



Service-Oriented Architecture (SWEN6307)

Assignment #: 5

Project Name: Pal-TripAdvisor

Supervisor: Dr-Nariman Ammar

By: Hanan Namrouti  
Qutiba Mustafa  
Zeina Daghlis

Date : 28/May/2018

## 1. Introduction

### 1.1. Proposed Web-Service Application

## 2. The objectives

## 3. Description

## 4. Data Flow

## 5. The structure of the web Composition service

## 6. Technologies and Tools

## 7. Implementation & Discussion

## 9. Testing for two developed Service

## 10. Future Enhancements

## 11. Conclusion

# *Abstract*

Web Services are the means by which devices communicate over the World Wide Web. Whether you use a mobile application, search engine or an enterprise system, the user piece of the application (the interface) resides on the used device. The data, and potentially the business rules, live on some other server on the network. How the interface communicates with the server piece is the role of Web Services. Here in this course project we will demonstrate the concepts of web services via implementing REST and SOAP web-service. The project is considered as a web application for tourism guidance. Our application name is Pal-TripAdvisor.

**KeyWord:** *Web service, API, SOAP, REST*

## 1. Introduction

These days people spending a lot of time before deciding to travel to a new country. They usually know where they want to go, but sometimes they don't. Our web service designed for people who are feeling hesitancy in searching and collecting information about the country they are going to visit..

Usually, traveler visit one website for check the flight, then they try to find information about hotel in different website looking for famous sights to decide where they should go first, after that they collect information about weather in this country at specific time. And the most important thing is their need to know the currency of this country and the change rate comparing with their local currency.

Our web-service application will provide the user a comprehensive global view of the complex modern travel ecosystem and unlocks fresh insights into how travelers research and decide their holidays. It introduces a needs based approach to know hotels, images, currency rate and city weather. Moreover, the objective of this application that we want to learn composition webservice ,we find this is a good way to understand all fundamental and concept of our course. For sure beside implementation for SOAP and REST APIs and composite the implemented APIS with External APIs

### 1.1. Proposed Web-Service Application

In our proposed application , we will use REST and SOAP APIs and a composite them to have the needed services. We will have the user ( client) will request the data from web-server using web-application ( as we implemented ) or mobile application ( future work).

The user should enter some inputs to use the application , firstly the application will ask the user to provide the destination country name , local currency , date of departure and date of arrival.

Our web service will be composite of (Rest and WSDL ) Services , client request data from web server using either web page or mobile application then provide basic information as ( destination country , from date , to date ) . all API work to return point of interest , weather currency change and available hotel near the destination city. as shown below :

We sequentially call the first api (point of interests), and when this api returns a list of interesting places, we iterate over those returned places, and for each one, we parallelly call the other for api's (get weather, get currency, get closest hotels, get language) for each place, then we combine the response from those 4 api's with the data returned from the (point of interest api) to form complete information for each place, both sequential and parallel calls combined together to form the flow pattern.

## 2. The objectives

We can summarize our objectives as the following:

1. Designing a composition web service that will be an added value to other composition web services.
2. Reusing of existing web services such as Weather and CurrencyFactor API, then we deployed our own APIs.
3. Composite a publishable web services and orchestrate different web service (REST and SOAP ).

## 3. Description

After the user enter the needed parameter , we will use sequential call request for the API , the API will return a list of interesting places in this country, this developed API will return city name in this destination country , city image and list of the available hotels in this city.

Then we will parallelly call the rest of APIs using different invocation and call types. We will use the city name as an input parameter for the weather API to get the weather status in the need city. But the weather API needs three parameter as input, one parameter from the developed service ( PointOfInterest) and the other two parameters will be collected from the user.

City name will be used as input parameter for the hotel API , which return a list of hotels of the destination city, the city name is an output from point of interest API. The hotel API and weather API will be called in parallel sequence.

And then the input of local currency will be used in calculating the exchange rate between the local currency and destination country currency . This service is implemented and developed by us as SOAP service.

So after this process , the data from hotel API, weather API, currency API will be combined with the point of interest API result to provide the response for the user.

The below summary table will represent the used service.

API	Type	Source
Weather API	SOAP	Internal
Hotel API	REST	Internal
Currency exchange	SOAP	Internal
Point of interest	REST	Internal

API links:

SOAP: Weather API: <http://qutaibamustafa.eastus.cloudapp.azure.com/GetWeather.asmx?wsdl>

REST: Hotel API: <http://qutaibamustafa.eastus.cloudapp.azure.com/GetClosestHotels.svc/json/getHotelsByCity/Istanbul>

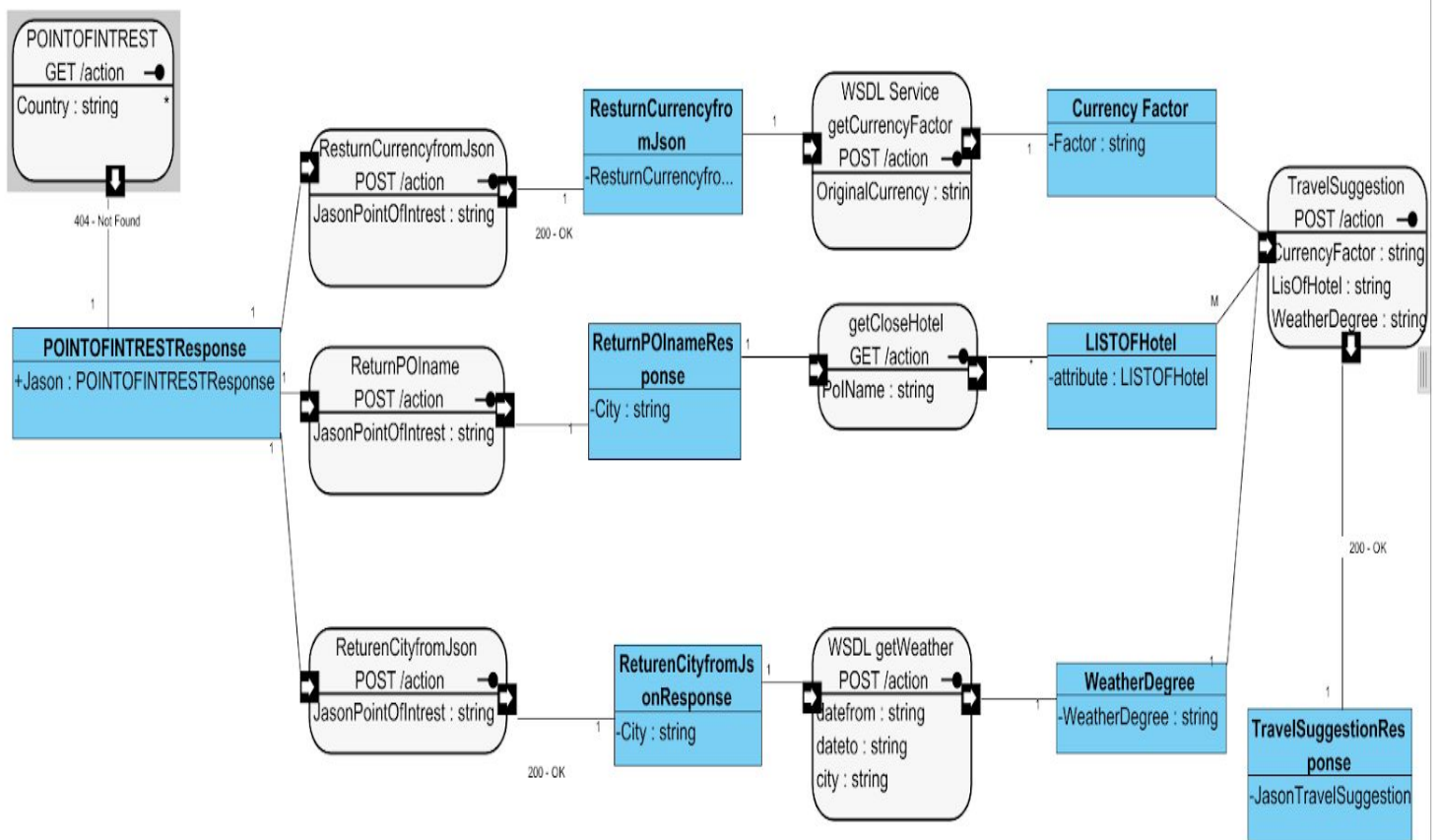
SOP: Currency Factor API: <http://qutaibamustafa.eastus.cloudapp.azure.com/CurrencyExchange.asmx>

REST: POI AOI: <http://qutaibamustafa.eastus.cloudapp.azure.com/PointsOfInterest.svc/json/GetPOIByCity/Turkey/Istanbul>

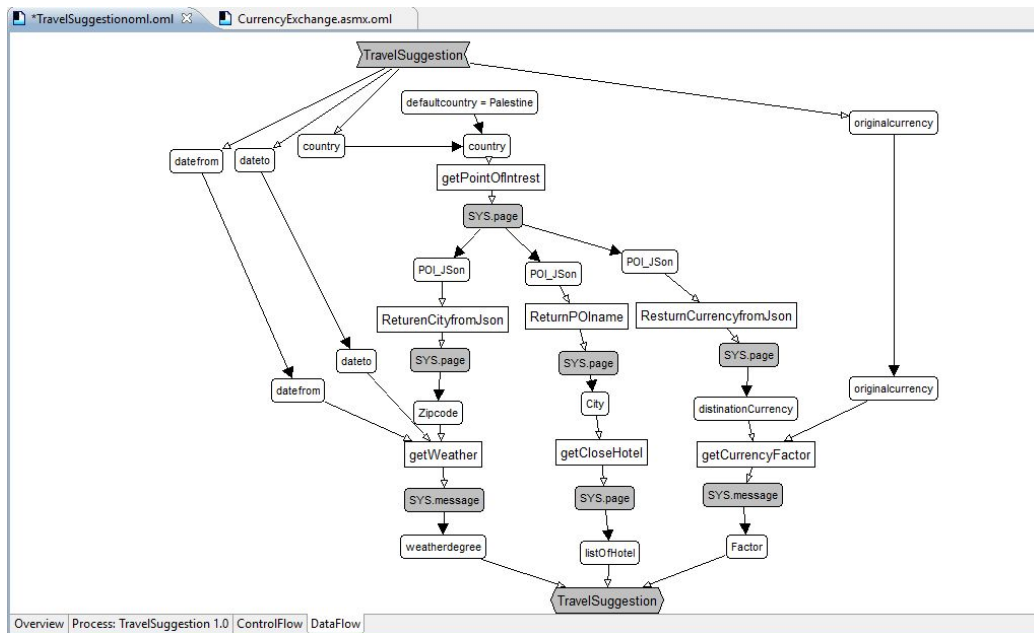
We used another 3 POST APIs to convert from Json to specific input for each API.

