
33          boss_serializers = serializers.BossSerializer(data=boss)
34          if boss_serializers.is_valid():
35              boss_serializers.save()
36          bossID = boss_serializers.data.get('id')
37          #新增 owner
38          owner = {
39              "bossID":bossID,
40              "restaurantID":restaurantID
41          }
42          owner_serializers = serializers.OwnerSerializer(data=owner)
43          if owner_serializers.is_valid():
44              owner_serializers.save()
```

```
        #新增 meal
        for _ in range(meal_number):
            meal_name = meal_generator.food_catcher()
            price = random.randrange(501)
            imgURL = meal_generator.rd_img()
            meal = {
                'mealName': meal_name,
                'price': price,
                'imgURL':imgURL
            }
            meal_serializer = serializers.MealSerializer(data = meal)
            if meal_serializer.is_valid():
                meal_serializer.save()
            mealID = meal_serializer.data.get('id')
            dish = {
                'restaurantID':restaurantID,
                'mealID':mealID
            }
            dish_serializer = serializers.DishSerializer(data = dish)
            if dish_serializer.is_valid():
                dish_serializer.save()
            #新增 random 1~2 customer 點餐
            for _ in range(random.randrange(1,3)):
                customer_name = customer_generator.customer_name_generator()
                customer_phone = customer_generator.phone_generator()
                customer = {
                    'customerName':customer_name,
                    'phone':customer_phone
                }
                customer_serializers = serializers.CustomerSerializer(data = customer)
                if customer_serializers.is_valid():
                    customer_serializers.save()
                customerID = customer_serializers.data.get('id')
                #新增 order
                count = random.randrange(1,4)
```

```
                total_sum = count * price
                order_date = order_generator.random_date()
                order = {
                    'totalSum' :total_sum,
                    'count':count,
                    'orderDate':order_date
                }
                order_serializers = serializers.OrderSerializer(data= order)
                if order_serializers.is_valid():
                    order_serializers.save()
                orderID = order_serializers.data.get('id')
                #新增 reciept
                reciept = {
                    "restaurantID":restaurantID,
                    "customerID":customerID,
                    "orderID":orderID,
                    "mealID":mealID
                }
                reciept_serializers = serializers.RecieptSerializer(data=reciept)
                if reciept_serializers.is_valid():
                    reciept_serializers.save()
                #新增 money
                money = {
                    "bossID":bossID,
                    "restaurantID":restaurantID,
                    "orderID":orderID
                }
                money_serializers = serializers.MoneySerializer(data=money)
                if money_serializers.is_valid():
                    money_serializers.save()
    #response可能需要更動
    return JsonResponse({'message':'create'},status=status.HTTP_201_CREATED)
```



生成一間 restaurant，每間生成一個 meal

Order.restaurant

Documents Aggregations Schema E

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61b84a4e1d9cd1a455d21454")
restaurantName: "胖胖老爹炸雞"
restaurantType: "American"
```

```
_id: ObjectId("61c361ca613213e96075ad34")
restaurantName: "yItis"
restaurantType: "Chinese"
```

Restaurant

Order.boss

Documents Aggregations Schema

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61b847191d9cd1a455d2143f")
name: "scott"
```

```
_id: ObjectId("61c361ca613213e96075ad35")
bossName: "uTvJOzv"
```

Boss

Order.owner

Documents Aggregations Schema Expla

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61bab4b92bb7d83f5348707d")
restaurantID: ObjectId("61b84a4e1d9cd1a455d21454")
bossID: ObjectId("61b847191d9cd1a455d2143f")
```

```
_id: ObjectId("61c361ca613213e96075ad36")
bossID: ObjectId("61c361ca613213e96075ad35")
restaurantID: ObjectId("61c361ca613213e96075ad34")
```

Owner

Order.meal

Documents Aggregations Schema Explair

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61b8488d1d9cd1a455d2144c")
price: 100
imgURL: "https://www.bing.com/images/search?view=detailV
mealName: "炸雞"
```

