Java Web Services: A Performance Analysis

Pedro Costa e Silva[1] and Jorge Bernardino[1,2][a]
[1]Polytechnic of Coimbra – ISEC, Rua Pedro Nunes, Quinta da Nora, 3030-199 Coimbra, Portugal
[2]CISUC - Centre of Informatics and Systems of University of Coimbra, Pinhal de Marrocos, 3030-290 Coimbra, Portugal

Keywords: Web Services, Soap, Rest, Performance, SOA.

Abstract: Service-oriented architecture (SOA) is being increasingly used by developers both in web applications and in mobile applications. Within web services there are two main implementations: SOAP communication protocol and REST. This work presents a comparative study of performance between these two types of web services, SOAP versus REST, as well as analyses factors that may affect the efficiency of applications that are based on this architecture. In this experimental evaluation we used an application deployed in a Wildfly server and then used the JMeter test tool to launch requests in different numbers of threads and calls. Contrary to the more general idea that REST web services are significantly faster than SOAP, our results show that REST web services are 1% faster than SOAP. As this programming paradigm is increasingly used in a growing number of client and server applications, we conclude that the REST implementation is more efficient for systems which have to respond to less calls but have more requests in a connection.

1 INTRODUCTION

Service Oriented Architecture (SOA) is a software development model for distributed application components that incorporates, among other elements, access control, data mapping, and security features. SOA has two main functions. The first is to create a broad architecture model that defines the objectives of the applications that communicate with it and their approaches that will help meet those goals. The second function is to define particular implementation specifications, usually linked to the formal specifications of Web Services Description Language (WSDL) and Simple Object Access Protocol (SOAP).

For decades, software development has required the use of modular functional elements that perform a particular function at various places within the same application. With application integration operations and the trend of component sharing among resource pools, distributed client-server architectures, and database connections, companies needed a way to adapt their procedures-based development model to the use of remote and distributed components. Simple models such as Remote Procedure Call (RPC) were a start in the right direction, but the RPC did not have the data security and independence features required for truly open and distributed operations.

The solution to this problem was to redefine the old operating model in a new, broader and clearer architecture of services that could be delivered to an application using fully distributed software components. The architecture that involved these services in mechanisms to support full open-source security and management was called a service-oriented architecture, or SOA.

Initially, SOA implementations were based on available RPC and Object-Broker technologies in the early 2000s. But this architecture was quickly split into two implementations. The first branch is the Web Services (WS) field, which represents a highly organized and formal management of procedures and remote components. The second is the representational state transfer field (REST), which represents the use of a technology strand that accesses remotely hosted components.

The SOA WS model uses the WSDL to bind to interfaces with services and SOAP to define procedure or component APIs. The principles of WS were used to connect applications through a "business service bus" (BSB), which helped companies

[a] https://orcid.org/0000-0001-9660-2011

Silva, P. and Bernardino, J.
Java Web Services: A Performance Analysis.
DOI: 10.5220/0007979205700576
In *Proceedings of the 14th International Conference on Software Technologies (ICSOFT 2019)*, pages 570-576
ISBN: 978-989-758-379-7

integrate them into their applications, ensure their efficiency and improve their data management.
The SOA WS model never reached the adoption levels that its proponents had predicted; in fact, it collided with another model of remote components based on the Internet "language", the REST. RESTful application program interfaces (APIs) offered less overhead and were easy to understand

There are many commercial and open source tools available for testing web services. To test the performance of the implemented services we will use JMeter tool, one of the most used and documented. This tool will help you measure service quality and network performance in real time. The comparison is performed based on response time and usability (Radhakrishna and Nachamai, 2018).

In this work two web services were created, one using the SOAP implementation and the other REST. These web services are implemented on an application server (Wildfly) to which requests are made through the JMeter application. With the data obtained in the requests made by the test program (average order time, minimum order time, maximum order time and standard deviation), we evaluated the performance of each service by varying the number of threads (calls to the application server) or by varying the number of sequential requests on the same call. In this work we conclude that for several sequential requests in a single thread the REST architecture is more efficient.

The remainder of this paper is structured as follows. In section 2, we analyse the current state of art regarding web services and SOA architecture. Section 3 presents the methodology used and section 4 presents the results of the experimental evaluation. Section 5 discusses the results obtained and their implications. Finally, section 6 presents the conclusions and some future work.

