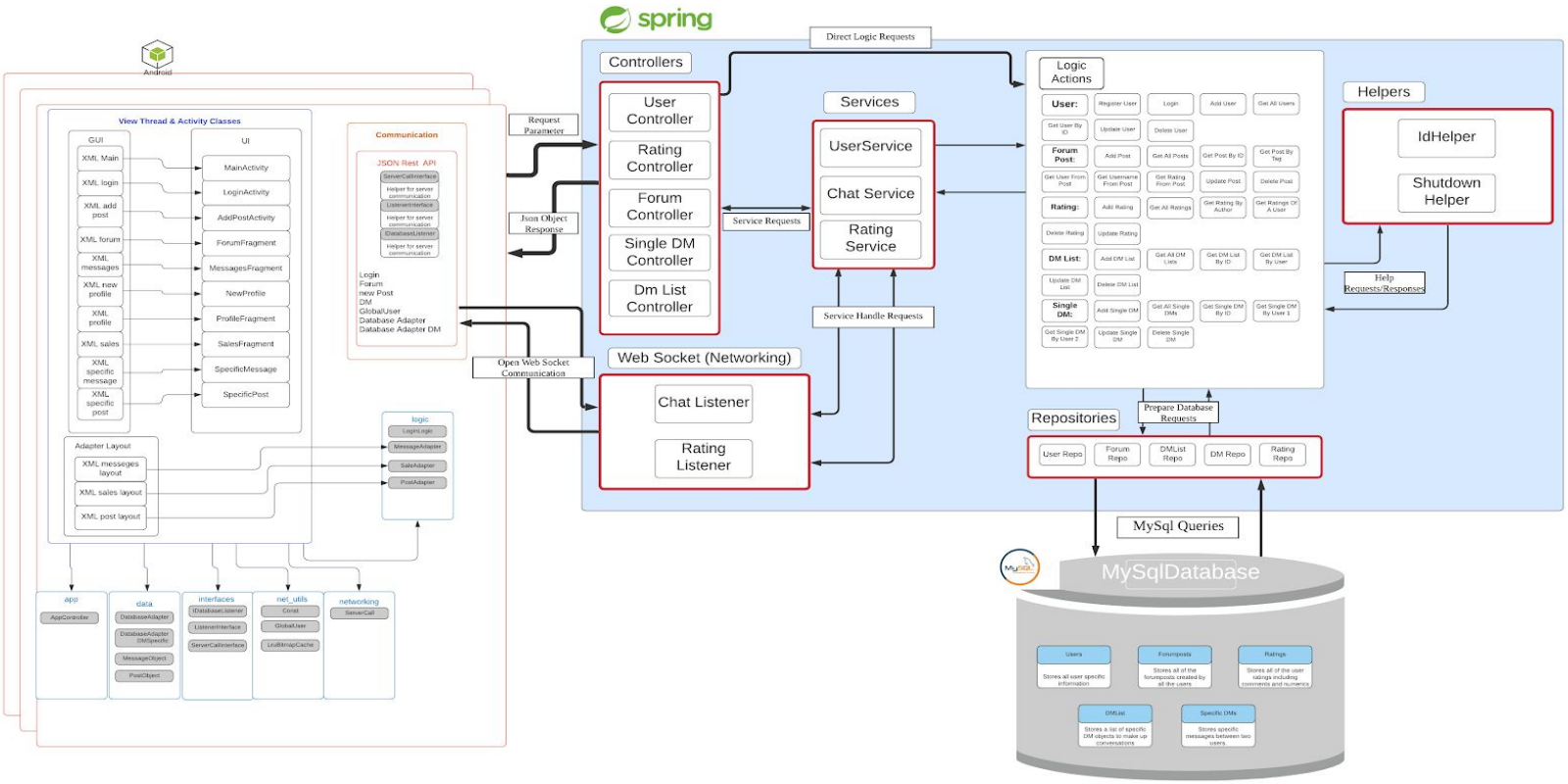
Design Document fo KK08 - CyTicket

Group KK-08
Robert Brustkern: %25
Harry Russell-Pribnow: %25
Logan Kroeze: %25
Carson Campbell: %25

Block Diagram



Design Description

FrontEnd:

The app begins its lifetime on the login activity, where it prompts the user to enter their username and password or create a new profile. If the new profile button is clicked the user is taken to the new profile activity where they will enter their user information. When they select the submit button the class will call the Servercall class to check if they entered valid information, and be sent back to the login activity if they did. When the login button is selected the activity will call the loginLogic class that will call the serverCall class to check if the username and password entered is valid. If they are valid then the LoginLogic class will call GlobalUser to set the user information, and set the GlobalUser object in the appController class to make it available throughout the app's lifetime. Then LoginLogic will call DatabaseAdapter to refresh the list of forum posts, and send the user to the mainActivity.

The main activity contains tabs for the sales, messages, forum, and profile fragments. The profile fragment calls the GlobaUser class to get and display the user information. The forum fragment retrieves calls Database adapter to retrieve the posts from the local database, and calls the PostAdapter to display the retrieved posts on the list. The fragment contains an add post button that, when clicked, brings the user to the addPost activity. The addPost activity prompts the user to enter the information needed to post to the forum. When the user clicks on the submit button the activity calls the ServerCall class to post the new post to the database, then it will refresh the local database of forum posts before returning to the forum fragment. The sales fragment calls the ServerCall class to get forum posts and displays them using the salesAdapter. The messages fragment calls the DatabaseAdapterDM to refresh and get DMs, and displays the DMs with the messagesAdapter.