```
_id: ObjectId("61c361ca613213e96075ad37")
mealName: "手捲"
imgURL: "https://picsum.photos/id/656200"
price: 244
```

Meal

Order.dish

Documents Aggregations Schema Exp

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61bab5842bb7d83f53487083")
restaurantID: ObjectId("61b84a4e1d9cd1a455d21454")
mealID: ObjectId("61b8488d1d9cd1a455d2144c")

_id: ObjectId("61c361ca613213e96075ad38")
restaurantID: ObjectId("61c361ca613213e96075ad34")
mealID: ObjectId("61c361ca613213e96075ad37")
```

Dish

Order.customer

Documents Aggregations Schema

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61b848121d9cd1a455d21447")
customerName: "geroge"
phone: "911111111"

_id: ObjectId("61c361ca613213e96075ad39")
customerName: "JOsvgrR"
phone: "9984304865"

_id: ObjectId("61c361cb613213e96075ad3d")
customerName: "oaYfCRQ"
phone: "0441055514"
```

兩個 customer

Order.order

Documents Aggregations Schema Ex

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61b849cb1d9cd1a455d21450")
totalSum: 500
count: 5
orderDate: 2021-12-14T16:00:00.000+00:00

_id: ObjectId("61c361cb613213e96075ad3a")
totalSum: 732
count: 3
orderDate: 2019-11-09T00:00:00.000+00:00

_id: ObjectId("61c361cb613213e96075ad3e")
totalSum: 732
count: 3
orderDate: 2021-01-03T00:00:00.000+00:00
```