2 RELATED WORK

Becker et al., (2009) compare two types of web services (SOAP and REST). This study presents a business network of manufacturers and service providers in the electronic area, for the implementation of a service-oriented architecture (SOA). For each of the types of web services analysed, a project with SOA architecture was developed and evaluated in relation to the previously defined set of requirements.

In the study by Belqasmi et al. (Belqasmi et al., 2012 a comparison of two web interfaces (a SOAP-based web service and a REST-based service) of multimedia conference applications is made. The results obtained in this study showed that SOAP-based request processing in a mobile environment can take 10 times longer and consume 8 times more memory than equivalent REST-based requests.

Tihomirovs and Grabis, 2016 summarize in their study the main advantages and disadvantages of REST and SOAP interfaces using evaluation of software metrics. Several metrics were analysed, namely cost, effort required for implementation or execution, efficiency, maintenance, etc.

Analysing the results, the researchers concluded that it is not possible to clearly identify the best approach to ensure data communication, and each project should be evaluated individually. Each protocol (SOAP and REST) has its advantages and disadvantages. However, it was possible to identify the main characteristics to choose the best approach. If the project requires great scalability, compatibility and performance the best option is to choose a REST service.

The complexity of the implementation, the execution speed, the consumed memory resources and the performance are better in the REST services when compared to the SOAP protocols. If the project requires mobile availability, REST is also the best choice.

If the project requires security, reliability and easy maintenance on the client side, the best choice will be the SOAP protocol. SOAP also has an advantage over REST if the project needs to process data asynchronously.

In the study by Malik et al. (Malik and Kim, 2017) a comparison of REST and SOAP interfaces is made in terms of ease of use, deployment and resource utilization. The objective of his study was to compare the two services through the projection of home networks based on the architectural styles of SOAP and REST.

In the study by Potti et al. (Potti, et al., 2012) a performance comparison of two service implementations (one based on SOAP and one based on REST) is made. SOAP and REST-based web services have been created that perform Create, Read, Update, and Delete (CRUD) operations on a database and retrieve local files. The authors used response times and file transfer rate metrics to compare the performance of both services. The results of this study reveal that, on average, the REST service performs better than the SOAP service, however not all the results were statistically conclusive. Through their study they concluded that the development of services using SOAP is easier since there is a greater support.

The development of services using REST was considered more difficult due to the need for high knowledge about the http protocol and the lack of support.

Mumbaikar et al. (Mumbaikar and Padiya, 2013) present in their study a comparison of the performance of SOAP and REST services based on different metrics of mobile applications and multimedia conferencing. The results of this research showed that REST services perform better than SOAP services.

In the study by Castillo et al. (Castillo et al., 2011) a high-level comparison is made between SOAP and REST. The results obtained show a significant time difference between REST and SOAP implementations. Although both are suitable for parallel systems development, SOAP implementations are heavier than REST implementations because XML increases the "translation" time of messages.

Wagh and Thool (2012) in their study make a detailed comparison between two frameworks used to provide web services through SOAP and REST and also address their problems and challenges. The comparison made can help in deciding which framework best suits wireless environments and which best responds to the needs of continuously accessing mobile web services from devices with limited resources.

In the work developed by Radhakrishna et al. (Radhakrishna and Nachamai, 2018) Two tools are described to test web services performance based on response times. In order to do this analysis, two test automation tools, JMeter and SoapUI, were analysed and the response times in each tool were analysed. The comparative study of the tools is done by performing an operation with different numbers of threads. The main difference in our study is that this paper explicitly measures the difference of performance (time to response) and only use a test tool (Jmeter) to avoid errors from application´s overhead.

3 METHODOLOGY

In order to verify the performance of the two types of services, a computer with an Intel Core i7-8550U @ 1.8Ghz processor with 16 Gb of Ram was used, a Windows 10 64-bit operating system (version 1803 build 17134.523) where two projects were created in Java 10 (version "10.0.1" 2018-04-17 with Java (TM) SE Runtime Environment 18.3 (build 10.0.1 + 10) and Java HotSpot (TM) 64-Bit Server VM 18.3 (build 10.0.1+ 10, mixed mode) in the IDE (Integrated Development Environment) Eclipse (Version IDE for Enterprise Java Developers Version 2018-12 (4.10.0) and Build id 20181214-0600.). In order to facilitate the construction of the deploy artefacts, we use Apache Maven (version 3.5.2) as a build tool since it is one of the most popular and used among java developers. To launch the requests, we used Jmeter (version 5.0 r1890935).

