Brian Engel

Modular Four Journal

The client-server pattern is a good choice in a web-based game running on multiple operating platforms. One reason for this is that it supports efficient delivery of data to the client from the server and is easy to secure. Also, the client can use any operating system or programming language they want, so it is ideal for a program that will be running on more than one machine, or is web based.

The REST API style is what makes this possible. REST stands for REpresentional State Transfer. There are six guiding principles for RESTful architecture. They have a uniform interface which keeps the architecture simple. They are client-server, which help both the client and server evolve separately and makes it easier to use across different systems. They are stateless, which means the request must contain all necessary information, and cannot rely on previous information from the server, making it necessary for the client to keep track of the entire session. They are labeled as cacheable or non-cacheable, so the client can decide to reuse the cacheable data. It is a layered system, so that a component cannot see beyond the layer they are interacting with. The last thing is Code on demand, which is the servers sending the client code, and the client just has to execute the code.

On the server side in REST architecture, the server can only answer a request for information from a client and cannot ask for information from the client. On the client side of the REST architecture, the client can only ask for information from a server and cannot send information to the server. Because of this relationship, the client must keep track of the state of the program, since the server would not be able to since it doesn’t get any information from the client. This keeps track of the passwords, usernames, and user types. The client sends a name and password and the GameAuthenticator validates that the name and password match up and GameAuthorizer validates if the user is authorized for the action chosen, then this info is sent back to the client. This makes it harder to change code in the client to gain access to the program and keeping it more secure.

In general, a developer for the client side of program that can run in multiple environments would have to develop each of the different clients to be able to run on each of their different operating systems, but if it is running off of a specific web browser, you would only have to make sure it worked on that specific browser. For example, you could be running one on Windows, one on macOS, and one on linux, and if you were using the same web browser on all three operating systems, you wouldn’t have to change anything between the three.

I’m not absolutely positive on adding a new user to the existing code in the game application, but it looks like the only way to add a user at this point is to hard code it into the HashMap in GameUserDB.java. This actually leads into some new functionality and features that would be useful in the app. A create new player that is accessible for any user would be useful, and an add player and delete player would be useful for the admin. If the game application wanted to add a xbox client and a ps4 client, as long as they had access to the same web browser as the other clients, you wouldn’t really have to do anything differently. That’s the beauty of the client-server architecture and RESTful architecture. It only sees the layer that it's on so it doesn’t have to worry about the operating system.