Nutrition Service - Composite Service Example

Elias Khalil, Ahmed Zarour and Ali Asfour

JMSE/ Birzeit University

April 29, 2017

1 Introduction

Nutrition Services are becoming more and more usable now a days, obesity diseases are conquering the new world. Many of the online resources introduce multiple solutions to help approaching more healthy life style by having the right diet , the right activity style and of course using scientific methodologies and medical advises for each patient by specialized doctors or diet experts, Yes nowadays obesity figures shows that obesity diseases are becoming the diseases of the Era.
lately many applications on the web or mobile are offering different services for diet information , healthy life style and type of food to eat based on the provided information form the user , a user can use multiple applications to get different information that he needs to combine together in order to get a proper advice for his life style.
In this Proposal we are trying to introduce a new service called NuriServe that can help developers for such applications to achieve the needs of their users by consuming one End point that can by design get multiple information from different resources and combine the result in a way the consumer can have most of is data in one response, a data that is collected in regards of input data and prepared in a way that is beneficial to the caller.

2 Architecture

Our composite service is Designed based on four main atomic services , some of the service are already online cloud services and some are developed specially for this solution, the developed services are not working only for this service , they are also deployed to be used by other services or consumed individually. The below Figure shows the High level architecture of our Composite service. Nutrition service is a REST based service running on cloud, following sections will describe inputs and outputs objects using JSON

2.1 NeturiServe Appengine Service Provider

This is the main NutriServe service provider. It provides a centric service provider aggregating four different atomic web services in order to provide a nutrition related services to uses in on place. This service is a RestFull service providing providing support for users to register, get services and store user related data on Google datastore service.

NutriServe is hosted on Google Appengine, providing a cloud based solution for this service. It gets benefit from Google cloud web services including compatibility with other Google services such as Datastore and authentication services. This service provides the following functionality:

1. Acting as centric web service support for NutriServe clients.
2. Handling Nutrition and health user management based on user data providing plans and recommendations
3. Providing data storage functionality by being integrated with Google Datastore service.
4. Communicating with 3rd party used by NutriServe to get extra functionality not provided in house.

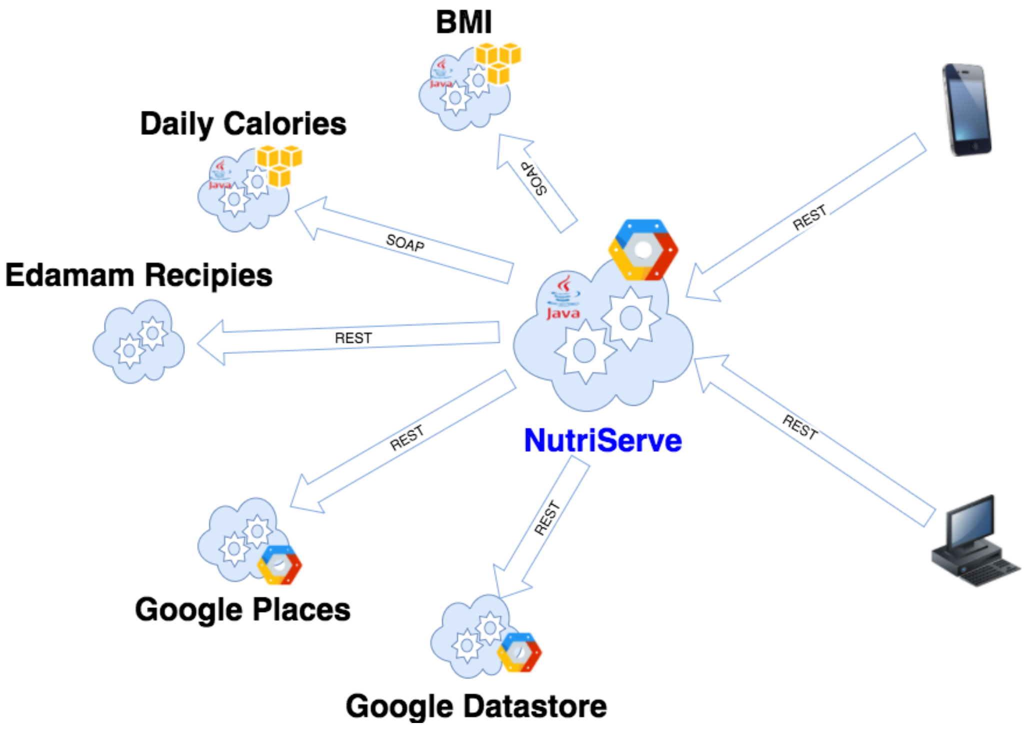


Figure 1: Composie Service Architecture.

