NATHAN CAIRNS | NCAI761 | 2726772286

SOFTENG325 ASSINGMENT ONE REPORT

Scalability is an incredibly important requirement for modern web services. With the potential for an application or service to grow rapidly at short notice a service must be able to adapt accordingly. The Seat Booking Service Project was made scalable through statelessness, caching and concurrency control.

The web service is a stateless REST service. This means that the service itself can easily be replicated across multiple servers. Replication is a quick and easy way to scale up a web service. When a client interacts with any replica it will have the same effect as calling the same method on any other replica. This effect is achieved via the state of the service being stored in a database. Each replica points to the same database meaning any replica can be interacted with, achieving the same effect. This stateless is accented by the resource-per-request life cycle used by the web service's resource classes.

State is communicated between client and server using cookies and authentication tokens. Upon authentication the server stores an authentication token against the client in the database and sends the same token to the client as a cookie. In subsequent requests the client sends the token, the server then checks the token against entries in the database and uses it to identify the client. This identification can be done by any replica of the web service as described in the previous paragraph.

The web service uses client-side caching to minimize the number of requests made to the server. Whenever a client requests a large amount of data (e.g. all performers, all concerts, all bookings, performer images) the client will store these values in a cache. If the client requests this data again it will just use the cached version of the values. If the cache becomes stale (expiry time set by server) the client will make another request to the web service to update it. Using caching like this minimizes the number of requests made to the web service and hence reduces server load. This will ensure as more users use the service unnecessary requests will be avoided, reducing blocking code and scaling appropriately.

The web service uses optimistic concurrency control to allow multiple instances to interact with the database at once. Optimistic access means multiple instances can access the same record and will only fail to commit their changes if the record has changed since it first read it. This checking is done using incremental version numbers. Having concurrent access is essential for a large system. Database records need to be accessible by multiple clients at once to ensure reasonable access times. Furthermore, records need to be able to be safely edited such that consistency is maintained. Optimistic concurrency control means requests are concurrent, consistent and efficient. Hence adding to the scalability of the system.

One aspect of the project which is not very scalable is the NewsItemResource. This resource is a singleton class and retains state. This is the only aspect of the service which retains state and is hence not very scalable. This was done due to the way asynchronous requests had to be processed. The implementation of this resource would probably have to be changed if the service was required to be scaled up significantly.

In Conclusion, the Seat Booking Service project is quite a scalable application, able to adapt to significant load changes. This was done using caching, statelessness and concurrency control. However, the news item aspect of the service has limited scalability with its current implementation.