一人一個 order

Order.reciept

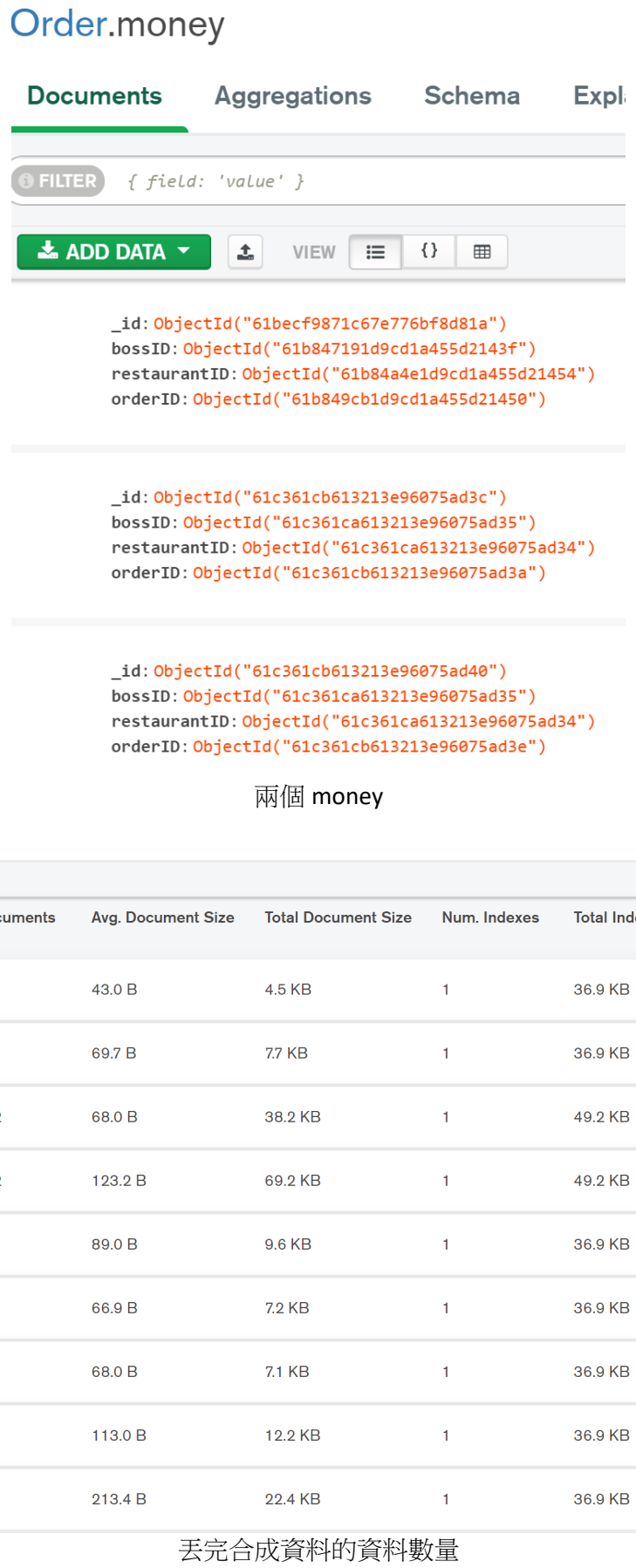
Documents Aggregations Schema Expla

FILTER { field: 'value' }

ADD DATA VIEW

```
_id: ObjectId("61bab6992bb7d83f53487089")
restaurantID: ObjectId("61b84a4e1d9cd1a455d21454")
customerID: ObjectId("61b848121d9cd1a455d21447")
orderID: ObjectId("61b849cb1d9cd1a455d21450")
mealID: ObjectId("61b8488d1d9cd1a455d2144c")

_id: ObjectId("61c361cb613213e96075ad3b")
restaurantID: ObjectId("61c361ca613213e96075ad34")
customerID: ObjectId("61c361ca613213e96075ad39")
orderID: ObjectId("61c361cb613213e96075ad3a")
mealID: ObjectId("61c361ca613213e96075ad37")

_id: ObjectId("61c361cb613213e96075ad3f")
restaurantID: ObjectId("61c361ca613213e96075ad34")
customerID: ObjectId("61c361cb613213e96075ad3d")
orderID: ObjectId("61c361cb613213e96075ad3e")
mealID: ObjectId("61c361ca613213e96075ad37")
```