2.2 NutriServe Storage

Users data is the the base of NutriServe services. Most functionality provided is Dependant on the user entries and updates. To store user related data in a reliable database, Google Datastore storage service is used. Datastore is a noSQL schemaless database. This datastore is used to save user data in entities as the following:

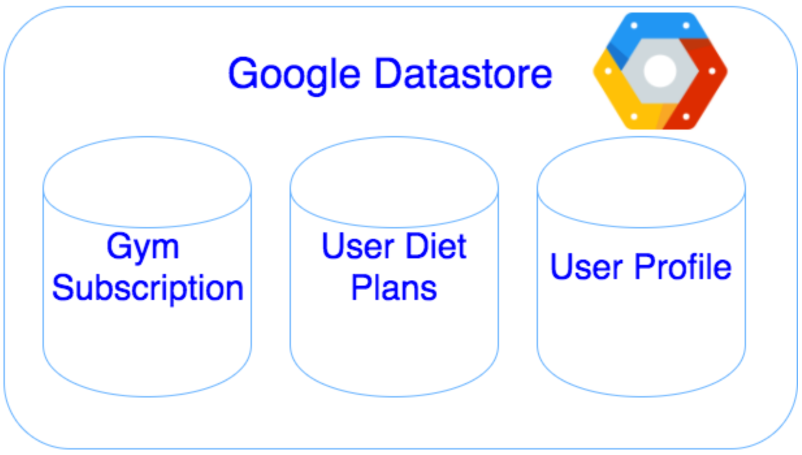


Figure 2: NutriServe Datastore Entities.

1. User profile: User profile data collected at registration or edited by user.
2. User plan: Data collected and estimated based on user nutrition input. Such information are used to manipulate user diet plans and system recommendations.
3. Gym subscription: NutriServe provide a networks of gym subscription where NutriServer registered users can use at any time.

Figure 2 lists the main entities used to save system and user data on the Datastore.

2.3 Gym Allocator

Gyms are part of any health or diet plan. NutriServe provides a service to allocate a nearby gym within the NutiServe subscription network. Users can use this service to allocate a nearby gym that they can join anytime, anywhere withing the NutriServe subscription model.

To allocate gyms in the network and indicate the closest gym to the user. Google places service is used to get gyms locations in the user near by area. User place location should be provided to the service to get the nearby gyms. The system filters the gyms based on the network subscription where users can join any network gym anytime in any place once there subscription is valid.

2.4 Body Mass Index (BMI)

BMI service is a Customer service Developed for this Composite application and works as a standalone SOAP API which calculated the Body Mass Indicator, the Body Mass Indicator (BMI) is a very important figure that is used in other calculations and it is also used to determine obesity indication.

BMI = (Weight in Kilograms / (Height in Meters x Height in Meters))

BMI Service Inputs are Weight in Kilograms and Height in Meters
BMI Service Output : rounded Integer Value

2.5 Calories Consumption Service

Calories Service is a standalone Soap web service that calculates how much calories per day a person needs, this service is also developed for the use of our composite application but can be used alone in its own context

Output of the calories service is a list of calories per day

2.6 Recipes Service

Based on user input data and calculated data such as age, BMI and daily calories, NutriServe suggests a daily food recipes for the user. This takes into account user food preferences while taking in consideration all other user related parameters.

Recipes service provided by NutriServe is based on a 3rd party service provided by Edamam. System handles all authentication or subscription required for this service.

3 API Documentation

The Nutrition Service accepts inputs in JSON format , it Includes below Operations

1. Registration
2. HealthStatus
3. InfoUpdate
4. TodayDietPlan

3.1 Registration

The Registration Operation Allows the User to register in the Nutrition Service,by Registering to the service a user gets a unique userId to be used in further operations.