Two files were created, one for the REST web service project and one for the SOAP web service project.

For the construction of the REST endpoint was used the implementation of Oracle, Jersey version 1.19.4. In the construction of the SOAP endpoint we used the standard 2.0 implementation of JAX-WS in version 2.1.3 (Java API for standardized XML web services to create and consume SOAP services). The implementation used was that of Oracle, as well as for the REST service.

To test web services a service has been implemented that returns the first n prime numbers. The request to the web service should indicate an integer, for example five, and the answer will indicate the first five prime numbers in the format: "FIRST 5 numbers: 2 3 5 7 11". For this purpose, a Java method was created which was also used in both implementations of the service. So, in the REST service the class responsible was as follows.

```
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Path("/prime")
public class RestWSPerformance {
@Path("/{number}")
@GET
@Produces(MediaType.TEXT_PLAIN)
public String GetPrimes(@PathParam("number") Str
ing number) {
… JAVA code that returns a string with the first
n prime numbers
 }
}
```

The Web.xml file is the default deployment descriptor for the web application that the service is part of. In it are declared the filters and servlets used by the service. On the Java EE platform, the servlet listening for SOAP calls is the WSServlet that is part of the JAX-WS reference. The file developed for the SOAP project is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/j
avaee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" version="3.0">
<display-name>SoapWSPerf</display-name>
<listener>
<listener-
class>com.sun.xml.ws.transport.http.servlet.WSSe
rvletContextListener</listener-class>
</listener>
<servlet>
 <servlet-name>SoapWSPerf</servlet-name>
 <servlet-
class>com.sun.xml.ws.transport.http.servlet.WSSe
rvlet</servlet-class>
</servlet>
 <servlet-mapping>
  <servlet-name>SoapWSPerf</servlet-name>
  <url-pattern>/soapwsperf</url-pattern>
 </servlet-mapping>
</web-app>
```

Another important file in the implementation of a WS SOAP is sun-jaxb.xml. This file provides details about endpoints when JAX-WS Web services are deployed to servlet containers, such as Wildfly. This file is placed in the WEB-INF directory and contains the endpoint name, implementation class, URL pattern, and other additional information. The sun-jaxb.xml file used in the SOAP project was as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
 <endpoints
xmlns='http://java.sun.com/xml/ns/jax-
ws/ri/runtime' version='2.0'>
  <endpoint name='soapwsperf'
       implementation='pt.isec.si.SoapWSPerf'
url-pattern='/prime'/>
</endpoints>
```