兩個 receipt



兩個 money

Collections

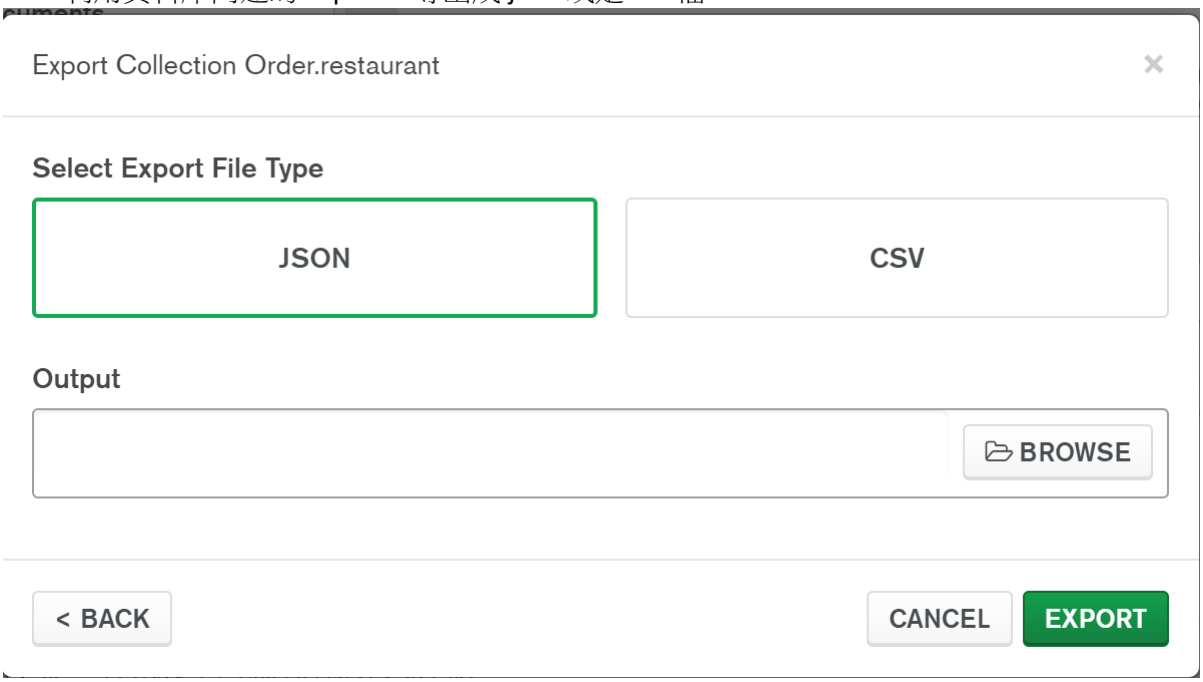
CREATE COLLECTION

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
boss	105	43.0 B	4.5 KB	1	36.9 KB	
customer	110	69.7 B	7.7 KB	1	36.9 KB	
dish	562	68.0 B	38.2 KB	1	49.2 KB	
meal	562	123.2 B	69.2 KB	1	49.2 KB	
money	108	89.0 B	9.6 KB	1	36.9 KB	
order	108	66.9 B	7.2 KB	1	36.9 KB	
owner	105	68.0 B	7.1 KB	1	36.9 KB	
reciept	108	113.0 B	12.2 KB	1	36.9 KB	
restaurant	105	213.4 B	22.4 KB	1	36.9 KB	