## 4. Data Flow

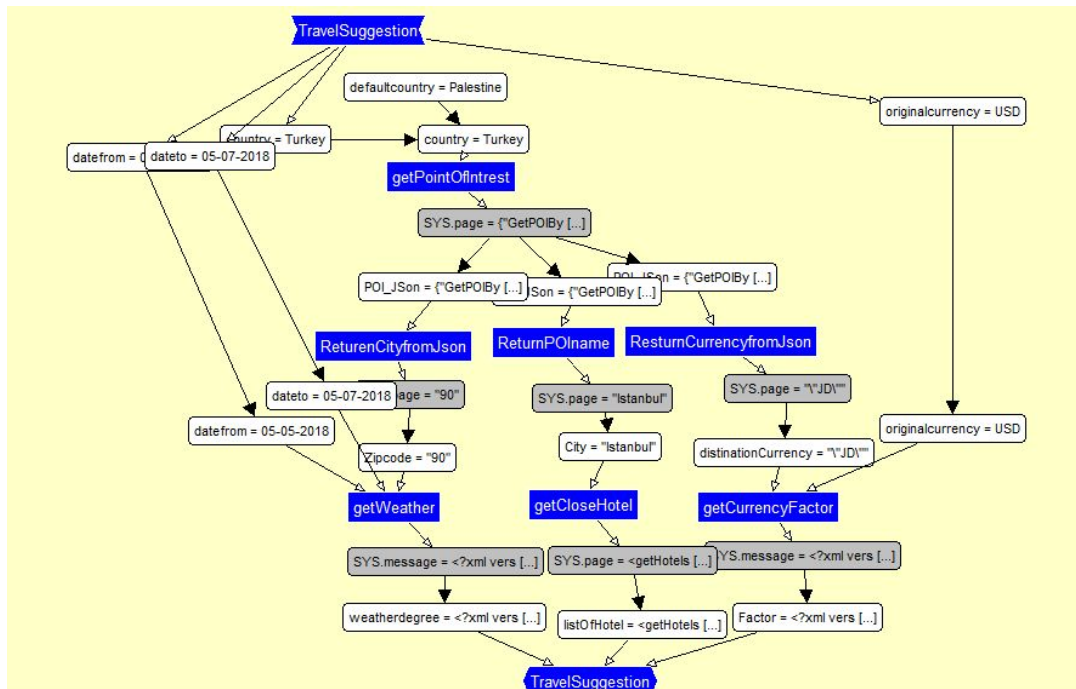
We used JOpera library on eclipse to represent the business process of our application and to test the APIs we invoked and developed . The below diagrams are shown below :



## Data flow before testing:

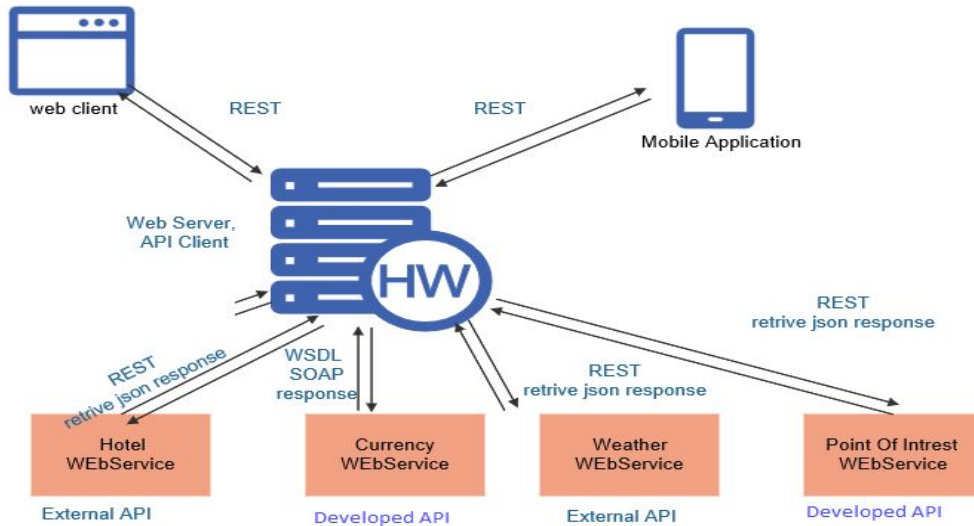


## Data flow after testing:

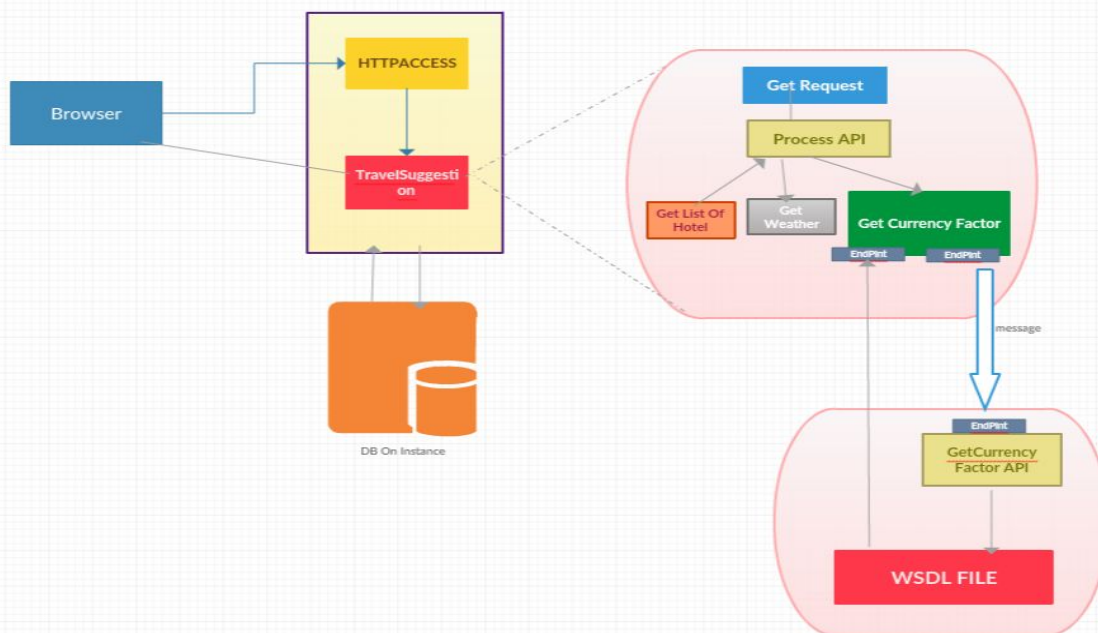


## 5. The structure of the web Composition service

This is our initial draft HW architecture, then we had to modify it and add another 3 APIs.