Method : Post
Content Type : Json
Output format : Json

Required Parameters	
Username	The Username for the Registered User, Username Must Be Unique
Email	Email Address for the user
Weight	Weight in Kilo grams
Height	Height in Meters
Age	Age in Years
Gender	One of the following Values (**M** for Male or **F** for Female)
ActivityStyle	**Active** [Exercise more than 3 days a week ,not less than 1 hour per day] **NotActive** [No exercise] **SemiActive** [Exercise 3 days or less a week , one]

To use this Operation , call an Http Post method to the following URL

Http://[SERVER]:[PORT]/neutrisrvs/Register

with below Sample data ;

```
{
  "UserName" : "test123",
  "Age": "30",
  "Weight": "100",
  "Height": 1.75,
  "Gender": "M",
  "ActivityStyle" :"NotActive",
  "Email" :"test@test.com"
}
```

Below is an example of output object

```
{
"UserName" : "test123",
"userid"   : "de4361c0-2a49-4d0b-8a58-20f9dc30641e"
}
```

3.2 HealthStatus

This Operation provides the user with his Health Status base don the Provided Information in Registration phase , a user can query his health Status using the HTTP Method **GET**
Method : Get
Output format : Json

Required Parameters	
userid	Userid of registered user

To use this Operation , call an Http Post method to the following URL

Http://[SERVER]:[PORT]/neutrisrvs/HealthStatus?userid=de4361c0-2a49-4d0b-8a58-20f9dc30641e

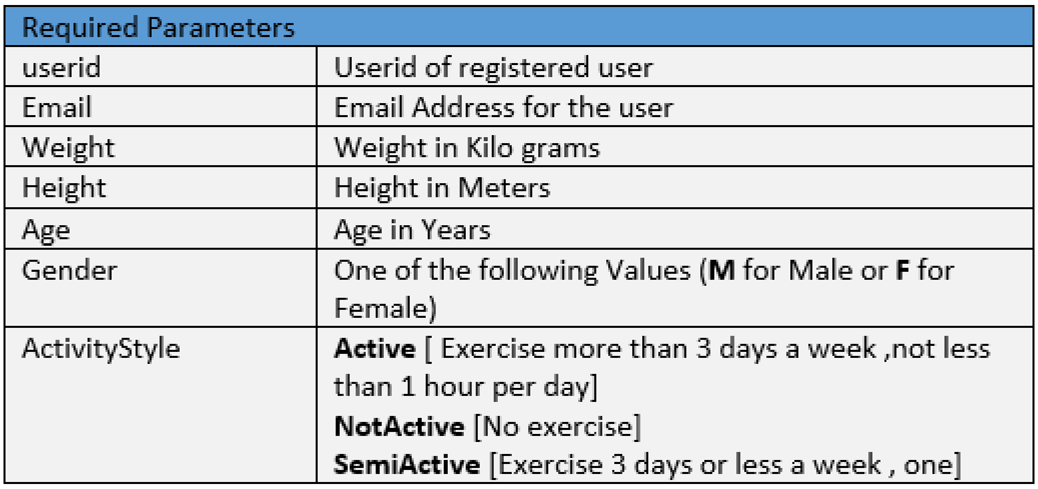
Below is an example of output object

```
{
"userid"    : "de4361c0-2a49-4d0b-8a58-20f9dc30641e"
"bmi" : "24.6",
"healthStatus" : "Healthy"
}
```

3.3 UpdateInfo

The Registration Operation Allows the User to Update his Registration in the Nutrition Service

Method : Put
Content Type : Json
Output format : Json



Required Parameters	
userid	Userid of registered user
Email	Email Address for the user
Weight	Weight in Kilo grams
Height	Height in Meters
Age	Age in Years
Gender	One of the following Values (**M** for Male or **F** for Female)
ActivityStyle	**Active** [Exercise more than 3 days a week ,not less than 1 hour per day] **NotActive** [No exercise] **SemiActive** [Exercise 3 days or less a week , one]

To use this Operation , call an Http Put method to the following URL

Http://[SERVER]:[PORT]/neutrisrvs/Register

with below Sample data ;

```
{
  "userid" : "de4361c0-2a49-4d0b-8a58-20f9dc30641e",
  "Age": "30",
  "Weight": "100",
  "Height": 1.75,
  "Gender": "M",
  "ActivityStyle" :"NotActive",
  "Email" :"test@test.com"
}
```