As for the REST project the web.xml file used was as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/
javaee http://xmlns.jcp.org/xml/ns/javaee/web-
app_4_0.xsd" version="4.0">
<display-name>RestWSPerformance</display-name>
<servlet>
<servlet-name>RestWSPerformance</servlet-name>
<servlet-class>
com.sun.jersey.spi.container.servlet.ServletCont
ainer</servlet-class>
 <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
 <servlet-name>RestWSPerformance</servlet-name>
 <url-pattern>/restwsperformance/*</url-pattern>
</servlet-mapping>
</web-app>
```

Both services were implemented on Wildfly application server version 15.0.1.Final. To test the performance of both services the JMeter tool was used.

4 RESULTS

The comparison of web services is based on the analysis of the times observed during the execution of each operation. Each web service operation is invoked by a certain number of threads. For each data set (different thread numbers, different order numbers per thread, different response sizes) the average, maximum and minimum execution times of a request / response sequence are calculated. In each of the data sets we also measured the times in an individual execution of each web service and also the joint execution times (concurrent REST and SOAP requests).

The requests to the services indicated that the latter should respond by calculating the first 100,000 prime numbers. These values were decided, based on the load experiments, in order to have significant values, since if the service requested fewer prime numbers, the response was made too fast (in the order of 0 to 3 ms). Thus, it is only with this amount that the server response time allowed to obtain data of sufficient size to allow performance analysis.

Another important aspect was the choice of the total number of requests to keep the data consistent. After the tests carried out, it was concluded that 40 requests were a reasonable number of tests once we increased that number, and due to the typical timeout value for a response in Wildfly, the response time was exceeded and we would get errors in the calls services to be tested. After these values have been fixed, we have decided to divide the data into 3 distinct groups according to the number of threads (server connections) and requests per connection. For a better understanding the analysed datasets are described in Table 1.

Table 1: Description of the data sets used.

Prime Numbers	Total requests	# threads	# requests per thread
100000	40	10	4
100000	40	40	1
100000	40	20	2

The time were obtained through the JMeter tool and the different services were tested for each one of the referred data sets. For each dataset, three measurements were taken. One for SOAP requests, one for RESTs and one for both simultaneously. The observed results are described in Tables 2 and 3.

Table 2: Time observed in requests by service (10 threads with for 4 requests).

WS	# requests	Average (ms)	Min (ms)	Max (ms)	St deviation
REST	40	29794	27596	31121	932.62
SOAP	40	29903	27430	31704	1166.15

Table 3: Time observed in simultaneous requests (10 threads with for 4 requests).

WS	# requests	Average (ms)	Min (ms)	Max (ms)	St deviation
REST	40	60704	53409	68194	3935.13
SOAP	40	60007	49572	70297	5264.34

In Table 2 and Table 3 the observed values for the execution of the different web services are registered, for a request of the first 100,000 prime numbers. There were 40 requests divided by 10 threads with 4 requests each, and for individual and simultaneous execution respectively. The observation of results allows to conclude that, for requests made separately (only REST requests, or only SOAP requests), the average execution time is lower for REST web services than for SOAP, however the minimum value was observed for a SOAP request. When executed simultaneously, lower and lower average times were observed in SOAP web services, compared to REST requests, however the observed standard deviation is much higher which indicates that there is a greater dispersion of the sample data.

Table 4: Time observed in requests by service (40 threads with for 1 requests).

WS	# requests	Average (ms)	Min (ms)	Max (ms)	St deviation
REST	40	122556	97697	129197	7786.56
SOAP	40	119883	90698	126885	9229.43