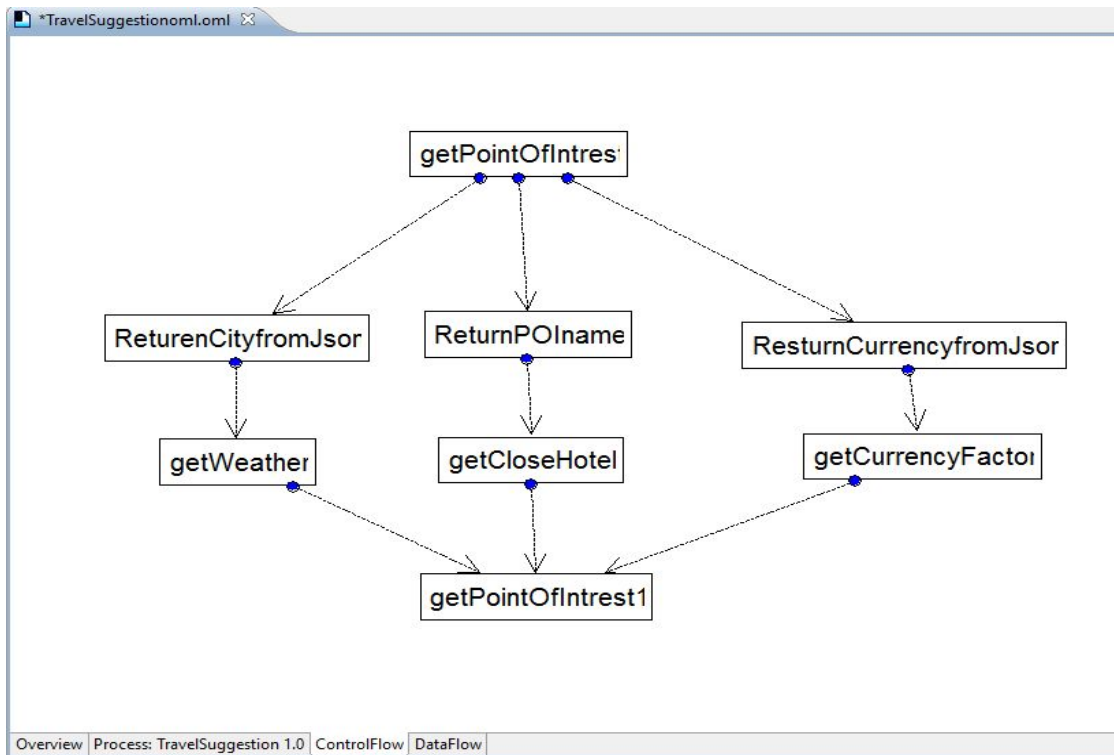


Then our system structure modified to as shown th diagram , we change the type of weather currency to SOAP and we develop it. Close to composition in Currency WebService.

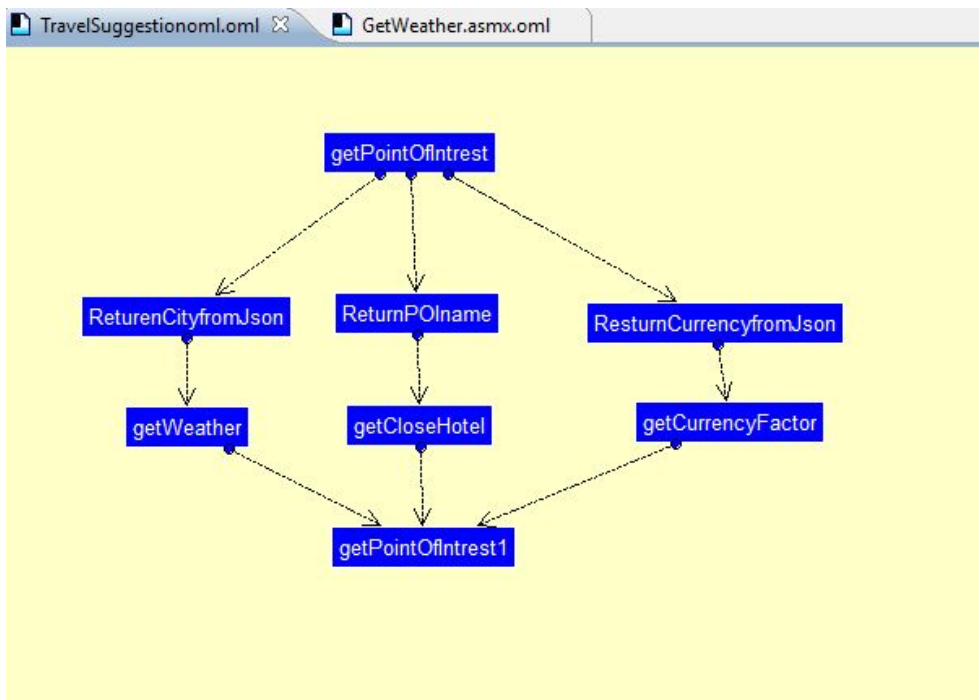




Control flow in JOpera before testing :



Control flow after testing:



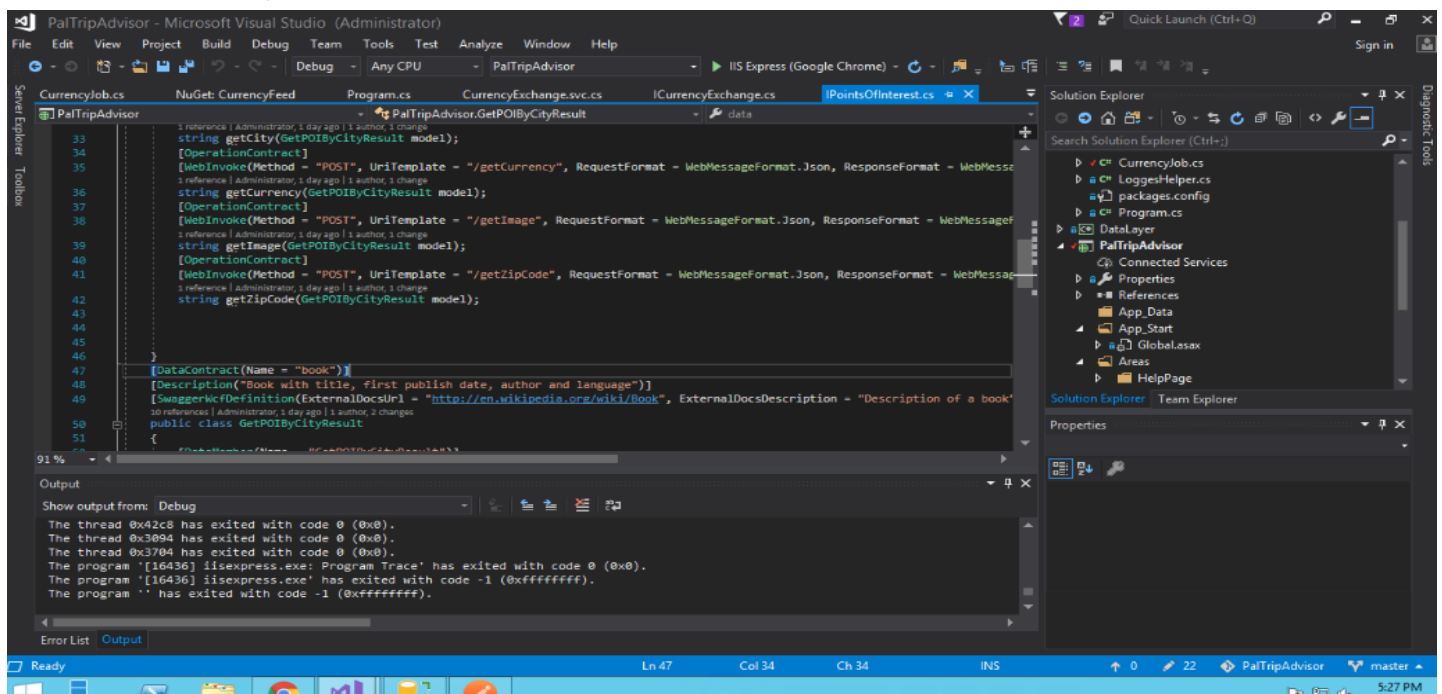
## 6. Technologies and Tools

In this project we will use the following technologies and tools :

1. **Swagger**: RESTful API Documentation Specification. is a project used to describe and document RESTful APIs. The Swagger specification defines a set of files required to describe such an API. These files can then be used by the Swagger-UI project to display the API and Swagger Codegen to generate clients in various languages. Additional utilities can also take advantage of the resulting files, such as testing tools.

**Note**: Swagger is build for only REST service, it's functionality depends on auto-generating the end-points from the controller and get the input and output from the description (comment) above the endpoints. In our case, we use WCF which doesn't contains controller, and swagger doesn't support it. We tries couple of open-source tools created by individuals and uploaded to github (such as: Swagger4WCF, SwaggerWCF, WCFSwagger), but those tools are not professional and contains lots of bug, none of those tools works with WCF, never.

2. **Visual Studio** :is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as web sites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. As the below screenshot



3. **GitHub repository**: we used github repository to push all new changes and to document out project .

The link for our repository :

<https://github.com/QMustafa/PalTripAdvisor>

← → ↻ <https://github.com/QMustafa/PalTripAdvisor> 🔍 ☆ 🏠 G

QMustafa / PalTripAdvisor Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights

PalTripAdvisor

20 commits 1 branch 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Administrator and Administrator fix json format Latest commit 3e1c17 a day ago

PalTripAdvisor	fix json format	a day ago
.gitattributes	first commit	15 days ago
.gitignore	Initial commit	15 days ago
README.md	Update README.md	15 days ago

README.md

## PalTripAdvisor

PalTripAdvisor

PalTripAdvisor is a web-service will be used as project for web service oriented course. The main purpose of this project is to prepare a platform for travelers in which they can get information to help them in deciding where to go, weather in the destination country, currency and language used in this country. Our service will suggest hotels also in order to offer a

#### 4. Microsoft windows Server :

We reserved a microsoft server from amazon to use it in developing the needed APIs and the database for our application.

IP: 54.71.170.54

Also we create three users for team members .