丟完合成資料的資料數量

導出資料

利用資料庫內建的 export，導出成 json 或是 csv 檔

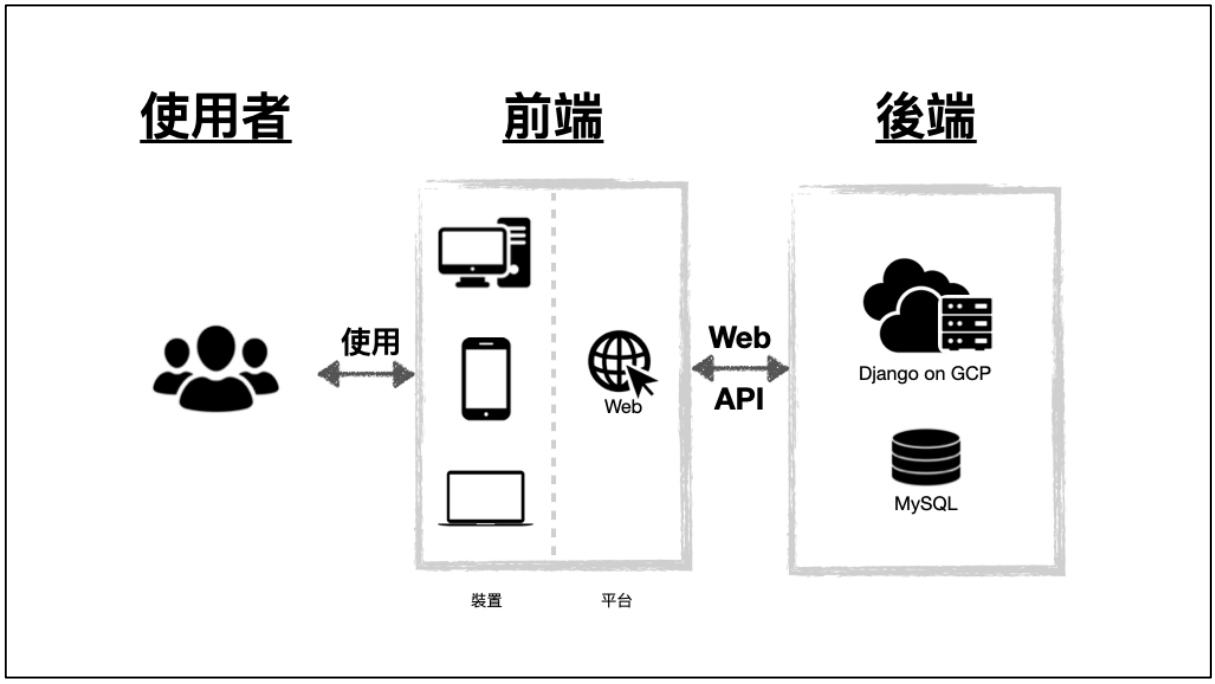


Outcome

Demo Website: https://simple-order-site.vercel.app/

網頁架構

使用前後端分離架構，讓前端對後端 API 發送 request。後端再回傳資料庫裡的 data 給前端，前端再把獲得的資料呈現給使用者。

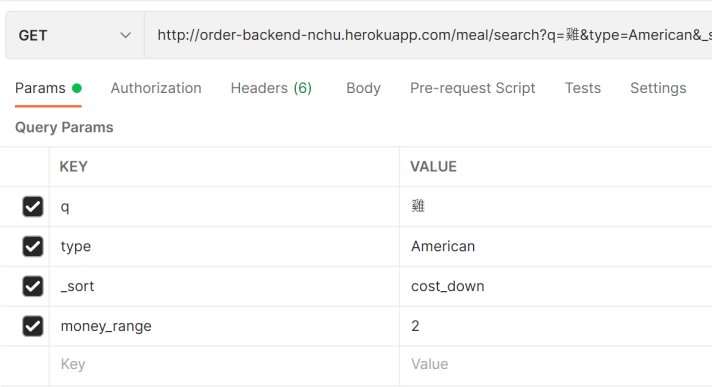


Query 使用介紹

1. **搜尋餐點**

   搜尋條件：

   - 搜尋文字
   - 餐點種類分類
   - 排列方式
   - 金額範圍



Pretty Raw Preview Visualize JSON


```
[
    {
        "id": "61c8347dd31a6749c0abfbbf",
        "mealName": "鹽酥雞",
        "restName": "TdAxT",
        "cost": 104,
        "url": "https://picsum.photos/id/441200"
    },
    {
        "id": "61c8347cd31a6749c0abfba5",
        "mealName": "鹽酥雞",
        "restName": "IIQoG",
        "cost": 147,
        "url": "https://picsum.photos/id/148200"
    }
]
```


```
if restaurant_type != "":
    restaurant = models.Restaurant.objects.filter(restaurantType = restaurant_type)
else :
    restaurant = models.Restaurant.objects.all()

filter_dish = models.Dish.objects.filter(restaurantID__in=list(restaurant.values_list('id')))
all_meal = models.Meal.objects.filter(id__in=filter_dish.values_list('mealID'))

if q !="":
    all_meal = all_meal.filter(mealName__contains = q)
if money_range == "1":
    all_meal = all_meal.filter(price__lte = 100)
elif money_range == "2":
    all_meal = all_meal.filter(price__lte = 250).filter(price__gt = 100)
elif money_range == "3":
    all_meal = all_meal.filter(price__lte = 500).filter(price__gt = 250)

if sort =="cost_up":
    all_meal_list = list(all_meal.order_by('price'))
elif sort =="cost_down":
    all_meal_list = list(all_meal.order_by('-price'))
```

2. **排名受歡迎餐點名稱**

   讓使用者能看到受歡迎的餐點排名。

GET http://order-backend-nchu.herokuapp.com/meal/popularity

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
Key	Value

Pretty Raw Preview Visualize JSON


```
[
    {
        "mealName": "炸雞",
        "shop": "胖胖老爹炸雞",
        "rank": 1,
        "click": 5
    },
    {
        "mealName": "美味享拼法士達",
        "shop": "Chili's 美式餐廳",
        "rank": 2,
        "click": 4
    },
    {
        "mealName": "涼麵",
        "shop": "CqdpS",
        "rank": 3,
        "click": 2
```


```
for i in range(len(all_meal_list)):
    count = models.Reciept.objects.filter(mealID = all_meal_list[i].id).count()
    meal_count[str(all_meal_list[i].id)] = count
result = sorted(meal_count.items(),key = lambda meal_count:meal_count[1],reverse=True)
rank_count = 1
number_prv = result[0][1]
for i in range(len(result)):
    number_now = result[i][1]
    if number_now == number_prv :
        rank = rank_count
    else :
        rank = rank_count + 1
        rank_count = rank_count + 1
        number_prv = number_now
    meal_id = result[i][0]
```