Table 5: Time observed in simultaneous requests (40 threads with for 1 requests).

WS	# requests	Average (ms)	Min (ms)	Max (ms)	St deviation
REST	40	262956	193942	276924	20846.69
SOAP	40	262904	205439	277604	18349.01

When ordering in 40 threads (Table 4 and Table 5) with only one order in each one, it can be seen that the differences between the averages are practically nil. However, it is also observed that in individual requests the standard deviation of REST requests is less than the standard deviation of SOAP requests, which indicates that the data dispersion is smaller. On the other hand, in concurrent requests, the opposite is observed, a lower standard deviation for SOAP requests, which indicates that the dispersion of sample data is smaller than in REST requests.

To complete the tests, we re-measure the values this time by sending two requests per thread in 20 threads. The values were those recorded in Table 6 and 7.

Table 6: Time observed in requests by service (20 threads with for 2 requests).

WS	# requests	Average (ms)	Min (ms)	Max (ms)	St deviation
REST	40	59954	50627	64452	3317.32
SOAP	40	62651	53524	68204	3981.41

Table 7: Time observed in simultaneous requests (20 threads with for 2 requests).

WS	# requests	Average (ms)	Min (ms)	Max (ms)	St deviation
REST	40	121045	99986	135177	9529.92
SOAP	40	121743	94213	135822	10703.33

From the analysis of Table 6 and Table 7 it can be observed that for individual requests, all registered values were lower in REST requests than in SOAP requests. In the simultaneous requests, the average and the maximum value observed were smaller in the REST requests, being only the smallest minimum value in the SOAP requests. In both cases, the standard deviation is lower in REST requests, which means that sample dispersion is lower in these requests than in SOAP requests.

By analyzing the chart in figure 1 we can see that for asynchronous requests with fewer threads and more requests per thread the average response times of REST services are less than the average times for SOAP services. When the number of threads (value = 20) is increased, the difference between the mean

response times is no longer significant, even though the REST service values remain lower. When the number of threads increases even more (value = 40), the observed trend reverses, since, although not very noticeable, the average response times of the SOAP service are lower than the REST service times.

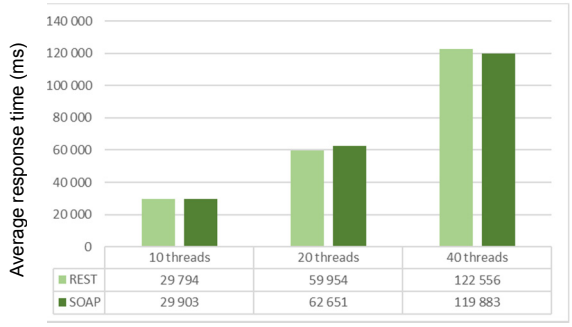

Figure 1: Comparative chart of the mean values for asynchronous requests.

Unlike asynchronous requests, when the concurrent request values are observed, only with 20 threads the average time of the REST service is less than the average time of the SOAP service. For an execution with 10 or 40 threads, the average time of the SOAP service, although notorious, is less than the average time observed in REST requests.

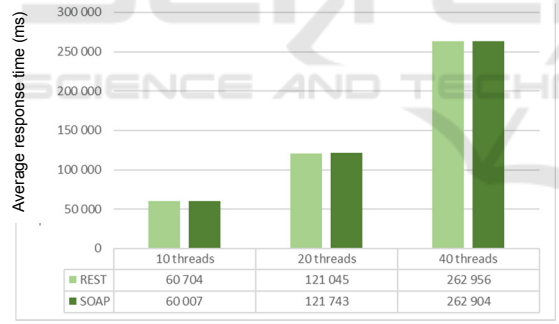

Figure 2: Comparative chart of the mean values for simultaneous requests.

When executing REST requests isolated from SOAP requests, it is verified (Figure 3 and Figure 4) that for a smaller number of threads (value = 10) the difference between the mean, minimum and maximum values is practically imperceptible. As the number of threads increases, the difference between these values is also increasing. It is important to note that the difference between the two cases (SOAP requests and REST requests and 20 and 40 threads) is greater than the difference to the maximum value.

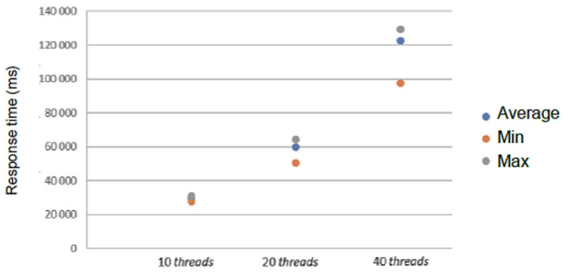

Figure 3: Comparative chart representing the average, minimum and maximum times of REST requests when executed separately.

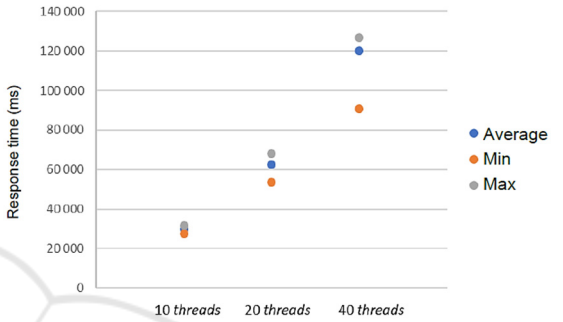

Figure 4: Comparative chart representing the average, minimum and maximum times of SOAP requests when executed separately.

From the analysis of the graph of Figure 5 it is possible to conclude that for both web services the average execution time for asynchronous requests is much lower when compared to concurrent requests for the same number of threads.

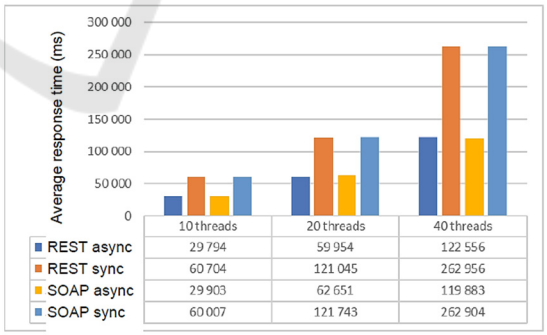

Figure 5: Summary chart of average order execution times.

5 DISCUSSION

