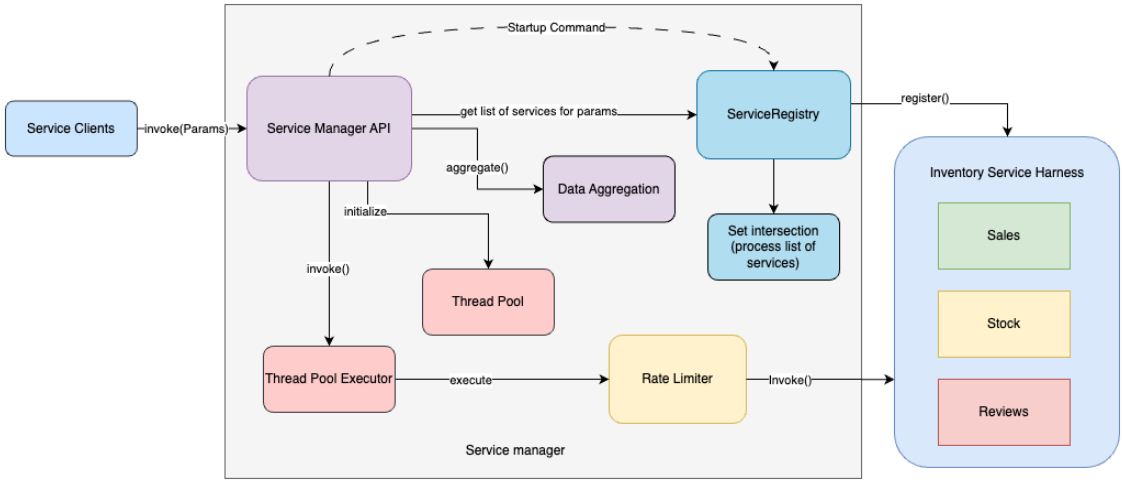
Service Manager



This project requires local installations of the following

- [Java](version > 11)
- Maven
- Postman

Table of Content

About

This prototype implementation of a service manager has the following parts

- 1. A test harness
- 2. The Service Manager framework
- 3. A postman collection

Test Harness

This test harness is an inventory api which provides service endpoints with data loaded from json files for stocks, sales and reviews.

For example testing the service manager with, - http://localhost:9090/invoke?brand=iphone gives the stocks, sales, reviews for iphone - http://localhost:9090/invoke?brand=samsung&date=__fromDate:01-10-2023 gives the stocks from the first of october - http://localhost:9090/invoke/aggregateByModel?brand=iphone, gives the aggregated sales and stock data for iphone

All the above test urls are configured in the given postman collection with visualization scripts.

Implementation details of the test harness are inside the `InventoryServiceHarness` folder

`Build and start the Inventory Service Harness`

- The inventory service starts in port 8080

```
cd InventoryServiceHarness

mvn package

java -jar target/InventoryApi-0.0.1-SNAPSHOT.jar
```

Service Manager

The Service Manager is implemented as a Spring Boot application, more details on the requirements and design are in the `ServiceManager` folder. This design looks at the solution to identify services from params as a `set intersection` problem. There are various set intersection algorithms discussed like List, BitMap in the README.md of the ServiceManager project

`Build and start the Service Manager`

- The service manager starts in port 9090 and is preconfigured with endpoints of the inventory harness. The inventory service must be started for the service manager to start successfully