3. **得到餐廳收入**

老闆端能看到每間餐廳的收入狀況。

GET http://order-backend-nchu.herokuapp.com/order/income

Params Authorization Headers (6) Body Pre-request Script Tests

Query Params

KEY	VALUE
Key	Value

Pretty Raw Preview Visualize JSON


```
[
    {
        "shop": "胖胖老爹炸雞",
        "income": 800
    },
    {
        "shop": "Chili's 美式餐廳",
        "income": 2820
    },
    {
        "shop": "TGI Fridays",
        "income": 1000
    },
    {
        "shop": "費城美式餐廳",
        "income": 1140
    },
```


```
for i in range(len(all_restaurant_list)):
    money_list = list(models.Money.objects.filter(restaurantID = all_restaurant_list[i].id))
    income = 0
    for j in range(len(money_list)):
        try:
            income += models.Order.objects.get(id = money_list[j].orderID).totalSum
        except:
            print(money_list[j].orderID)
    result.append({"shop" : all_restaurant_list[i].restaurantName,"income" : income,})
```

4. **取得歷史收據**

讓使用者得到以前的訂單資訊。

GET http://order-backend-nchu.herokuapp.com/order/reciept?name=geroge&phone=911111111

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESC
name	geroge	
phone	911111111	

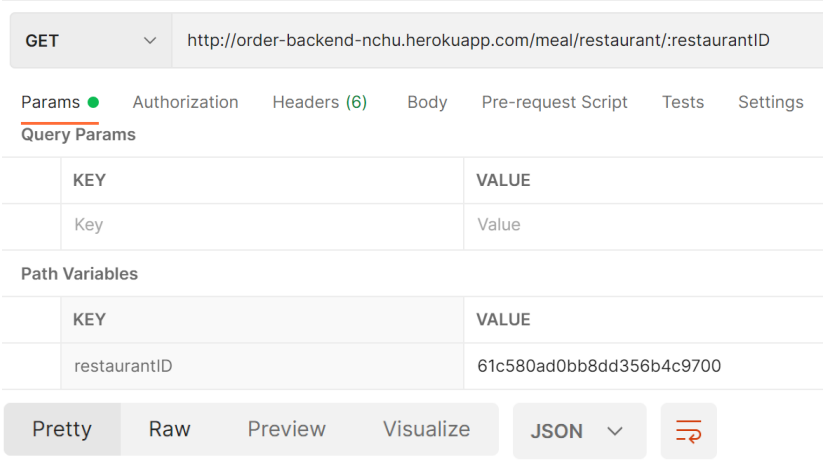
Pretty Raw Preview Visualize JSON

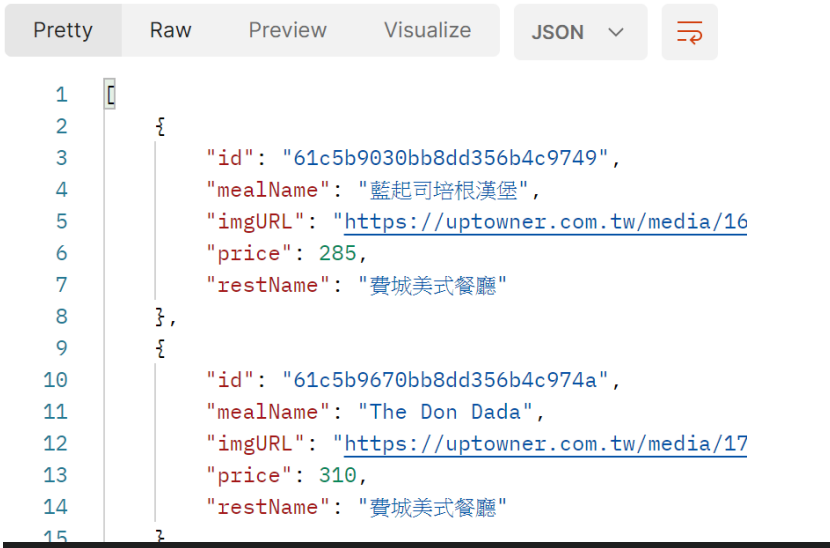

```
[
    {
        "date": "2021-12-14",
        "shopList": [
            "胖胖老爹炸雞"
        ],
        "costList": [
            500
        ],
        "mealList": [
            "炸雞"
        ]
    },
    {
        "date": "2021-12-30",
```

```
customer = list(models.Customer.objects.filter(customerName = customer_name,phone = customer_phone))[0]
reciept = list(models.Reciept.objects.filter(customerID = customer.id))
order_dict = {}
result = []
for i in reciept:
    try:
        order_id = i.orderID
        order = models.Order.objects.get(id = order_id)
        order_date = order.orderDate
        order_dict[order_id] = order_date
    except:
        print("error")
sort_list = sorted(order_dict.items(),key = lambda sort_list:sort_list[1])
```