Remote Desktop Connection

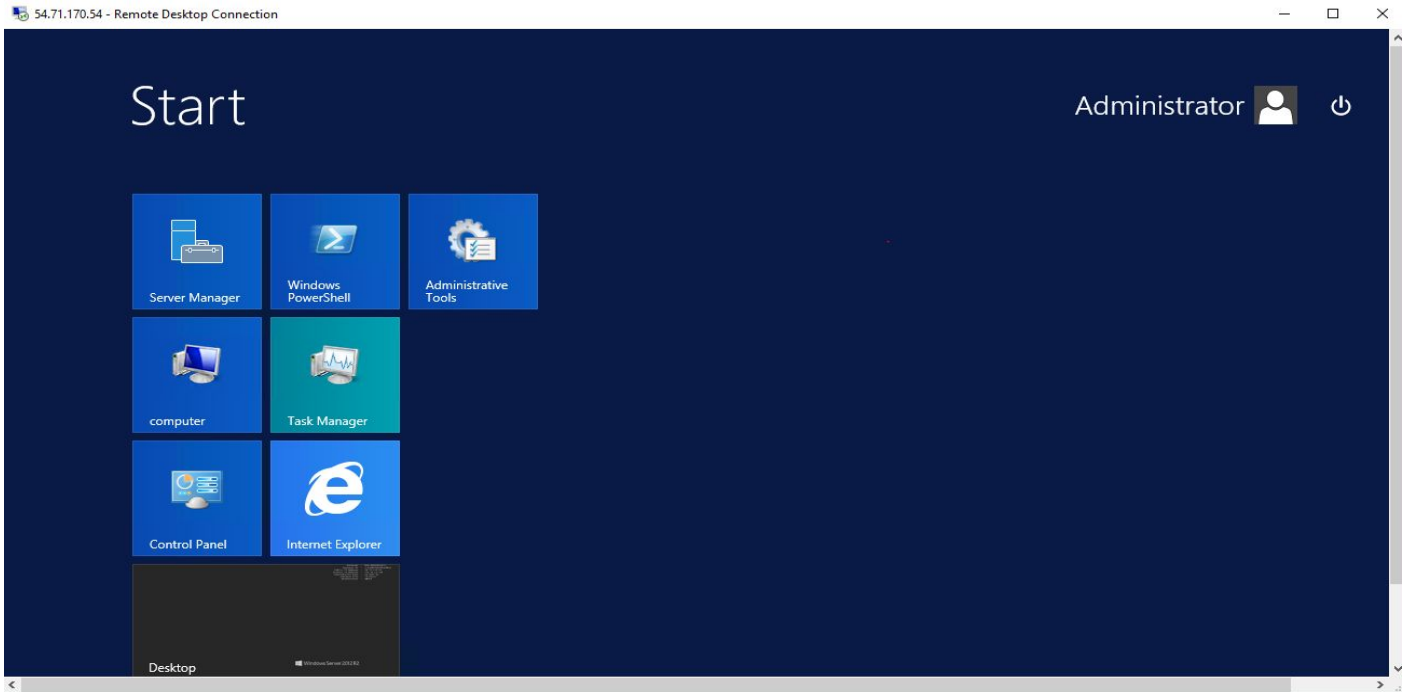
Remote Desktop Connection

Computer: 54.71.170.54

User name: WIN-48TK3BTCA7U\Administrator

You will be asked for credentials when you connect.

Show Options Connect Help

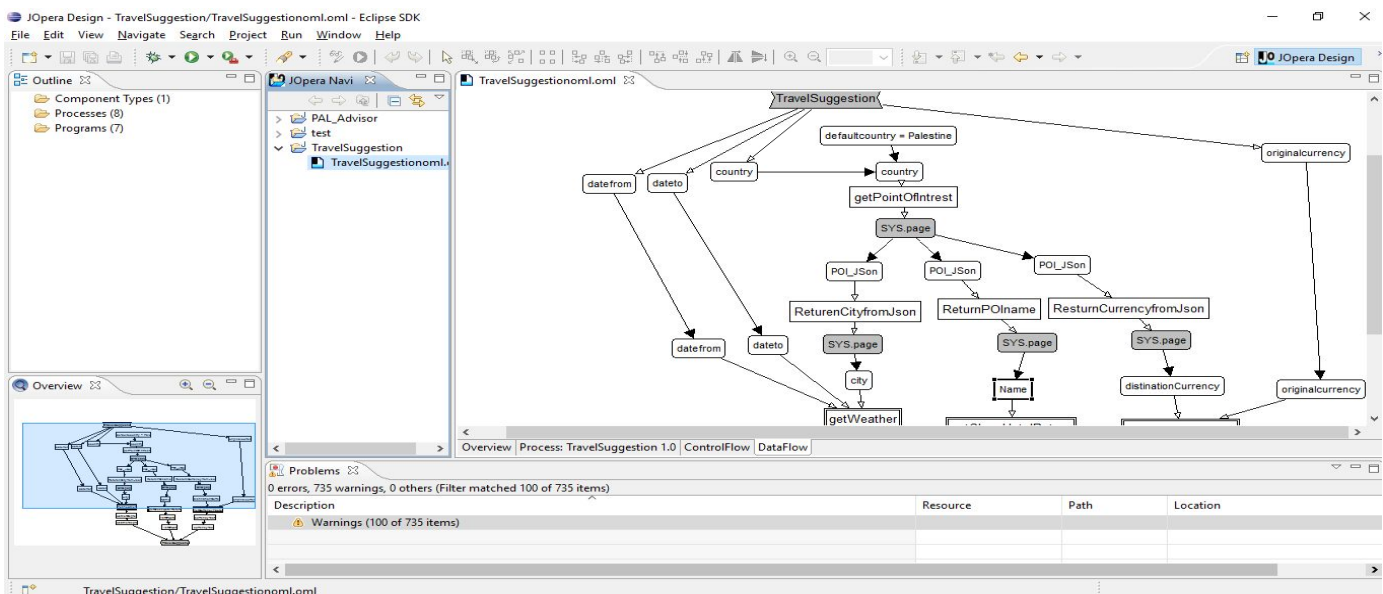


This server was closed by amazon twice , so reserved another server from Microsoft using Azure:

Server IP:40.114.70.230

Soin this report we will demonstrate data from both servers, since we was preparing the report while doing development, cimosition , integration with JOpera and testing.

**4. JOpera :** JOpera for Eclipse is a rapid service composition tool offering a visual language and autonomic execution platform for building distributed applications out of reusable services, which include but are not strictly limited to Web services. We used JOpera in testing our service one by one and for testing the composition as the below screenshot



## 5. Microsoft SQL server Management studio:

Since we will provide a trip guidance system we had to build a database and we fill it with real data and dummy data, we create many tables, for currency, currency exchange rate, hotel and point of interest table. Since we will call many APIs to return values from this database. As show in the below screenshot

The screenshot displays the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'localhost\SQLEXPRESS (SQL Server 14.0.11)'. The 'Currencies' table is selected under the 'dbo' schema. The SQL Query window shows the following query:

```
SELECT TOP (1000) [Id]
, [Name]
, [Slug]
FROM [PalTripAdvisorServices].[dbo].[Currencies]
```

The Results pane shows the following data:

Id	Name	Slug
1	Jordanian Dinar	JD
2	Turkish Lira	TL
3	British Pound	GBP
4	United State Dollar	USD
5	Shekel	NIS
6	Ruby	IR

The Properties pane on the right shows the current connection parameters for 'localhost\SQLEXPRESS'.

Aggregate Status

Property	Value
Connection failure	
Elapsed time	00:00:00.149
Finish time	5/30/2018 3:42:28 PM
Name	localhost\SQLEXPRESS
Rows returned	6
Start time	5/30/2018 3:42:28 PM
State	Open

Connection

Property	Value
Connection name	localhost\SQLEXPRESS

Connection Details

Property	Value
Connection elapsed	00:00:00.149
Connection encrypt	Not encrypted
Connection finish t	5/30/2018 3:42:28 PM
Connection rows r	6
Connection start ti	5/30/2018 3:42:28 PM
Connection state	Open
Display name	localhost\SQLEXPRESS
Login name	WIN-48TK3BTCA7U\Ad
Server name	localhost\SQLEXPRESS
Server version	14.0.1000
Session Tracing ID	
SPID	56


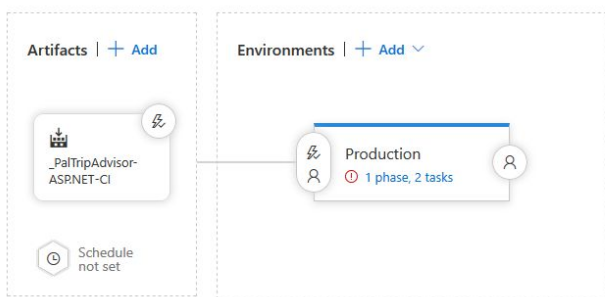
Name

The name of the connection.

**6. VSTS (Visual studio team server):** we used VSTS to support continuous Integration & continuous deployment, we linked it with our github repository, and our server, and identified the build definitions which automatically build the solution after each push to the github repository, publishing the artifact of the the build, then deploying the artifact to our server which runs the IIS.

[illegible]

Pipeline Tasks Variables Retention Options History



The screenshot shows the IntelliJ IDEA interface with the 'PalladioAdvisor.ASPIN-CI' build configuration. The 'Properties' tab is active, displaying the following fields:

- Name:** PalladioAdvisor.ASPIN-CI
- Agent queue:** Hosted VCS21V
- Parameters:** (empty)
- Path to solution or packages config:** (empty)
- Artifact name:** (empty)
- Output:** chp

## 7. Implementation & Discussion

In order to test the services, we had to fill a Demo data in the DB tables, then an idea came to create a client project (windows service) that runs 24 hours on the server (background job - windows service).

The main objective of this job is to keep requesting an external web api (called CurrencyLayer), and gets the real factors between the currencies added to our db, and save the factors again into the db.