Updating Information returns to the user with updated health status base don updated entries,Below is an example of output object

```
{
"userid"    : "de4361c0-2a49-4d0b-8a58-20f9dc30641e"
"bmi" : "25.0",
"healthStatus" : "Healthy"
}
```

3.4 TodayDietPlan

This Operation provides the user with Todays Diet plan , base don entered data in the registration and input data to the service , an object with a diet plan including recipes and calories to burn in this day is returned along with bets offers for Gyms to exercise near the user location

Method : Get
Output format : Json

Required Parameters	
Userid	Userid of registered user
Location	Coordinates of the user current location
FoodType	A string value like chicken, carrots, apples...etc
Calories	amount of calories required

Output Parameters	
Userid	Userid of registered user
Recipes	List of Recipes a user can have during his day , each recipe contains a list of parameters including calories, ingredients and a ULR for how to prepare
Gym Offers	List of offers for Gyms near your place

To use this Operation , call an Http Post method to the following URL

Http://[SERVER]:[PORT]/neutrisrvs/Plan?userid=de4361c0-2a49-4d0b-8a58-20f9dc30641e&location
0.126446&foodtype=chicken&calories=400

Below is an example of output object

```
{
        "Reciepies": [
                {
                        "Id": 1,
                        "Name": "Meal 1",
                        "Description": null,
                        "Cuisine": "Western",
                        "Image": "http://somimage.com",
                        "Calories": 200
                },
                {
                        "Id": 2,
                        "Name": "Meal 2",
                        "Description": null,
                        "Cuisine": "eastern",
                        "Image": "http://somimage2.com",
                        "Calories": 300
                }
        ],
        "Gyms": [
                {
                        "Id": "1",
                        "Name": " Gym1",
                        "Address": " Street 1",
                        "OfferDescription": " We have a good offer"
                },
                {
                        "Id": "2",
                        "Name": " Gym2",
                        "Address": " Street 2",
                        "OfferDescription": " We have a better offer"
                }
        ],
        "Calories": [
                {
                        "DayNumber": 1,
                        "Calories": 1800
                },
                {
                        "DayNumber": 2,
                        "Calories": 2000
                },
                {
                        "DayNumber": 3,
                        "Calories": 2300
                }
        ]
}
```

4 Atomic Services

In This Section we describe each atomic service and its role in our composite Application , each service have its own context and not necessarily works with same protocol of other service

4.1 BMI Service

Body mass index web service is a stand alone web service provided by NutriServe. This service provides a SOAP based service to calculate the BMI based on user given input for weight an length. Figure 3