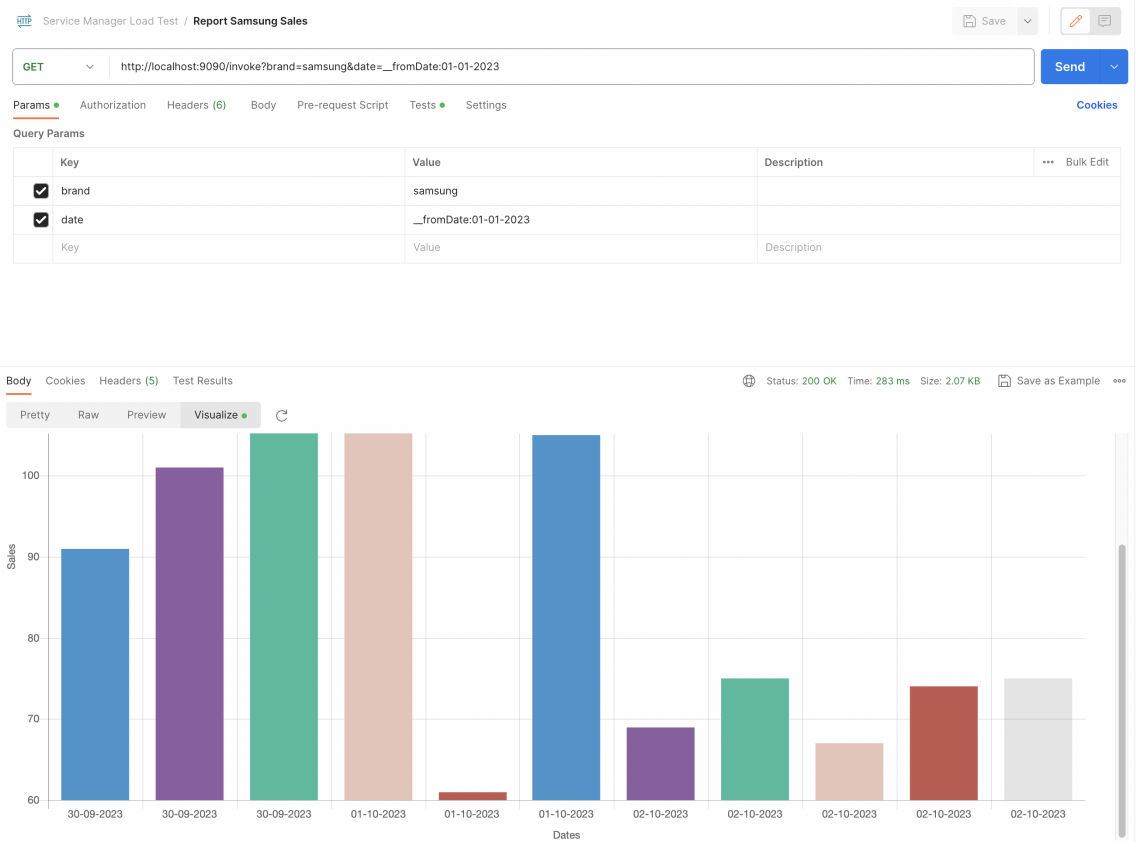
```
cd ServiceManager

mvn package

java -jar target/ServiceManager-0.0.1-SNAPSHOT.jar
```

Postman Collection

The postman collection has test scripts with params for category, model, brand, zip, sales, sales amount, available stock. It can test the service manager and visualize from the results, it also has a load test suite

