**REST vs SOAP and RPC based systems**

During the creation of this API I have considered the design principles and constraints of Representational State Transfer (REST). Whilst REST may not be perfect, it has many benefits over alternative methods. The most commonly used alternative to REST is SOAP (a predecessor of XML-RPC), used to create RPC-based systems.

One of the main factors that drove this decision is the nature of REST and SOAP. REST focuses on managing resources and data through CRUD operations, achieved through a single and consistent interface (most commonly HTTP). SOAP, however, is focused on operations (application logic) and can be used across a variety of underlying protocols. It is the single interface that makes REST much simpler than SOAP which, as mentioned, can use a number of underlying protocols such as SMTP, FTP as well as HTTP to perform its operations. Not only is REST simpler, it is lightweight in comparison to SOAP. REST rides directly on HTTP meaning that HTTP is all that is needed to send data/request and get the required response through the use of URIs. SOAP messages, however, require additional information detailing how the message should be processed using RPC methods. As a result, SOAP messages are not human-readable whereas REST resource representations can be in different formats, such as JSON and XML, and are both human and machine readable.

As the client in this project uses JavaScript and AJAX, it made sense to use a REST based service to accompany it. Although it is possible to call SOAP from JavaScript and AJAX, it is much easier to call a REST service due to its HTTP interface. REST uses nouns as resources and verbs as methods. The verbs are usually HTTP methods such as GET, POST, UPDATE, PUT and DELETE which are easily sent using AJAX and JavaScript through the built in HTTP Request method (this can be seen in app.js). SOAP uses nouns and verbs, such as getUser, to make data available as services. Because this is not built into the HTTP Request it is much more difficult to implement and in general means that SOAP requires more learning that REST.

Another factor is performance. REST is much faster than SOAP. REST is less CPU intensive and uses less bandwidth; REST is the best option because it can work well even on a limited bandwidth.

REST is not always ideal. Although REST only works over HTTP making it simple, this can also be seen as a disadvantage too because it limits REST to a certain type of application.

SOAP has a service contract, unless this is implemented into REST anything goes. If a strict contract is needed between the client and server, REST may not be the best approach. In addition, security and authentication is much harder to enforce over REST in comparison to SOAP.

REST is now much more popular among web developers than SOAP meaning there is more support and more APIs that can be used and/or implemented, further enforcing the idea of the open web. An example of this can be seen in my project where I have used another RESTful API to update the rates within my application.

**My Project**

How might the application you have built be extended and improved?

How does your application and/or its components promote re-use?