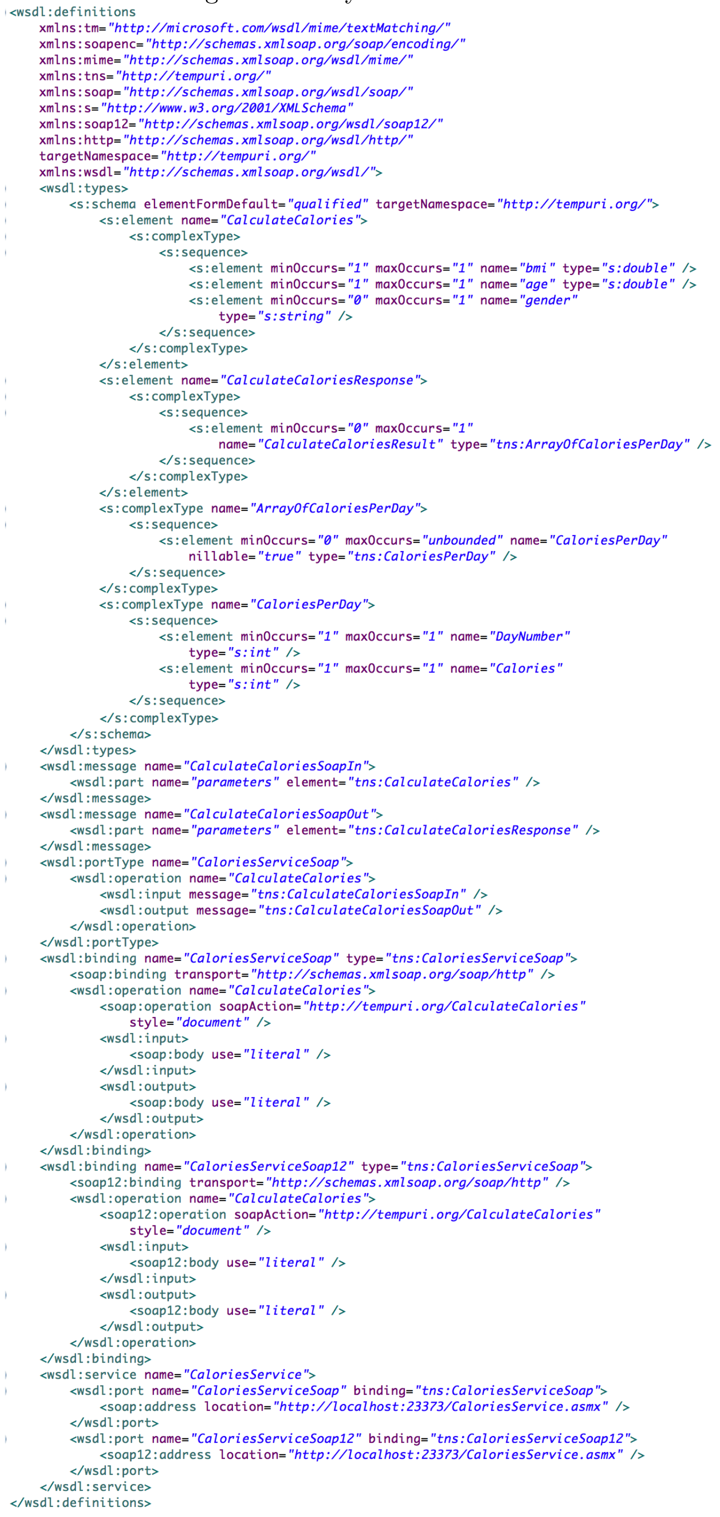
Figure 3: Body Mass Index Service WSDL

```xml
<definitions name="BMIService"
    targetNamespace="http://bzu.cloud:8080/wsdl/HelloService.wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://bzu.cloud:8080/wsdl/BMIService.wsdl"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <message name="getBMIRequest">
        <part name="weight" type="xsd:float" />
        <part name="height" type="xsd:float" />
    </message>

    <message name="getBMIResponse">
        <part name="bmi" type="xsd:float" />
    </message>

    <portType name="BMI_PortType">
        <operation name="calculateBMI">
            <input message="tns:getBMIRequest" />
            <output message="tns:getBMIResponse" />
        </operation>
    </portType>

    <binding name="BMI_Binding" type="tns:BMI_PortType">
        <soap:binding style="rpc"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <operation name="calculateBMI">
            <soap:operation soapAction="calculateBMI" />
            <input>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="urn:examples:BMIservice" use="encoded" />
            </input>
            <output>
                <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="urn:examples:BMIservice" use="encoded" />
            </output>
        </operation>
    </binding>

    <service name="BMI_Service">
        <documentation>WSDL File for BMIService</documentation>
        <port binding="tns:BMI_Binding" name="BMI_Port">
            <soap:address location="http://localhost:3030/bmicalculator/" />
        </port>
    </service>
</definitions>
```

4.2 Calories Consumption Service

A stand alone Soap web service provided by NutriServe. This service provides a daily calories consumption.

