International Bot

Choon Wei Tong	Shijie Liu	Simonas Ramonas
<17205786>	<18439314>	<18763829>

Synopsis:

Describe the system you intend to create:

A centralised cake shop which obtains prices of user customised cakes from different cake shops. Users can input their desired specifications for a cake and this system will quote prices from a selection of available cake shops. Each cake shop is made to be a service for a centralised managing application which can query the different prices from the cake shops. The system is made to be scalable as more cake shops can be added just by including a separate service for it.

This application will grab all the relevant information from the frontend and send it to a backend system as a JSON payload. The JSON payload is then collected and is converted into a Java object in a centralise service. Then, the object information is passed into the microservices that will return a given value which will be relayed to the centralised service and back to the frontend.

Technology Stack

List of the main distribution technologies used,

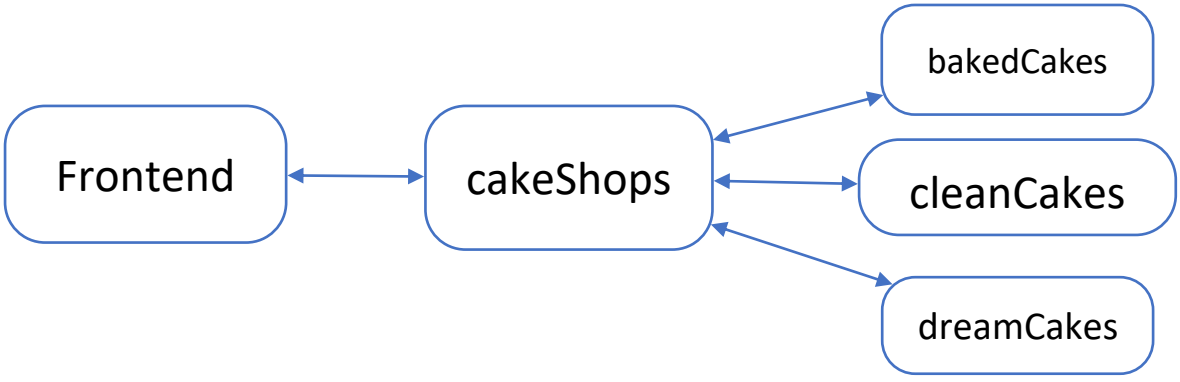
- *REST: The REST framework is used for distributing the services into microservices that can be accessed by a centralised application. REST framework allows each service to be accessible by the centralised application through a given URI. POST and GET methods allow data to be written and read in order to allow these services to communicate with each other.*

- *REACT JS: REACT is used to implement the frontend where it obtains user selected data in order to send it to the backend. JSON payload is generated from data obtained in the frontend and is sent to the backend that is implemented in Java.*

System Overview

Describe the main components of your system.

***Frontend** allows for user input and output. **CakeShops** is the middleman. bakedCakes, cleanCakes and dreamCakes are the **microservices**.*



Include your system architecture diagram in this section.

Explain how your system works based on the diagram.

***CakesShops** is the middleman that takes the user input from the frontend and distributes it to the microservices. The **microservices** (bakedCakes, cleanCakes, dreamCakes) receives the data from cakeShops, calculates price and sends back to cakeShops. The frontend then takes the prices from cakeShops and displays as an alert.*

Explain how your system is designed to support scalability and fault tolerance.

It is easy to add new microservices to the system and if some of the microservices are unavailable the system does not break and simply just shows the prices from the services that are available.

Contributions

Provide a sub section for each team member that describes their contribution to the project. Descriptions should be short and to the point.

Choon Wei Tong – Implemented the JavaScript frontend and wired it to the services on the backend. Implemented the middleware which collects all the relevant data from the frontend and sends it to all the microservices at the backend. Written one microservice called 'Clean Cakes'.

Shijie Liu 18439314 – Further developed the middleware allowing necessary data to be collected by the backend from services and passed to the frontend and displayed. Implemented microservice ' –Dream Cakes'.

Simonas Ramonas - Implemented microservice called 'Baked Cakes'.

Reflections

One of the challenges we first faced was how do we transfer data from frontend to backend. And we looked into React, where user input data is packaged as a json object and then sent to our Rest backend which can set to accept a json.

Another challenge we faced was an issue of serialisation on the List of cakeInvoices returned by the 3 services. Originally, we tried to return a clientApplication object to the frontend consisting of customer ID, cakeSpecs and a list of cakeInvoices generated by the microservices. But we kept failing to get the list (not even an empty list would appear) even though it exists in the backend. So, we changed the middleware to return a Gson instead which solved the issue.

What would you have done differently if you could start again?

The application still works even when not all services are running, but the error messages are not great. So, we could display the information better rather than using an alert box to tell the users what services are available and what aren't.

We could try to even let the user decide on what services (shops) they'd like to see the prices from rather than giving them the result of all services that's running.

What have you learnt about the technologies you have used? Limitations? Benefits?

Restful services are great as they are stateless, which means they have great fault tolerance since stateless components can be freely deployed if something fails. It can also be easily scaled whether it's adding more services

or accommodating load changes. Binding services through an API is very efficient and eliminates dependencies amongst each microservices. Error handling however can be a bit tricky as its hard to distinguish the issue from a response code.

ReactJs – is becoming more and more widely used when dealing with larger applications (scalable applications). The idea of reusable components saves a lot of time, and the use of virtual DOM makes the application much more responsive. The use a JavaScript library provides familiarity and flexibility to developers. However, it uses JSX allowing HTML and JS to be mixed together which can be a bit difficult to get used to.

Link to video: https://youtu.be/Rp5wiSIk5Vo