To achieve this, we built a .NET console application, added the logic:

1. Get the existing currencies in db.
2. Requesting the external api (Currency layer) by sending one of the currencies as the original currency and the rest of them as the destination currencies.
3. Getting the response from the api (the response is list of factors between the original currency and each one of the destination currencies).
4. Save the response to the db (Feed the currency table with a real exchange factors).

Note: we used a nuget package called "TopShelf", to convert the console application to a windows service which runs in the background.

Note: we used another nuget package called "Quartz", which is a job scheduler package, using this package, we scheduled our project (the windows service) to run every 2 hours and update the factors again to keep the data real and up-to-date (the previous logic will be triggered every 2 hours).

Note: we added logs to keep monitoring this windows service and facilitate debugging and error fixing .



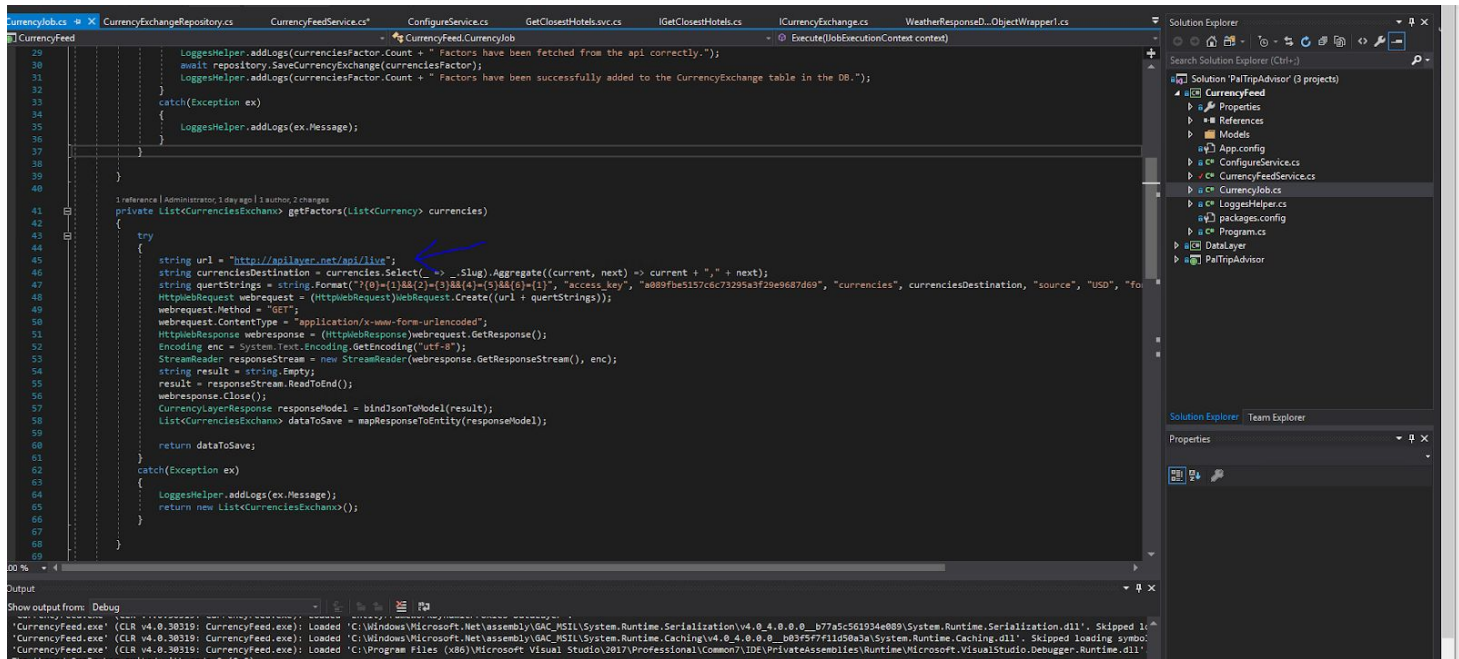
Services (Local)					
PalTripAdvisorJob	Name	Description	Status	Startup Type	Log On As
Start the service	Netlogon	Maintains a secure channel between this computer and the do...	Manual	Local System...	
	Network Connection Broker	Brokers connections that allow Windows Store Apps to receive n...	Running	Manual (Trig...	Local System...
Description: Sync Currencies to PalTripAdvisor database	Network Connections	Manages objects in the Network and Dial-Up Connections folde...	Manual	Manual (Trig...	Local System...
	Network Connectivity Assis...	Provides DirectAccess status notification for UI components	Running	Manual (Trig...	Local System...
	Network List Service	Identifies the networks to which the computer has connected, c...	Running	Manual	Local Service
	Network Location Awareness	Collects and stores configuration information for the network a...	Running	Automatic	Network S...
	Network Setup Service	The Network Setup Service manages the installation of network ...	Running	Manual (Trig...	Local System...
	Network Store Interface Ser...	This service delivers network notifications (e.g. interface additio...	Running	Automatic	Local Service
	Offline Files	The Offline Files service performs maintenance activities on the ...	Disabled	Local System...	
	Optimize drives	Helps the computer run more efficiently by optimizing files on s...	Manual	Local System...	
	PalTripAdvisorJob	Sync Currencies to PalTripAdvisor database	Running	Automatic	Local System...
	Performance Counter DLL ...	Enables remote users and 64-bit processes to query performanc...	Manual	Local Service	
	Performance Logs & Alerts	Performance Logs and Alerts Collects performance data from lo...	Running	Manual	Local Service
	Phone Service	Manages the telephony state on the device	Manual	Manual (Trig...	Local Service
	Plug and Play	Enables a computer to recognize and adapt to hardware change...	Running	Manual	Local System...
	Portable Device Enumerato...	Enforces group policy for removable mass-storage devices. Ena...	Running	Manual (Trig...	Local System...
	Power	Manages power policy and power policy notification delivery.	Running	Automatic	Local System...
	Print Spooler	This service spools print jobs and handles interaction with the pr...	Running	Automatic	Local System...
	Printer Extensions and Notif...	This service opens custom printer dialog boxes and handles noti...	Manual	Local System...	
	Problem Reports and Soluti...	This service provides support for viewing, sending and deletion ...	Manual	Local System...	
	Program Compatibility Assi...	This service provides support for the Program Compatibility Ass...	Running	Automatic	Local System...
	Quality Windows Audio Vid...	Quality Windows Audio Video Experience (qWave) is a networki...	Manual	Local Service	
	Radio Management Service	Radio Management and Airplane Mode Service	Manual	Local Service	
	RdAgent		Running	Automatic	Local System...
	Remote Access Auto Conne...	Creates a connection to a remote network whenever a program ...	Manual	Local System...	
	Remote Access Connection...	Manages dial-up and virtual private network (VPN) connections ...	Manual	Local System...	
	Remote Desktop Configur...	Remote Desktop Configuration Service (RDCS) is responsible for ...	Running	Manual	Local System...
	Remote Desktop Services	Allows users to connect interactively to a remote computer. Re...	Running	Manual	Network S...
	Remote Desktop Services U...	Allows the redirection of Printers/Drives/Ports for RDP connecti...	Running	Manual	Local System...
	Remote Procedure Call (RPC)	The RPCSS service is the Service Control Manager for COM and ...	Running	Automatic	Network S...
	Remote Procedure Call (RP...	In Windows 2003 and earlier versions of Windows, the Remote P...	Manual	Network S...	
	Remote Registry	Enables remote users to modify registry settings on this comput...	Running	Automatic (T...	Local Service
	Resultant Set of Policy Prov...	Provides a network service that processes requests to simulate a...	Manual	Local System...	
	Routing and Remote Access	Offers routing services to businesses in local area and wide area ...	Disabled	Local System...	
	RPC Endpoint Mapper	Resolves RPC interfaces identifiers to transport endpoints. If this ...	Running	Automatic	Network S...
	Secondary Logon	Enables starting processes under alternate credentials. If this ser...	Running	Manual	Local System...
	Secure Socket Tunneling Pr...	Provides support for the Secure Socket Tunneling Protocol (SST...	Manual	Local Service	
	Security Accounts Manager	The startup of this service signals other services that the Security...	Running	Automatic	Local System...
	Sensor Data Service	Delivers data from a variety of sensors	Manual	Local System...	
	Sensor Monitoring Service	Monitors various sensors in order to expose data and adapt to sy...	Manual	Manual (Trig...	Local Service

The screenshot displays the Visual Studio IDE with the `CurrencyFeed.cs` file open in the editor. The code implements a background service that starts a scheduler, adds a job to fetch data from a service, and logs the results. The `Start()` method is asynchronous and uses `await` for scheduling and logging. The `Build()` method is used to configure the job and the trigger. The `RepeatForever()` method is used to keep the job running indefinitely. The `Shutdown()` method is used to stop the scheduler when the application is closed. The `Console.WriteLine()` method is used to log the results of the job.

```
13 {
14     1 reference | Administrator, 1 day ago | 1 author | 3 changes
15     public async void Start()
16     {
17         try
18         {
19             // Grab the Scheduler instance from the Factory
20             IScheduler scheduler = await StdSchedulerFactory.GetDefaultScheduler();
21
22             // and start it off
23             await scheduler.Start();
24
25             // Define the job and tie it to our HelloJob class
26             IJobDetail job = JobBuilder.Create<CurrencyJob>()
27                 .WithIdentity("job1", "group1")
28                 .Build();
29
30             // Trigger the job to run now, and then repeat every 10 seconds
31             ITrigger trigger = TriggerBuilder.Create()
32                 .WithIdentity("trigger1", "group1")
33                 .StartNow()
34                 .WithSimpleSchedule(x => x
35                     .WithIntervalInHours(2)
36                     .RepeatForever())
37                 .Build();
38
39             // Tell quartz to schedule the job using our trigger
40             await scheduler.ScheduleJob(job, trigger);
41
42             // some sleep to show what's happening
43             Thread.Sleep(TimeSpan.FromSeconds(60));
44
45             // and last shut down the scheduler when you are ready to close your program
46             await scheduler.Shutdown();
47         }
48         catch (SchedulerException se)
49         {
50             Console.WriteLine(se);
51         }
52
53         Console.WriteLine("Press any key to close the application");
54     }
55 }
```

The Solution Explorer on the right shows the project structure for `PalTripAdvisor`, which includes `App.config`, `ConfigurationService.cs`, `CurrencyFeedService.cs`, `CurrencyJobs.cs`, `LogstashHelper.cs`, `packages.config`, `Program.cs`, `DataSource`, and `PalTripAdvisor`.