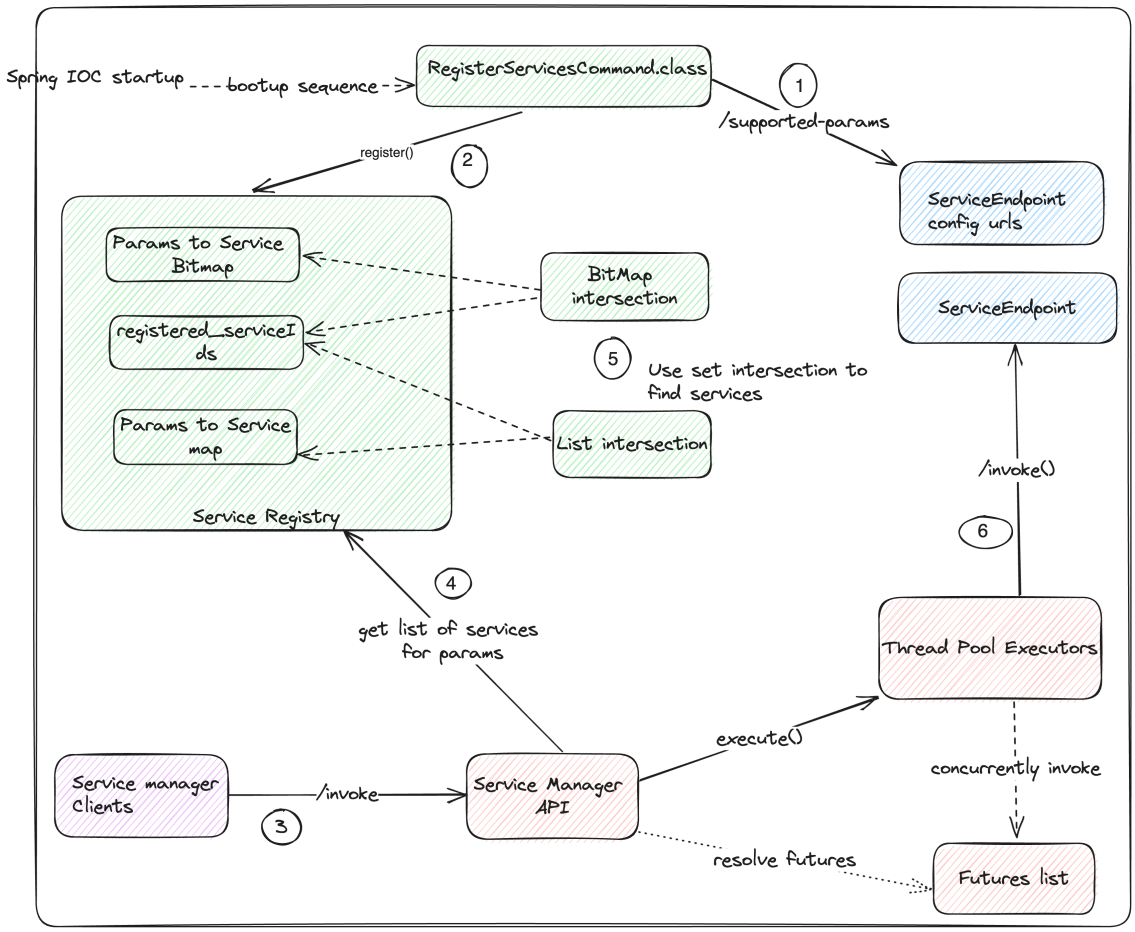
`Test with the given postman collection`

Import the collection into postman and test the given APIs


Service Manager Load Test / Report Samsung Sales
Save
GET
http://localhost:9090/invoke?brand=samsung&date=__fromDate:01-01-2023
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
brand
samsung
date
__fromDate:01-01-2023
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 283 ms
Size: 2.07 KB
Save as Example
Pretty
Raw
Preview
Visualize
Sales
100
90
80
70
60
30-09-2023
30-09-2023
30-09-2023
01-10-2023
01-10-2023
01-10-2023
02-10-2023
02-10-2023
02-10-2023
02-10-2023
02-10-2023
Dates

Service Manager Design and Implementation

Design



- The service manager starts in port 9090 and is preconfigured with endpoints of the inventory harness. The inventory service must be started for the service manager to start successfully

```
cd ServiceManager

mvn package

java -jar target/ServiceManager-0.0.1-SNAPSHOT.jar
```

Service Registry

Service Registry registers all the services and initializes maps that connect the supported params to the services, which are later used for set intersection.

