**Business scenario:**

FoodBear is a service based on FoodPanda’s business model that allows customers to look up restaurants that can deliver food right up to their doorstep.

**Existing Systems:**

* **Customer Order System (COS)** – This is a web-based application that will allow customers to log in, search for restaurants that can deliver to the region and then place orders based on the list of restaurants that are returned.
* **Restaurant Management System (RMS)** – This system allows the retrieval of a list of restaurants (and their corresponding menus) that can deliver to a specified location. It is connected to a MySQL restaurant database that stores restaurant details as well as menus offered by the restaurants
* **Customer Relation Management System (CRM)** – This system manages customer information, including personal details such as email and handphone number as well as the list of restaurants that the customer has previously ordered from. It is connected to a MySQL customer database.

**Technologies used:**

* **Electronic Messaging System (via Tibco EMS server)** – JMS was used to send data in between internal systems. JMS was used because the FoodBear application potentially serves a large number of customers concurrently, and using messaging internally will allow FoodBear to process a large number of requests quickly and reliably (using queues)
* **XML documents** –XML was selected as the data protocol as it is well structured and can be verified with an XML Schema Definition (XSD) to ensure that the XML being passed is valid.
* **Databases** – Databases were used to store restaurant information (name, location, available menus) as well as customer information (personal data, past orders). Databases were used because it was necessary to store persistent data as well as retrieve subsets of information based on specific criteria (eg. Query to retrieve restaurant information based on location) quickly (data is indexed). This is important because of the potentially large number of potential users.
* **Web services** – External web services were used to provide information (delivery time), as well as invoke API from external sources to meet business requirements (external restaurant’s ordering web services, PayPal’s payment services)
* **Integration Middleware (Tibco BusinessWorks)** – Tibco BusinessWorks was used as an integration platform to do data transformation and dynamic content-based routing
* **Plugins (Tibco BusinessWorks)** – An email plugin for Tibco BusinessWorks was used to send business process related information (acknowledgement of order and total time till delivery) to the customer

**Business Process walkthrough**

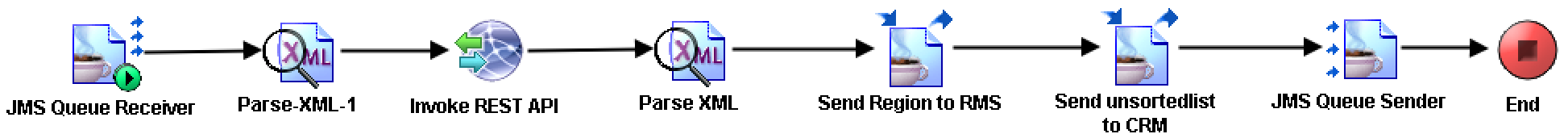
1. The business process starts when a customer logs onto the Customer Order System (COS) and enters a postal code.
2. The COS will send a synchronous request-reply JMS message to the Integration Middleware (IM) via a queue (q.request.search). The JMS message will contain the postal code and customer ID of the customer.
3. The IM will send a HTTP GET request to a Google web service to convert the region name of the postal code specified
4. The IM will then send a synchronous JMS message to the Restaurant Management System (RMS) via a queue (q.request.region). The JMS message will contain the region name.
5. The RMS will query the restaurant database to retrieve all restaurants that deliver to the specified region and return the list of restaurants to the IM via a queue (q.reply.region)
6. The IM will send the customer ID and restaurant list to the Customer Relation Management System (CRM) through a synchronous request-reply JMS message via a queue (q.request.sortedlist).
7. The CRM will retrieve the customer’s email, phone number and a list of past orders from the customer database and sort the list of restaurants received from the IM based on the list of past orders. The CRM will then send a reply to the IM with the sorted list of restaurants, customer email and customer handphone number via a queue (q.reply.sortedlist).
8. The IM will then send the customer’s handphone number, email and the sorted restaurant list as a reply to the COS via a queue (q.reply.search). The COS will then store the customer’s email and handphone number in the current session and display the sorted list of restaurants and menus for the customer to pick from.
9. When the customer has made a selection, he will be asked to fill in his credit card details. These details are then sent from the COS through HTTPS to a PayPal RESTFUL web service to process the transaction
10. The COS then sends the order details, customer handphone number and email and the delivery location postal code to the IM with a fire-and-forget JMS message via a queue (q.receiveorder).
11. The IM sends the order details to the CRM with a fire-and-forget JMS message via a queue (q.sendtocrm).
12. The IM invokes the relevant restaurant’s order RESTFUL web service though HTTP with the order details, and receives the estimated time for preparation by the restaurant.
13. The IM invokes a Google RESTFUL web service via HTTP with the geolocation of the restaurant and the postal code specified by the customer to calculate the estimated time for delivery.
14. The IM then sums the preparation time and delivery time and invokes an SMS RESTFUL web service with the customer’s handphone number to send a sms with an acknowledgement of the order and the estimated total time.
15. The IM utilizes an email plugin (Tibco BusinessWorks) to send an email with an acknowledgement of the order and the estimated total time to the customer’s email.