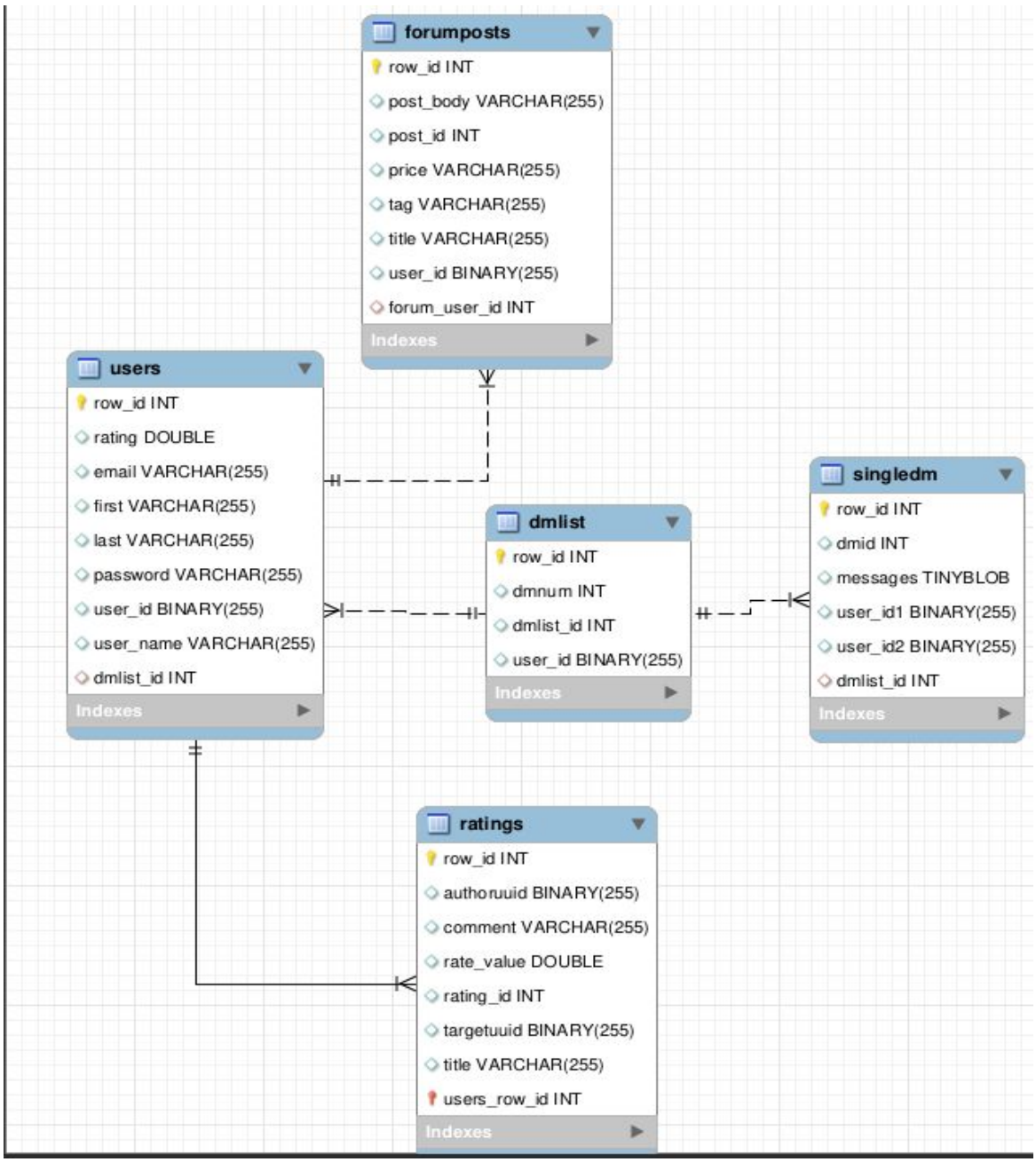
Backend:

The first important aspect of the backend is how it communicates with the front end. It has two methods of retrieving and sending data: Controllers and Websockets. The controllers handle any data that needs to be stored or retrieved from the database. The websockets on the other hand are responsible for more event based requests, such as chat messages or rating updates. It is important to note, that when information is passed between the websockets, that information is forwarded to the services module, which will handle the requests accordingly. However, as of now, that has yet to be completely implemented, so the overall design is unclear.

The controllers however are completely functional. Once they accept data in via HTTP requests the data is processed based on the logic operation that is being performed. This will then determine two things. Whether a service needs to be utilized, and what data needs to be read/written from the database. Each controller has direct access to the services that it may need to manipulate/create data for further calculations. It also has a path to the respective repositories that can handle the data requests. Because we use a MySQL database, the repos will craft MySQL queries based on what the logic demands. Those queries then are passed to the database, and the data is forwarded back up the chain to the controllers, where they can be sent back if needed via JSON objects.

Finally, there is also a side helper module. This just includes classes that assist in certain logic operations, such as id creation.

Table Relationships



Relations: There are a few relationships between the tables. These include a one-many relation between users and forums/ratings. There is also a one to many relationship between each dmlist and a specific dm. Finally, there is a many to one relationship between the users and the dmlist.