5. **取得一間餐廳的所有餐點**

讓使用者得到一間餐廳的所有餐點。





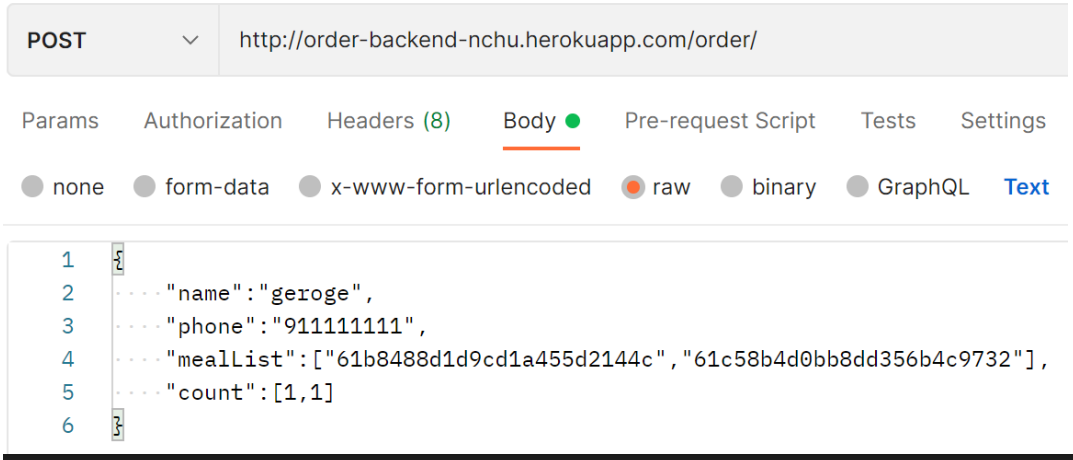

```
dish_list = list(models.Dish.objects.filter(restaurantID = restaurant_id))
meal_list = []
for i in range(len(dish_list)):
    meal_serializer = dict(serializers.MealSerializer(models.Meal.objects.get(id = dish_list[i].mealID)).data)
    meal_serializer['restName'] = models.Restaurant.objects.get(id = restaurant_id).restaurantName
    meal_list.append(meal_serializer)
```

6. 下訂單

訂購人資訊

訂購餐點

訂購數量




```
dish = models.Dish.objects.get(mealID = meal_list[i])
restaurant_id = dish.restaurantID
customer = list(models.Customer.objects.filter(customerName = customer_name,phone = customer_phone))
customer_id = customer[0].id
meal = models.Meal.objects.get(id = meal_list[i])
meal_price = meal.price
owner = models.Owner.objects.get(restaurantID = restaurant_id)
boss_id = owner.bossID
order = {
    "count":count_list[i],
    "totalSum":count_list[i] * meal_price,
}
order_serializer = serializers.OrderSerializer(data = order)
if order_serializer.is_valid():
    order_serializer.save()
    result.append(order_serializer.data)
else:
    print("order error",order_serializer.data)

order_id = order_serializer.data.get("id")

reciept = {
    "mealID":meal_list[i],
    "restaurantID":restaurant_id,
    "orderID":order_id,
    "customerID":customer_id
}
reciept_serializer = serializers.RecieptSerializer(data = reciept)
if reciept_serializer.is_valid():
    reciept_serializer.save()
else:
    print("reciept error",reciept_serializer.data)
```