- Parameters to List of ServiceIds map
- Parameters to Services Bitmap

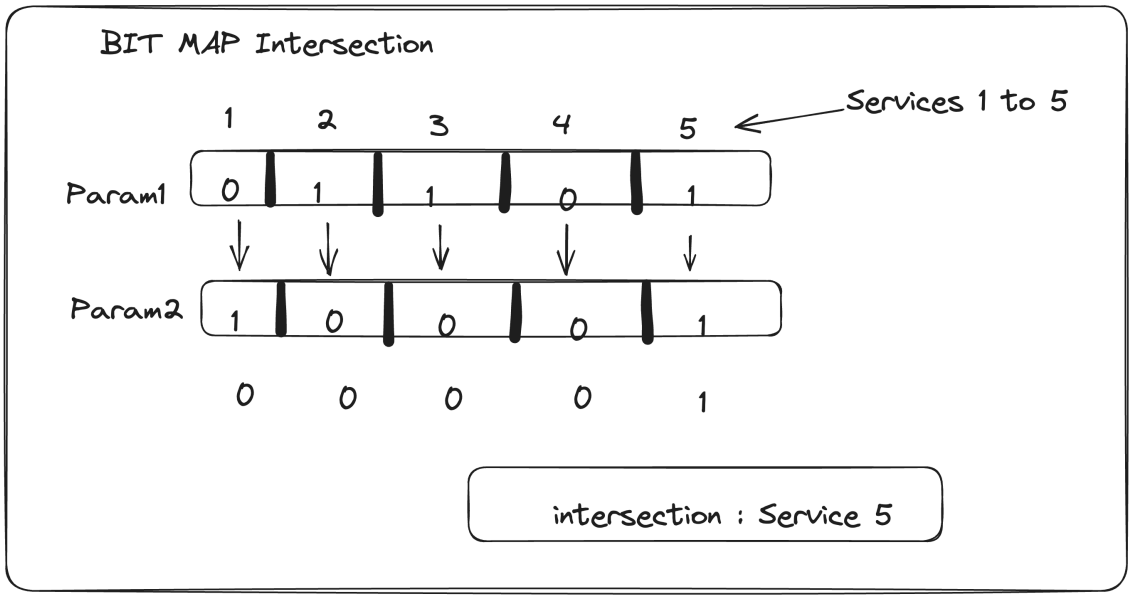
This implementation looks the core of the solution to identify services from params as a Set Intersection problem.

List Intersection

When a list of params are sent for querying, the java collections framework is utilized to find the intersection of all serviceIds

