Using Microservices Backend Architecture

Sergiu Marcus

This system was built to show the use of the microservices architecture and it's presenting an easy way to find restaurants and see what kind of food they serv.
The system contains 2 major components, the client side, which is represented with a native Android app and a backend side, which is based on the microservices architecture.

Client-side app:
The app shows a list of available restaurants. When tapping on a restaurant in the list, a list of all the types of food that are served in that particular restaurant is shown. The app is built as a native Android app, using Kotlin and it is consuming the REST API exposed by the backend-side.
For the networking part of the application it was used Retrofit 2. Retrofit is a type-safe REST client for Android, Java and Kotlin developed by Square. The library provides a powerful framework for authenticating and interacting with APIs and sending network requests with OkHttp. This library makes downloading JSON or XML data from a web API fairly straightforward. Once the data is downloaded then it is parsed into a Plain Old Java Object (POJO) which must be defined for each "resource" in the response.

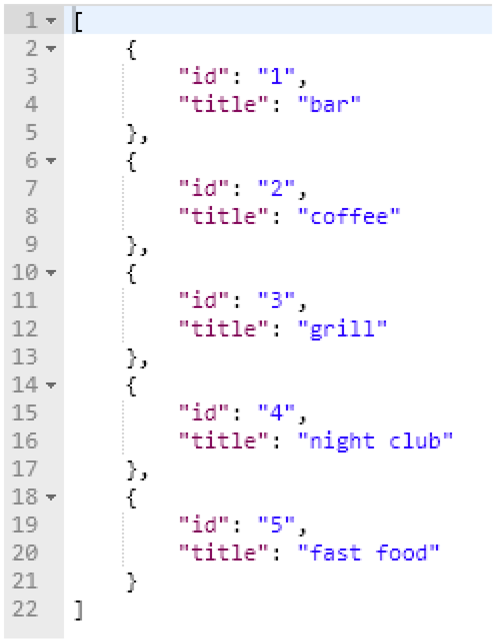
Backend side:
The backend side is build using a microservice architecture. This structures an application as a collection of services that are:

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities.

A microservice is a single self-contained unit which, together with many others, makes up a large application. By splitting your app into small units every part of it is independently deployable and scalable, can be written by different teams and in different programming languages and can be tested individually. The backend consists of two microservices, that can work independently, but also, they talk to one another to exchange the information they contain. The microservices are written in Node.JS. The microservices are:

Types Service

This service stores all the types as a list and exposes an *@GET* API endpoint in port 8080, where other clients can connect and request the list of types. The response sent is encoded as a JSON.



```
[
    {
        "id": "1",
        "title": "bar"
    },
    {
        "id": "2",
        "title": "coffee"
    },
    {
        "id": "3",
        "title": "grill"
    },
    {
        "id": "4",
        "title": "night club"
    },
    {
        "id": "5",
        "title": "fast food"
    }
]
```

Restaurants Service

This microservice has the task to keep a list of all the restaurants and also a list with the types that run in each restaurant. There is a mapping of the type id to the restaurant id. This microservice exposes two *@GET* API endpoints. One is for the restaurants list, which is returned directly as a JSON. The other endpoint is called to get a list of the types that run in the selected restaurant. This endpoint requires a parameter on the request which specifies the restaurant id, so it knows for what restaurant to compose the list. Then it filters the types list according to the ones that run in the selected restaurant, then returns the list with only the types for the selected restaurant.

Microservices architecture:

Access to the microservices is done through the cloud. Only the Restaurants Service is exposed publicly, the app will access only the API exposed by this one. The microservice for types is accessed only by the other microservice.

Containers

In order to have a complete solution for a Service Oriented System Architecture the microservices were delivered using containers and Docker.

Docker is a containerization platform that packages your application and all its dependencies together in the form of a docker container to ensure that your application works seamlessly in any environment.

Docker Container is a standardized unit which can be created on the fly to deploy a particular application or environment. It could be an Ubuntu container, CentOs container, etc. to full-fill the requirement from an operating system point of view.

Docker Image can be compared to a template which is used to create Docker Containers. They are the building blocks of a Docker Container. These Docker Images are created using the build command. These Read only templates are used for creating containers by using the run command.