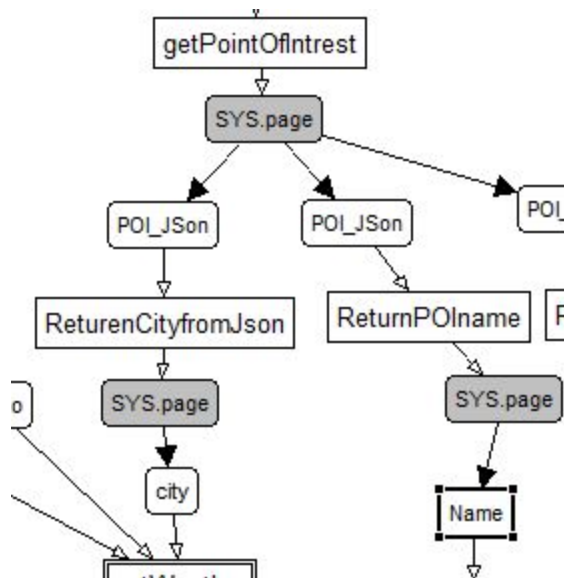




## 8. Challenges

We will discuss the challenges we faced in this project starting from the high level diagram to implementation as below

- Difficulty to simplify the application: we tried to make our application helpful as much as we could and this add some difficulties in adding and choosing services to provide with the time limitation of 4 month course.
- We faced problem in choosing the SOAP APIs since they are few.
- JOpera composition : we faced a challenge in implementing our big application in JOpera since we have many APIs with different types.
- We faced problem when reserving the needed server and we had to reserve another microsoft server in the last 3 days, since amazon closed the first server without reasons.
- When we tried to composite the APIs which we developed we surprised that the JOpera need a page between each 2 APIs to pass the output to the next API as below



- When we implemented the 4 services we want, we faced problem to pass the Json file from POI API toward getweather, getCurrency and getCloseHotel. Since these APIs need string or specific value but the getPointOfInterest APIs return Json file. We had to build another 3 APIs which convert from Json to specific values .
- We used Postman and Advanced testclient to test the APIs as well as below service

We used postman to test the service which take Json as input ( json from POI API-Rest service) and the output will be the city. The city will be used as input for the next API to get the closed hotels.

POI API: this API returns values from our DB , this rest API was developed by us.

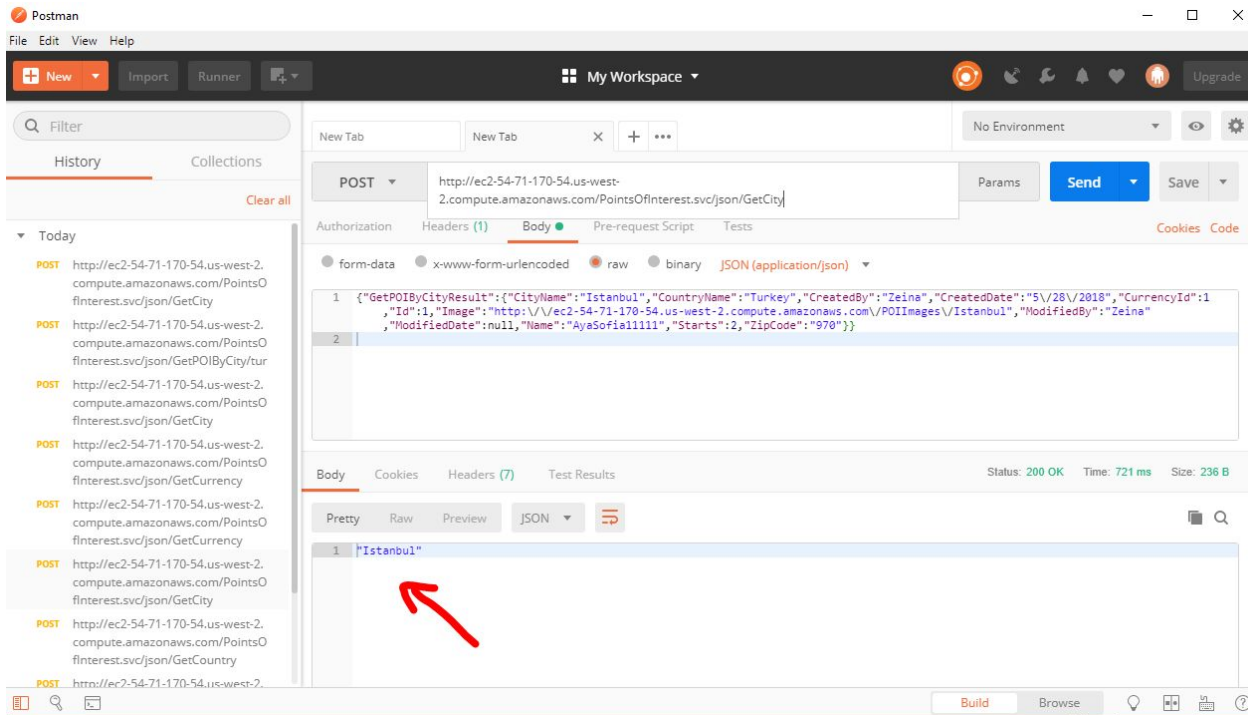
<http://ec2-54-71-170-54.us-west-2.compute.amazonaws.com/PointsOfInterest.svc/json/GetPOIByCity/turkey/istanbul>

Json:

```
{
  "GetPOIByCityResult": {
    "CityName": "Istanbul",
    "CountryName": "Turkey",
    "CreatedBy": "Zeina",
    "CreatedDate": "5/28/2018",
    "CurrencyId": 1,
    "Id": 1,
    "Image": "http://ec2-54-71-170-54.us-west-2.compute.amazonaws.com/POIImages/Istanbul",
    "ModifiedBy": "Zeina",
    "ModifiedDate": null,
    "Name": "AyaSofia11111",
    "Starts": 2,
    "ZipCode": "970"
  }
}
```



Then we developed another API to get city from the above Json, when we test it with POST method and the above Json in the Body we got the city name. And This rest API was developed by us as well:



- We created 3 APIs as the above example to get the country, city and zipCode to be used in the following APIs.
  - returnCityfromJson
  - returnCuurencyfromJson
  - returnZipCodefromJson

Then we passed the output of this three APIs toward two external APIs and one SOAP developed API.

## 9. Testing for the Preojce composition process and two developed Service

- PalTripAdvisor composition process ( TravelSuggestion)

### Start Process TravelSuggestionoml.TravelSuggestion [1.0]

country	Turkey
originalcurrency	USD
datefrom	05-05-2018
dateto	05-07-2018

[Get all instances of TravelSuggestionoml.TravelSuggestion \[1.0\]](#)

GET this content in:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [Plain Text](#) (text/plain)

localhost:8080/rest/TravelSuggestionoml/TravelSuggestion/1.0/

TravelSuggestionoml.TravelSuggestion [1.0].6  
Finished  
{Factor=<?xml version="1.0" encoding="UTF-8"?>  
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body>  
<ExchangeCurrencyResponse xmlns="http://tempuri.org/"><ExchangeCurrencyResult><Factor xsi:nil="true"/><MessageResponse>404, targeted currency does not exist, please make sure you entered  
the correct slug</MessageResponse></ExchangeCurrencyResult></ExchangeCurrencyResponse></soap:Body></soap:Envelope>, weatherdegree=<?xml version="1.0" encoding="UTF-8"?>  
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body>  
<GetWeatherByZipCodeResponse xmlns="http://tempuri.org/"><GetWeatherByZipCodeResult><WeatherStatus><WeatherResponseDomainObject><Day>5/5/2018</Day><Degree>27.00</Degree>  
<City>Istanbul</City><Country>Turkey</Country><zipCode>90</zipCode></WeatherResponseDomainObject></WeatherStatus><MessageResponse>200, result fetched  
successfully</MessageResponse></GetWeatherByZipCodeResult></GetWeatherByZipCodeResponse></soap:Body></soap:Envelope>, listOfHotel=<getHotelsByCityResponse xmlns="http://tempuri.org/">  
<getHotelsByCityResult xmlns:a="http://schemas.datacontract.org/2004/07/PalTripAdvisor" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><a:AveragePrice>55.00</a:AveragePrice>  
<a:CityName>"Istanbul"&#xD;  
</a:CityName><a:CountyName>Turkey</a:CountyName><a:CreatedBy>Qutaiba</a:CreatedBy><a:CreatedDate>5/5/2018</a:CreatedDate><a:Id>1</a:Id>  
<a:Image>http://qutaibamustafa.eastus.cloudapp.azure.com/POIImages/to be added later</a:Image><a:ModifiedBy i:nil="true"/><a:ModifiedDate i:nil="true"/><a:Name>Sheraton</a:Name>  
<a:Rating>5</a:Rating></getHotelsByCityResult></getHotelsByCityResponse>