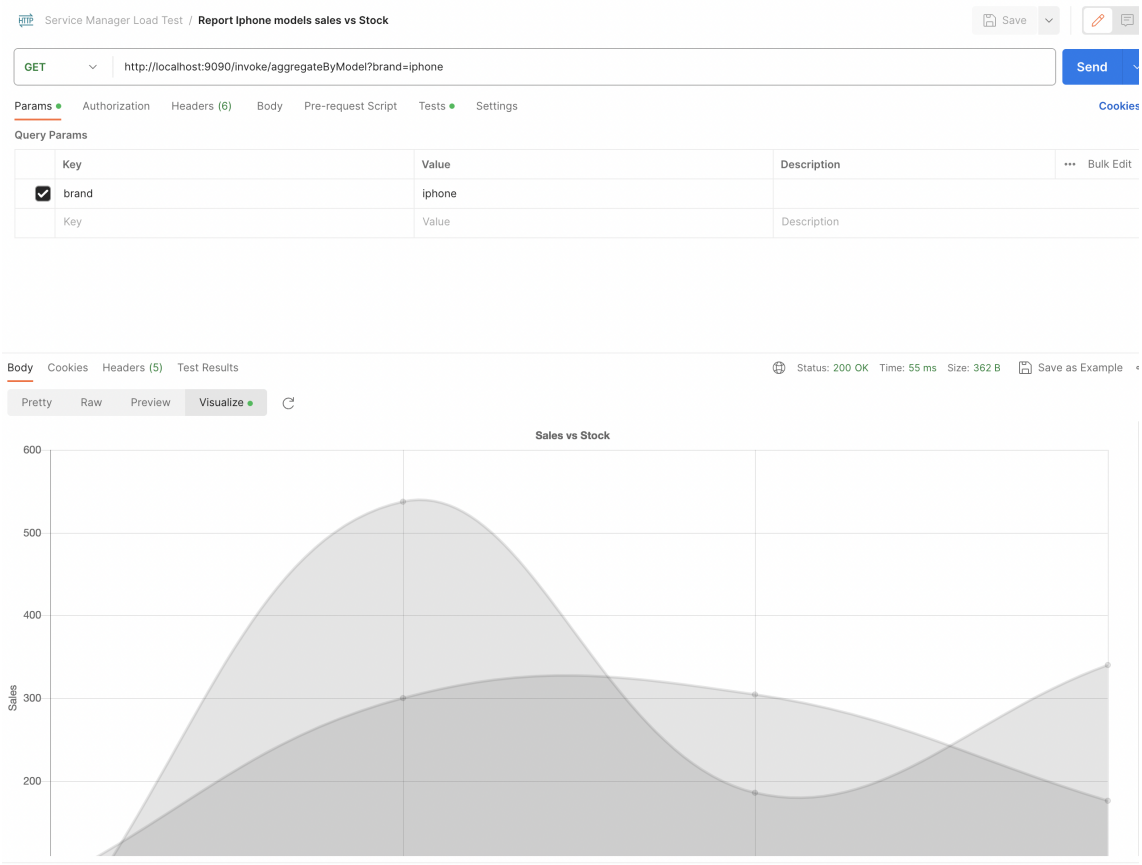
BitMap Intersection

When a list of params are sent for querying, a BitMap utility is used to find the intersection of all serviceIds



Aggregation

A custom aggregation endpoint is included in the ServiceManager to demo on data aggregation. It is not a generic implementation and is very demo specific.



Endpoint call limit

Every endpoint has a concurrency limit, this implementation uses one local semaphore for every registered endpoint. But in a distributed scenario where multiple service managers are expected global counters should also be used.

Concurrency

The endpoints are to be invoked concurrently for managing call latency. A thread pool is used to assign executors and wait for turns using futures.

Test Harness

Starting The Application

This test harness is an inventory api which provides service endpoints with data loaded from json files for stocks, sales and reviews

Implementation details of the test harness are inside the `InventoryServiceHarness` folder

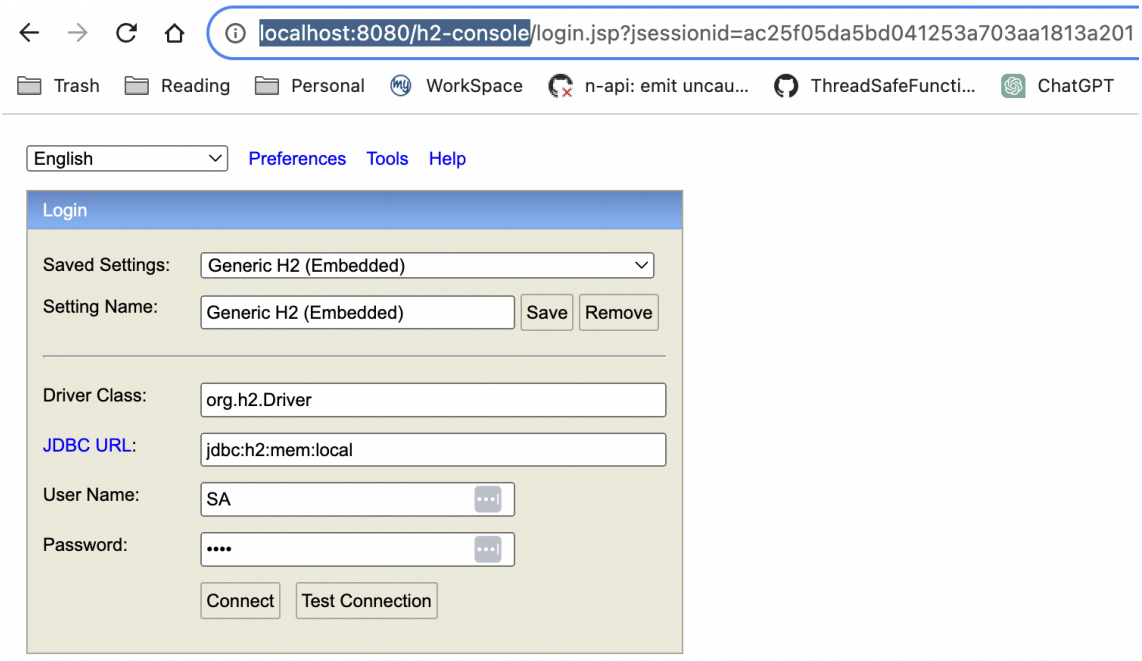
`Build and start the Inventory Service Harness`

