Ty Simpson

CS-230

Randolph

1/29/2023

Module 4 Journal Entry

The client-server pattern is useful for when servers and clients communicate over the web. The client-side is everything the user sees and/or interacts with. Clients request data and information from the server and then the server provides (serves) that information. Multiple clients can request data from a single server. Within this pattern, there is a separation of concerns between the client-side and server-side components. This improves the capabilities to port applications across multiple platforms – which is a key requirement for the game application.

In REST (REpresentational State Transfer) API style, the server responds to HTTP requests made by the client. In turn, the server returns data in a standardized format, such as JSON or XML. This is done in a stateless environment, which means each client request and server response is isolated.

You wouldn’t want any old user to delete information from your application – having authenticated and authorized users help secure your data by only allowing certain roles to make specific calls to the server. In our earlier assignment, we used the principal interface which represents the authenticated user. The authenticator class was used to determine if the provided credentials for a user were valid. We used the authorizer class to match roles with certain actions – this would allow a user to only take the actions the authorizer permits them to do. You can then implement the @Auth annotation into the API secure it. Using @PermitAll means that a user must be authenticated (logged in) to take that action – but is not restricted by their role. However, we use @RolesAllowed, the user must have a specified role to call the method. For example, if the @RolesAllowed( {“ADMIN”} ) tag was used, only Admins could call that method.

Developers on the client-side are responsible for what the end-user interacts with on their respective platforms. This could be front end, like the UI a user may see when playing the game, or back-end, like if a new player gets registered. A client-side developer may want to consider the best way to present information to the player in a way that’s easy to interact with. For instance, someone with vision problems may want to play the game. They may want to consider adding accessibility features into the game like screen readers or a high-contrast display. Additionally, they may want to think about ways to personalize the user experience by developing user customization features like creating player avatars. Over at the back-end, it’s important that new users are assigned a unique identifier so that when they are added to the database there can only be one of that particular user. Using these identifiers would also allow users to change their information if desired. The code that’s used to develop for the client end should also be carefully considered – especially if there’s a desire to develop for multiple platforms. If The Gaming Room wanted the game to compatible on additional platforms, like the PS5 or Nintendo Switch, it’s important that whichever language is selected, it needs to be compatible with whatever OS the game is being designed for.