TravelSuggestionoml.TravelSuggestion [1.0].6

**Finished**

```
{Factor=<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><ExchangeCurrencyResponse
xmlns="http://tempuri.org/"><ExchangeCurrencyResult><Factor xsi:nil="true"/><MessageResponse>404,
targeted currency does not exist, please make sure you entered the correct
slug</MessageResponse></ExchangeCurrencyResult></ExchangeCurrencyResponse></soap:Body></soap:Envelope>, weatherdegree=<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><GetWeatherByZipCodeResponse
```



```

xmlns="http://tempuri.org/"><GetWeatherByZipCodeResult><WeatherStatus><WeatherResponseDomainObject
><Day>5/5/2018</Day><Degree>27.00</Degree><City>Istanbul</City><Country>Turkey</Country><zipCode>
"90"</zipCode></WeatherResponseDomainObject></WeatherStatus><MessageResponse>200,
result fetched
successfully</MessageResponse></GetWeatherByZipCodeResult></GetWeatherByZipCodeResponse></soap:
Body></soap:Envelope>, listOfHotel=<getHotelsByCityResponse
xmlns="http://tempuri.org/"><getHotelsByCityResult
xmlns:a="http://schemas.datacontract.org/2004/07/PalTripAdvisor"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><a:AveragePrice>55.00</a:AveragePrice><a:CityNam
e>"istanbul"&#xD;
</a:CityName><a:CountyName>Turkey</a:CountyName><a:CreatedBy>Qutaiba</a:CreatedBy><a:CreatedDat
e>5/5/2018</a:CreatedDate><a:Id>1</a:Id><a:Image>http://qutaibamustafa.eastus.cloudapp.azure.com/POIIm
ages/to be added later</a:Image><a:ModifiedBy i:nil="true"/><a:ModifiedDate
i:nil="true"/><a:Name>Sheraton</a:Name><a:Rating>5</a:Rating></getHotelsByCityResult></getHotelsByCity
Response>}

```

#### - All instance

← → ↻ ⓘ localhost:8080/rest/TravelSuggestionoml/TravelSuggestion/1.0/\*/ ☆ 📄 📄 📄

**Note:** This HTML representation is meant only to help developers of clients of the JOpera REST API. It shows what information about the state of running processes can be read through simple HTTP GET requests. Click on the hyperlinks to filter the results. To read out the data, you do not have to scrape this HTML page. Instead, you can get the same information represented using the following formats by sending the corresponding mime types in the Accept HTTP header:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [ATOM](#) (application/atom+xml)
- [Plain Text](#) (text/plain)

TravelSuggestionoml.TravelSuggestion [1.0]

0 <a href="#">Input</a>	
<a href="#">country</a>	Turkey
<a href="#">datefrom</a>	05-05-2018
<a href="#">dateto</a>	05-07-2018
<a href="#">originalcurrency</a>	USD
<a href="#">Output</a>	
<a href="#">Factor</a>	<?xml version="1.0" encoding="UTF-8"?> <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><ExchangeCurrencyResponse xmlns="http://tempuri.org/"><ExchangeCurrencyResult><Factor xsi:nil="true"/><MessageResponse>404, targeted currency does not exist, please make sure you entered the correct slug</MessageResponse></ExchangeCurrencyResult></ExchangeCurrencyResponse></soap:Body></soap:Envelope>
<a href="#">listOfHotel</a>	<getHotelsByCityResponse xmlns="http://tempuri.org/"><getHotelsByCityResult xmlns:a="http://schemas.datacontract.org/2004/07/PalTripAdvisor" xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><a:AveragePrice>55.00</a:AveragePrice><a:CityName>"istanbul"&#xD;</a:CityName><a:CountyName>Turkey</a:CountyName>

getPointOfIntrest	Input	
	country	Turkey
	Output	
	POI_JSon	
System	ENDDT	2018-06-01 24:44:44.763
	HOST	10.100.17.211
	NAME	getPointOfIntrest
	PROGNAME	{TravelSuggestionoml}getPointOfIntrest[1.0]
	READYDT	2018-06-01 24:44:44.123
	RESTART	
	START	1
	STARTDT	2018-06-01 24:44:44.123
	STATE	4
	TYPE	ACTY
	WALL	0.64
	SystemInput	
	AM	POI_REST_HTTPAdapter
	method	GET
	urlstring	http://qutaibamustafa.eastus.cloudapp.azure.com/PointsOfInterest.svc/json/GetPOIByCity/Tur
	SystemOutput	
	errormsg	OK
	headout	null:[HTTP/1.1 200 OK] X-AspNet-Version:[4.0.30319] Date:[Thu, 31 May 2018 22:44:44 GMT] Content-Length:[312] Content-Type:[application/json; charset=utf-8] Server:[Microsoft-IIS/10.0] X-Powered-By:[ASP.NET] Cache-Control:[private]
	page	{"GetPOIByCityResult":
		{"CityName":"Istanbul","CountryName":"Turkey","CreatedBy":"Qutaiba","CreatedDate":"5/31/2018 22:44:44 GMT"}

## - Test for Developed REST Service getPointOfIntrest

We developed Here in this section , we will demonstrate testing for one service using JOpera and Postman

- The service is getPointofIntrest service in : <http://localhost:8080/rest/> as below :

[←](#)
[→](#)
[↻](#)
localhost:8080/rest/
☆
🔍
G
⋮

## Processes Published as RESTful Web Services

- [PAL\\_TEST.Test\\_getIntrest \[1.0\]](#) - 2018-05-28 20:21:42.388
- [TravelSuggestionoml.Test\\_ReturenCityfromJson \[1.0\]](#) - 2018-05-30 21:48:32.991
- [TravelSuggestionoml.Test\\_getCurrencyFactor \[1.0\]](#) - 2018-05-30 21:48:32.970
- [TravelSuggestionoml.Test\\_getCloseHotel \[1.0\]](#) - 2018-05-30 21:48:32.970
- [TravelSuggestionoml.Test\\_getWeather \[1.0\]](#) - 2018-05-30 21:48:32.986
- [TravelSuggestionoml.TravelSuggestion \[1.0\]](#) - 2018-05-30 21:48:32.955
- [TravelSuggestionoml.Test\\_getPointOfIntrest \[1.0\]](#) - 2018-05-30 21:48:32.991
- [TravelSuggestionoml.Test\\_ReturnPOIname \[1.0\]](#) - 2018-05-30 21:48:32.991
- [TravelSuggestionoml.Test\\_ResturnCurrencyfromJson \[1.0\]](#) - 2018-05-30 21:48:32.991

GET this content in:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [ATOM](#) (application/atom+xml)
- [Plain Text](#) (text/plain)

- For the targeted service POI, we run the service using Turkey :



localhost:8080/rest/TravelSuggestionoml/Test\_getPointOfIntrest/1.0/

## Start Process TravelSuggestionoml.Test\_getPointOfIntrest [1.0]

country Turkey

Start Run

[Get all instances of TravelSuggestionoml.Test\\_getPointOfIntrest \[1.0\]](#)

GET this content in:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [Plain Text](#) (text/plain)

- Then we got the Json file from the API:

localhost:8080/rest/TravelSuggestionoml/Test\_getPointOfIntrest/1.0/

```
TravelSuggestionoml.Test_getPointOfIntrest [1.0].1
Finished
{POI_JSon={GetPOIByCityResult":
{"CityName":"Istanbul","CountryName":"Turkey","CreatedBy":"Zeina","CreatedDate":"5/28/2018","CurrencyId":1,"Id":1,"Image":"http://ec2-54-71-170-54.us-west-2.compute.amazonaws.com/POIImages/Istanbul","ModifiedBy":"Zeina","ModifiedDate":null,"Name":"AyaSofia11111","Starts":2,"ZipCode":"970"}}}
```

- We got the below Plain txt , which was documented in the test log file for POI API
- We got also all instances for this API .

localhost:8080/rest/TravelSuggestionoml/Test\_getPointOfIntrest/1.0/

Note: This HTML representation is meant only to help developers of clients of the JOpera REST API. It shows what information about the state of running processes can be read through simple HTTP GET requests. Click on the hyperlinks to filter the results. To read out the data, you do not have to scrape this HTML page. Instead, you can get the same information represented using the following formats by sending the corresponding mime types in the Accept HTTP header:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [ATOM](#) (application/atom+xml)
- [Plain Text](#) (text/plain)

TravelSuggestionoml.Test\_getPointOfIntrest [1.0]