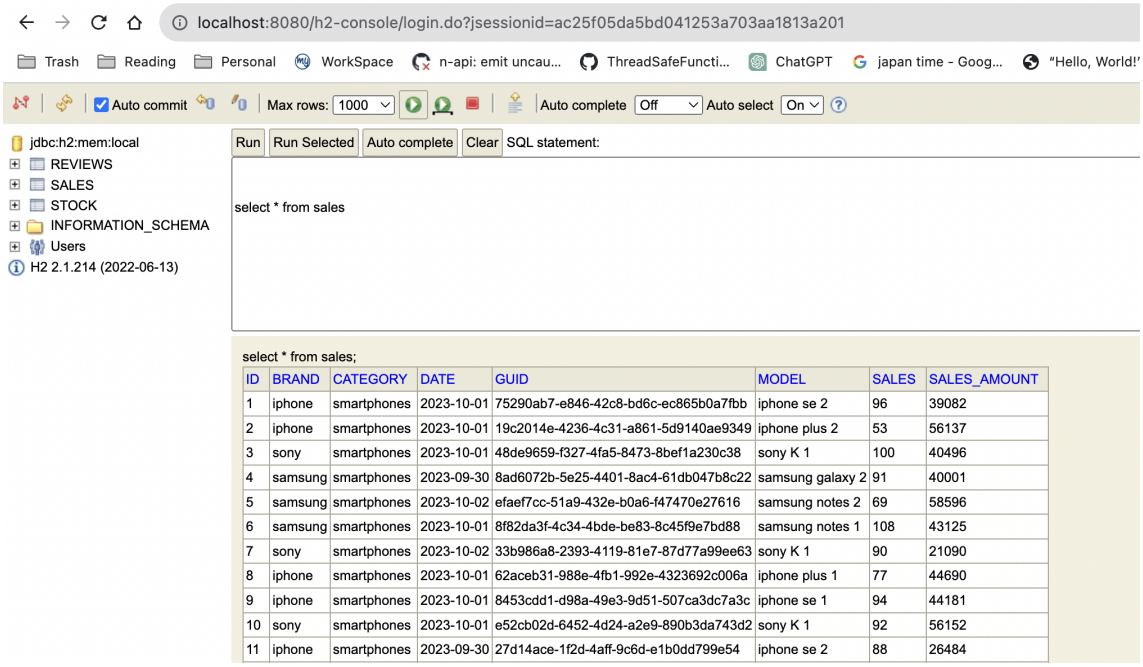
- The inventory service starts in port 8080

```
cd InventoryServiceHarness

mvn package

java -jar target/InventoryApi-0.0.1-SNAPSHOT.jar
```

Database

This test harness uses a H2 database to act as a replacement for an actual database. Use http://localhost:8080/h2-console for logging in and running queries against the stock, sales and reviews data loaded. Default password is `pass`