Figure 4: Daily Calories Calculator



```xml
<wsdl:definitions
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:tns="http://tempuri.org/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    targetNamespace="http://tempuri.org/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
            <s:element name="CalculateCalories">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="1" maxOccurs="1" name="bmi" type="s:double" />
                        <s:element minOccurs="1" maxOccurs="1" name="age" type="s:double" />
                        <s:element minOccurs="0" maxOccurs="1" name="gender"
                            type="s:string" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="CalculateCaloriesResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element minOccurs="0" maxOccurs="1"
                            name="CalculateCaloriesResult" type="tns:ArrayOfCaloriesPerDay" />
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:complexType name="ArrayOfCaloriesPerDay">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="unbounded" name="CaloriesPerDay"
                        nillable="true" type="tns:CaloriesPerDay" />
                </s:sequence>
            </s:complexType>
            <s:complexType name="CaloriesPerDay">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="DayNumber"
                        type="s:int" />
                    <s:element minOccurs="1" maxOccurs="1" name="Calories"
                        type="s:int" />
                </s:sequence>
            </s:complexType>
        </s:schema>
    </wsdl:types>
    <wsdl:message name="CalculateCaloriesSoapIn">
        <wsdl:part name="parameters" element="tns:CalculateCalories" />
    </wsdl:message>
    <wsdl:message name="CalculateCaloriesSoapOut">
        <wsdl:part name="parameters" element="tns:CalculateCaloriesResponse" />
    </wsdl:message>
    <wsdl:portType name="CaloriesServiceSoap">
        <wsdl:operation name="CalculateCalories">
            <wsdl:input message="tns:CalculateCaloriesSoapIn" />
            <wsdl:output message="tns:CalculateCaloriesSoapOut" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="CaloriesServiceSoap" type="tns:CaloriesServiceSoap">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="CalculateCalories">
            <soap:operation soapAction="http://tempuri.org/CalculateCalories"
                style="document" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="CaloriesServiceSoap12" type="tns:CaloriesServiceSoap">
        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="CalculateCalories">
            <soap12:operation soapAction="http://tempuri.org/CalculateCalories"
                style="document" />
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CaloriesService">
        <wsdl:port name="CaloriesServiceSoap" binding="tns:CaloriesServiceSoap">
            <soap:address location="http://localhost:23373/CaloriesService.asmx" />
        </wsdl:port>
        <wsdl:port name="CaloriesServiceSoap12" binding="tns:CaloriesServiceSoap12">
            <soap12:address location="http://localhost:23373/CaloriesService.asmx" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

4.3 Edamam Recipies Service

The Edamam Recipe Search is a standalone REST service used to build custom diet plans. The developer with this API will be able to search for recipes from over 1.5 million recipes web sources and organized databases. In addition, it provides filtering capabilities to search for recipes by calories, type of foods and diet. The following are inputs for this API:

1. app_id (our service ID) app_key (our service key) for authentication access for Edamam API

2. Query text to determine the type of food (chicken, vegetables, etc.)
3. The desired calories range per serving.

Edamam Recipe search service output: List of recipes contains information about each recipe such as recipe name, calories, image, URL (Useful for web applications to open the original recipe page).
API Endpoint: https://api.edamam.com/search

4.4 Google Places Service

The Google Places is a REST Web Service allows searching for place information base on a variety categories, such as: distance, type, geographic locations, and more. This service will be used to query for nearby gyms. The following are inputs for this API:

1. Our application's API key for purposes of quota management.
2. Location by specify latitude and longitude.
3. Radius in meter to specify the distance between the location and the list of returned places.
4. Type of places (always gym)

Output of Google Places: List of gym with summary information about each gym based on a user's location.
API Endpoint: https://maps.googleapis.com/maps/api/place/nearbysearch

5 Deployment

Nutriserve is built as a composition of in house services and 3rd party services. All services are hosted on the cloud providing high level of availability and inter services communication.

5.1 Google AppEngine

The Main NutriServe service application server is build on Google cloud using Appengine. This choice is made based on the scalability requirement of the application. Appengine provides a free quota which will cut deployment cost at the initial stage where user traffic is low. Appengine can scale up our application to support high traffic and load.

5.2 Google Datastore

Due to the composition nature of the data provided to users, need of relational database decreases. System saves the basic user information, settings and plans. Datastore provides a non schema based storage where data is stored in entities (table like) providing multiple properties. No hard relation is done between entities. This approach by Google boosts the performance of of data access.

5.3 Google places

A service used to retrieve location info about gyms part of the NetriServe subscription network. User can search for nearest gyms in the network around her. Integration with Google maps would be helpful for routing and other map services.

5.4 Amazon EC2

Both BMI and Daily calories are built as stand alone services. Any user can access these two services as the will be hosted on a tomcat 8 server on an AWS EC2 instance. Micro instances are selected for this purposes due to the simple nature of the services. These services have low CPU consumption and don't need any data access.