**Technical Overview Diagram:**

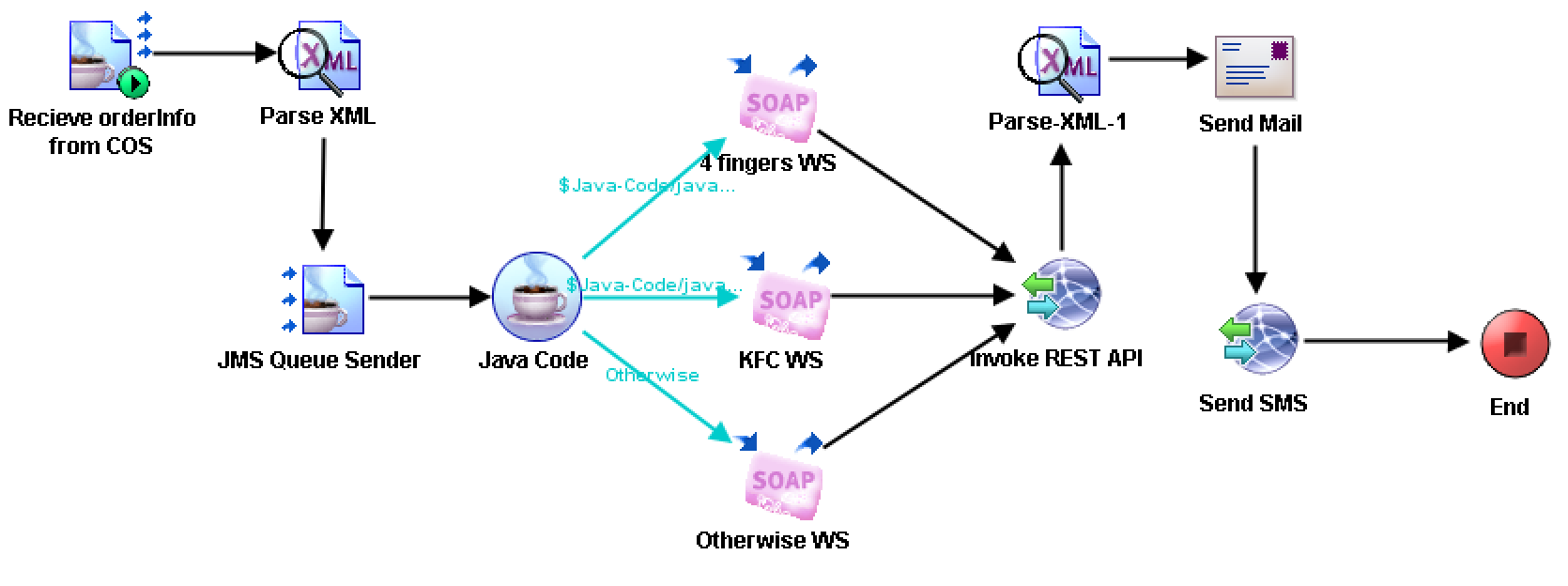


**Process Definition Diagrams:**

**Process 1**



**Process 2**



**JMS Implementation:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | From | To | Sync/Async | Fire and Forget (FF) /Request Reply (RR) | Pattern (Pub-sub / P2P) | JMS Queue/Topic (or Topic with Durable subscription) | Queue/Topic Name used |
| 1 | COS | IM | Sync | RR | P2P | Queue | q.request.search |
| IM | COS | Sync | RR | P2P | Queue | q.reply.search |
| 2 | IM | RMS | Sync | RR | P2P | Queue | q.request.region |
| RMS | IM | Sync | RR | P2P | Queue | q.reply.region |
| 3 | IM | CRM | Sync | RR | P2P | Queue | q.sendform |
| CRM | IM | Sync | RR | P2P | Queue | q.reply.sortedlist |
| 4 | COS | IM | - | FF | P2P | Queue | q.receiveorder |
| 5 | IM | CRM | - | FF | P2P | Queue | q.sendtocrm |