ID	BRAND	CATEGORY	DATE	GUID	MODEL	SALES	SALES_AMOUNT
1	iphone	smartphones	2023-10-01	75290ab7-e846-42c8-bd6c-ec865b0a7fbb	iphone se 2	96	39082
2	iphone	smartphones	2023-10-01	19c2014e-4236-4c31-a861-5d9140ae9349	iphone plus 2	53	56137
3	sony	smartphones	2023-10-01	48de9659-f327-4fa5-8473-8bef1a230c38	sony K 1	100	40496
4	samsung	smartphones	2023-09-30	8ad6072b-5e25-4401-8ac4-61db047b8c22	samsung galaxy 2	91	40001
5	samsung	smartphones	2023-10-02	efaef7cc-51a9-432e-b0a6-f47470e27616	samsung notes 2	69	58596
6	samsung	smartphones	2023-10-01	8f82da3f-4c34-4bde-be83-8c45f9e7bd88	samsung notes 1	108	43125
7	sony	smartphones	2023-10-02	33b986a8-2393-4119-81e7-87d77a99ee63	sony K 1	90	21090
8	iphone	smartphones	2023-10-01	62aceb31-988e-4fb1-992e-4323692c006a	iphone plus 1	77	44690
9	iphone	smartphones	2023-10-01	8453cdd1-d98a-49e3-9d51-507ca3dc7a3c	iphone se 1	94	44181
10	sony	smartphones	2023-10-01	e52cb02d-6452-4d24-a2e9-890b3da743d2	sony K 1	92	56152
11	iphone	smartphones	2023-09-30	27d14ace-1f2d-4aff-9c6d-e1b0dd799e54	iphone se 2	88	26484

Test Data

Test data was generated from https://json-generator.com/ , generated files are placed under resource and are loaded into the H2 database during startup

Sales :

```
[
  '{{repeat(30)}}',
  {
    guid: '{{guid()}}',
    category: "smartphones",
    brand: '{{random("iphone", "samsung", "sony")}}',
    model: function (rand) {
      var model_map = {
        iphone: ['plus', 'se'],
        samsung: ['galaxy', 'notes'],
        sony: ['M', 'K']
      };
      return this.brand + ' ' + model_map[this.brand][rand.integer(0, 1)] + ' ' +
rand.integer(1, 2);
    },
    salesAmount: '{{integer(20000, 60000)}}',
    sales: '{{integer(45, 120)}}',
    date: '{{ date(new Date(2023, 09, 1), new Date(2023, 09, 4), "dd-MM-YYYY")}}'
  }
]
```

Stock :

```
[
  '{{repeat(30)}}',
  {
    guid: '{{guid()}}',
    category: "smartphones",
    brand: '{{random("iphone", "samsung", "sony")}}',
    model: function (rand) {
      var model_map = {
        iphone: ['plus', 'se'],
        samsung: ['galaxy', 'notes'],
        sony: ['M', 'K']
      };
      return this.brand + ' ' + model_map[this.brand][rand.integer(0, 1)] + ' ' +
rand.integer(1, 2);
    },
    availableStock: '{{integer(20000, 60000)}}',
    releaseDate: '{{ date(new Date(2023, 09, 1), new Date(2023, 09, 4), "dd-MM-
YYYY")}}'
  }
]
```

Reviews :

```
[
  '{{repeat(30)}}',
  {
    guid: '{{guid()}}',
    category: "smartphones",
    brand: '{{random("iphone", "samsung", "sony")}}',
    model: function (rand) {
      var model_map = {
        iphone: ['plus', 'se'],
        samsung: ['galaxy', 'notes'],
        sony: ['M', 'K']
      };
      return this.brand + ' ' + model_map[this.brand][rand.integer(0, 1)] + ' ' +
rand.integer(1, 2);
    },
    rating: '{{floating(5, 10, 1)}}',
    comment: '{{random("manageable", "good", "ok")}}'
  }
]
```