0	Input	country	Turkey
	Output	POI_JSon	{GetPOIByCityResult": {"CityName":"Istanbul","CountryName":"Turkey","CreatedBy":"Zeina","CreatedDate":"5/28/2018","CurrencyId":1,"Id":1,"Image":"http://ec2-54-71-170-54.us-west-2.compute.amazonaws.com/POIImages/Istanbul","ModifiedBy":"Zeina","ModifiedDate":null,"Name":"AyaSofia11111","Starts":2,"ZipCode":"970"}}}
	System	CALLER	
		delete_on_finish	
		ENDDT	2018-05-31 24:50:02.912
		Finalized	Yes
		READYDT	2018-05-31 24:50:02.399
		START	1
		STARTDT	2018-05-31 24:50:02.399
		STATE	4
		SuspendAfterCreation	
		WALL	0.513

getPointOfIntrest

Input	country	Turkey
Output	POI_JSon	
System	ENDDT	2018-05-31 24:50:02.912
	HOST	192.168.56.1
	NAME	getPointOfIntrest
	PROGNAME	{TravelSuggestionoml}getPointOfIntrest[1.0]

getPointOfIntrest

Input	country	Turkey
Output	POI_JSon	
System	ENDDT	2018-05-31 24:50:02.912
	HOST	192.168.56.1
	NAME	getPointOfIntrest
	PROGNAME	{TravelSuggestionoml}getPointOfIntrest[1.0]
	READYDT	2018-05-31 24:50:02.399
	RESTART	1
	STARTDT	2018-05-31 24:50:02.403
	STATE	4
	TYPE	ACTY
	WALL	0.513
SystemInput	AM	POI_REST_HTTPAdapter
	method	GET
	urlstring	http://ec2-54-71-170-54.us-west-2.compute.amazonaws.com/PointsOfInterest.svc/json/GetPOIByCity/turkey/istanbul
SystemOutput	errormsg	OK
	headout	null:[HTTP/1.1 200 OK] X-AspNet-Version:[4.0.30319] Date:[Wed, 30 May 2018 22:50:06 GMT] Content-Length:[319] Content-Type:[application/json; charset=utf-8] Server:[Microsoft-IIS/8.5] X-Powered-By:[ASP.NET] Cache-Control:[private]
	page	{GetPOIByCityResult": {"CityName":"Istanbul","CountryName":"Turkey","CreatedBy":"Zeina","CreatedDate":"5/28/2018","CurrencyId":1,"Id":1,"Image":"http://ec2-54-71-170-54.us-west-2.compute.amazonaws.com/POIImages/Istanbul","ModifiedBy":"Zeina","ModifiedDate":null,"Name":"AyaSofia11111","Starts":2,"ZipCode":"970"}}}
	responseheaders	{null:[HTTP/1.1 200 OK], X-AspNet-Version=[4.0.30319], Date=[Wed, 30 May 2018 22:50:06 GMT], Content-Length=[319], Content-Type=[application/json; charset=utf-8], Server=[Microsoft-IIS/8.5], X-Powered-By=[ASP.NET], Cache-Control=[private]}
	status	200



- The returned JSon was documented in POI API testing logs as well.
- **Developed SOAP Service getCurrencyFactor**

We created 2 SOAP services, this service to get the currency factor, and the input is destination currency from the previous API ( return currency from Json ) and user input with the original currency.

SOAP service :

<http://qutaibamustafa.eastus.cloudapp.azure.com/CurrencyExchange.asmx?op=ExchangeCurrency>

WSDL file :

CurrencyExchange1

Click [here](#) for a complete list of operations.

---

### ExchangeCurrency

**Test**

The test form is only available for requests from the local machine.

**SOAP 1.1**

The following is a sample SOAP 1.1 request and response. The placeholders shown need to be replaced with actual values.

```

POST /CurrencyExchange.asmx HTTP/1.1
Host: qutaibamustafa.eastus.cloudapp.azure.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/ExchangeCurrency"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ExchangeCurrency xmlns="http://tempuri.org/">
      <from>string</from>
      <to>string</to>
    </ExchangeCurrency>
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ExchangeCurrencyResponse xmlns="http://tempuri.org/">
      <ExchangeCurrencyResult>
        <Factor>decimal</Factor>
        <MessageResponse>string</MessageResponse>
      </ExchangeCurrencyResult>
    </ExchangeCurrencyResponse>
  </soap:Body>
</soap:Envelope>

```

**SOAP 1.2**

The following is a sample SOAP 1.2 request and response. The placeholders shown need to be replaced with actual values.

- The XML file is here as well

<http://qutaibamustafa.eastus.cloudapp.azure.com/CurrencyExchange.asmx?wsdl>

← → ↻ ① qutaibamustafa.eastus.cloudapp.azure.com/CurrencyExchange.asmx?wsdl ☆

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```

<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://tempuri.org/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://tempuri.org/">
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://tempuri.org/">
      <s:element name="ExchangeCurrency">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="from" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="to" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="ExchangeCurrencyResponse">
        <s:complexType>
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="ExchangeCurrencyResult" type="tns:CurrencyExchangeResponseDomainModel"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:complexType name="CurrencyExchangeResponseDomainModel">
        <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="Factor" nillable="true" type="s:decimal"/>
          <s:element minOccurs="0" maxOccurs="1" name="MessageResponse" type="s:string"/>
        </s:sequence>
      </s:complexType>
    </s:schema>
  </types>
  <message name="ExchangeCurrencySoapIn">
    <part name="parameters" element="tns:ExchangeCurrency"/>
  </message>
  <message name="ExchangeCurrencySoapOut">
    <part name="parameters" element="tns:ExchangeCurrencyResponse"/>
  </message>
  <portType name="CurrencyExchange1Soap">
    <operation name="ExchangeCurrency">
      <input message="tns:ExchangeCurrencySoapIn"/>
      <output message="tns:ExchangeCurrencySoapOut"/>
    </operation>
  </portType>
</definitions>

```

- Testing the API using localhost:8080

← → ↻ ① localhost:8080/rest/TravelSuggestionoml/Test\_getCurrencyFactor/1.0/ ☆

## Start Process TravelSuggestionoml.Test\_getCurrencyFactor [1.0]

originalcurrency	USD
distinationCurrency	"JD"

Start Run

[Get all instances of TravelSuggestionoml.Test\\_getCurrencyFactor \[1.0\]](#)

GET this content in:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [Plain Text](#) (text/plain)

- Get all instance result:

- [XML](#) (application/xml)
- [JSON](#) (application/json)
- [ATOM](#) (application/atom+xml)
- [Plain Text](#) (text/plain)

TravelSuggestionoml.Test\_getCurrencyFactor [1.0]

0	0	Input	distinationCurrency	"JD"
			originalcurrency	USD
		Output	currencyfactor	
		System	CALLER	
			delete_on_finish	
			ENDDT	2018-05-31 22:38:02.970
			Finalized	Yes
			READYDT	2018-05-31 22:37:56.301
			START	1
			STARTDT	2018-05-31 22:37:56.301
			STATE	4
			SuspendAfterCreation	
			WALL	6.669
		SystemInput	Response	Response to {TravelSuggestionoml}Test_getCurrencyFactor[1.0].0.0/1
	getCurrencyFactor	Input	distinationCurrency	"JD"
			originalcurrency	USD
		Output	Factor	
		System	ENDDT	2018-05-31 22:38:02.970
			HOST	192.168.56.1
			NAME	getCurrencyFactor
			PROGNAME	{TravelSuggestionoml}getCurrencyFactor[1.0]
			READYDT	2018-05-31 22:37:56.301
			RESTART	
			START	1
			STARTDT	2018-05-31 22:37:56.301
			STATE	4
			TYPE	ACTY
			WALL	6.669
		SystemInput	AM	SOAP_CURRENCY_INVOKEAdapter
			destination	http://qutaibamustafa.eastus.cloudapp.azure.com/CurrencyExchange.asmx
			message	<soapEnv:Envelope xmlns:soapEnv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:soapEnc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsdns0="http://tempuri.org/" xmlns:bns="http://tempuri.org/"><soapEnv:Body><bns:ExchangeCurrency soapEnv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><bns:from xsi:type="xsd:string">USD</bns:from><bns:to xsi:type="xsd:string">"JD"</bns:to></bns:ExchangeCurrency></soapEnv:Body></soapEnv:Envelope>
			SOAPAction	http://tempuri.org/ExchangeCurrency
		SystemOutput	message	<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><ExchangeCurrencyResponse xmlns="http://tempuri.org/"><ExchangeCurrencyResult><Factor>0.500</Factor><MessageResponse>200, factor exist.</MessageResponse></ExchangeCurrencyResult>

## 10. Future Enhancements

Starting from the current interest in travel web service and give suggestion about ( point of interest , weather , available hotel there and currency factor) .In future we want to get in details for more and more composition API like language API,booking flights . we also hope to improve our web service as website and mobile application the user can detect from anywhere.our plan is to cover all answers from travellers about anythings relating to there trips. This could be an entry point for encourage tourist to visit good and recommended

## 11. Conclusion

PlatripAdvisor is an API developed to cover the composition of web services (RestFul and WSDL) as shown above.learning what is REST API's and How to use and implement them achieved in our small application. We also learned about WSDL API and how to combine them. We can summarize the outcomes from this project as follows:

- 1- discover the market and existing API.
- 2- try to do reverse engineering to know how API's can be used in our application.
- 3- build new services from scratch and document them.
- 4- Learn the composition techniques and implementation methodologies.
- 5- Learn compositions general architecture and diagrams.
- 6- Discover new technology as Jopera and Azure.
- 7- learn how to test web service composition and check the correctness of data and control flow using JOpera.
- 8- Using many other testing clients for SOAP and REST APIs like Advanced REST client and Postman.
- 9- Teamwork and team spirit, we worked day and night to finalize this project as one team. We faced many challenges in JOpera, reserving the server, and building WSDL APIs and we came over them together.

**Thanks**