In the individual 10- and 20-threaded tests the REST implementation is 1% faster than SOAP, however, if we increase the number of threads (in our case 40), SOAP becomes 1% faster. This apparent performance

gain from the SOAP implementation may be linked to the server's internal latency, since 40 requests are made for 1 second, only on a server connection.

In the concurrent tests, that is, when the SOAP and REST requests were sent to the server at the same time by Jmeter, the results were not conclusive, since SOAP was faster with 20 threads and REST with 10 and 40 threads. Here the results can be explained by the way Wildfly treats requests to the server and internally forwards them to the intended service, since the two services are available, the results may have been tainted by internal rules of order forwarding for web services many different. Still, REST remains 1% faster.

In order to better gauge the results, we can also verify that as the requests per thread increased, the dispersion of data accompanies this same difference, so there is a greater difference between the values of the maximum and minimum response time. This is confirmed by the values of the standard deviation that are significantly larger in the 40 threaded assays.

6 CONCLUSIONS AND FUTURE WORK

This paper quantifies a pre-established idea that REST type services are faster than SOAP because of the "machinery" they bring on the server side. After a series of trials, it was concluded that actually the REST implementation is faster, but only at 1% than SOAP. However, it has also been proved that, under certain conditions, when many requests are made in a single thread, SOAP is slightly (1%) more efficient. The essays focused only on a web service that returned a text and it would be interesting to approach in another similar study, the use of more complex objects in the services answers, for example through Json or Xml more elaborated. The language used in this article was Java but this work could also be replicated in another language, for example .NET.

REFERENCES

Becker, J., Matzner, M., & Müller, O. (2009). Comparing architectural styles for service-oriented architectures - A REST vs. SOAP case study. *Information Systems Development: Towards a Service Provision Society*, 207–215. https://doi.org/10.1007/b137171_22

Belqasmi, F., Singh, J., Bani Melhem, S. Y., & Glitho, R. H. (2012). SOAP-based vs. RESTful web services: A case study for multimedia conferencing. *IEEE Internet Computing*, *16*(4), 54–63. https://doi.org/10.1109/MIC.2012.62

Castillo, P. A., Bernier, J. L., Arenas, M. G., Merelo, J. J., & Garcia-Sanchez, P. (2011). *SOAP vs REST: Comparing a master-slave GA implementation*. Retrieved from http://geneura.wordpress.com

Malik, S., & Kim, D. H. (2017). A comparison of RESTful vs. SOAP web services in actuator networks. *International Conference on Ubiquitous and Future Networks, ICUFN*, 753–755. https://doi.org/10.1109/ICUFN.2017.7993893

Mumbaikar, S., & Padiya, P. (2013). Web Services Based On SOAP and REST Principles. *International Journal of Scientific and Research Publications*, *3*(5), 3–6. Retrieved from www.ijsrp.org

Potti, P. P. K., Ahuja, S., Umapathy, K., & Prodanoff, Z. (2012). Comparing Performance of Web Service Interaction Styles: SOAP vs. REST. *Proceedings of the Conference on Information Systems Applied Research ISSN*, *2167*, 1508. https://doi.org/http://dx.doi.org/10.1016/j.plantsci.2009.07.009

Radhakrishna, S., & Nachamai, M. (2018). Performance inquisition of web services using soap UI and JMeter. *2017 IEEE International Conference on Current Trends in Advanced Computing, ICCTAC 2017*, *2018–Janua*, 1–5. https://doi.org/10.1109/ICCTAC.2017.8249993

Service-oriented architecture (SOA) news, help and research - ComputerWeekly.com. (n.d.). Retrieved January 12, 2019, from https://www.computerweekly.com/resources/Service-oriented-architecture-SOA

Tihomirovs, J., & Grabis, J. (2016). Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics. *Information Technology and Management Science*, *19*(1), 92–97. https://doi.org/10.1515/itms-2016-0017

Wagh, K., & Thool, R. (2012). A Comparative Study of SOAP Vs REST Web Services Provisioning Techniques for Mobile Host. *Journal of Information Engineering and Applications*, *2*(5), 12–16. Retrieved from www.iiste.org