Pickmeup

Block Diagram

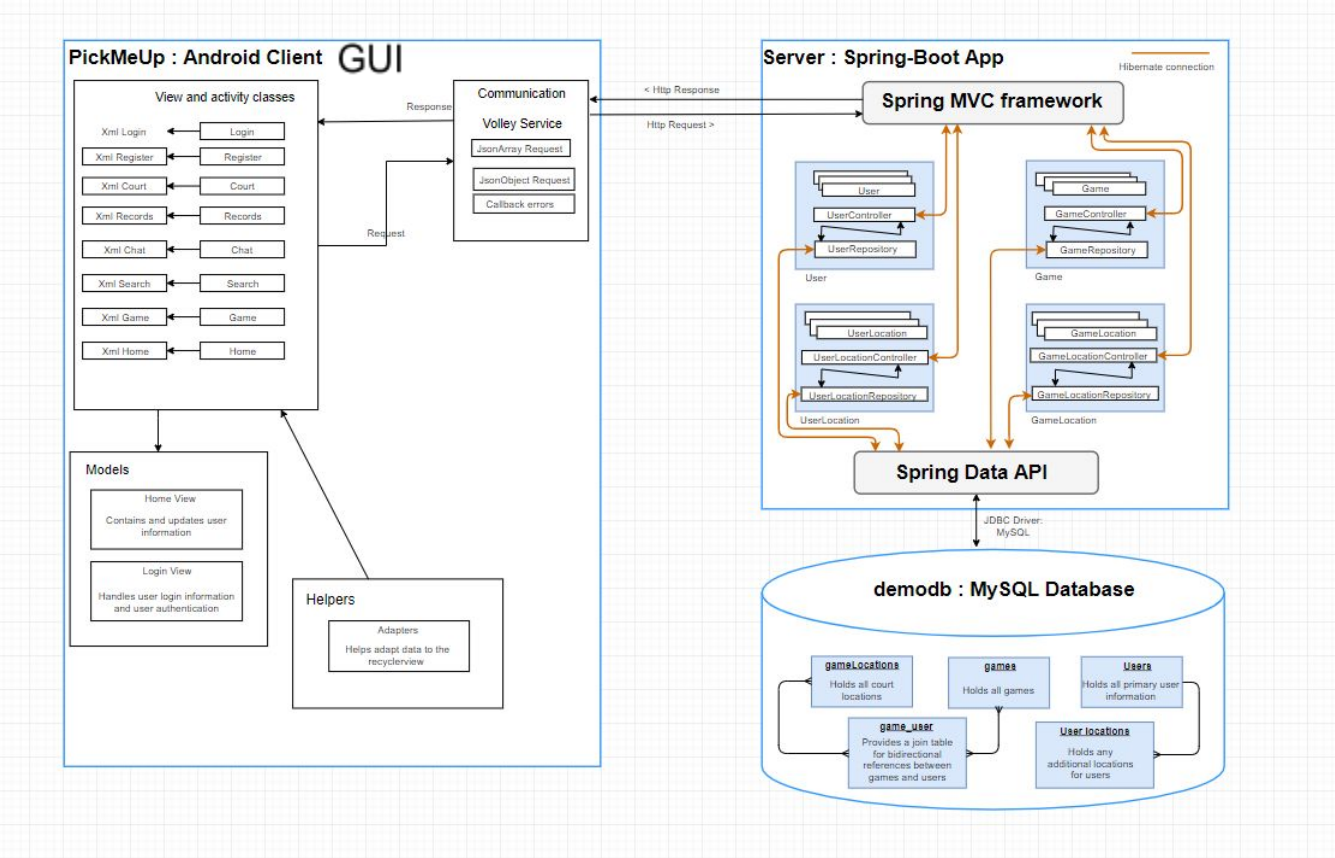
Jonathan Novak

Chad Morrow

Declan Costello

Jonathan Limon-Chavez

Diagram



Our client consists of communications, models, helpers, view and activity classes. The server consists of network controllers, data repositories, and abstract data models (java objects to represent our tables) for the user, user location, game, and game location related HTTP requests; as well as the Spring MVC framework and Data API that ties it all together. The client's activity classes and the server's Spring MVC framework push and request from the client communication's volley service. The Server uses Spring's MVC Framework to map, parse, and handle network requests/responses using a controller. Then, with implemented data repositories and Spring's Data API, data is stored, retrieved, and communicated to our MySQL database. Our MySQL database holds tables for courts (game locations), games, users, and additional(secondary) user locations. We use the Google Maps API to pinpoint the user's location, but never display the map form of it.

Design Descriptions

Android GUI -

At the moment our Android client-side has 8 activities with corresponding XML files. The XML files contain most of the layout and customization, with a few commands programmatically controlled within the activity and also in the Adapter classes. Some values are passed between activities, such as court information.

Android Models -

We have HomeViewModel and LoggedInUser which contain user information accessible on all activities. We use this information across the app for pulling backend data and displaying personal information on the settings screen.

Android Helpers -

We have multiple Adapter helper classes for every RecyclerView screen we have. This includes: courts, games and chat. The adapters help translate the information we load, into the XML formatted layout we created.

Android Communication -

Volley is used for all front to backend communication. The volley library manages and services requests on threads separate from the UI thread. We use the library for grabbing user, court, and game info. We can recognize distinct errors when communicating in different environments.

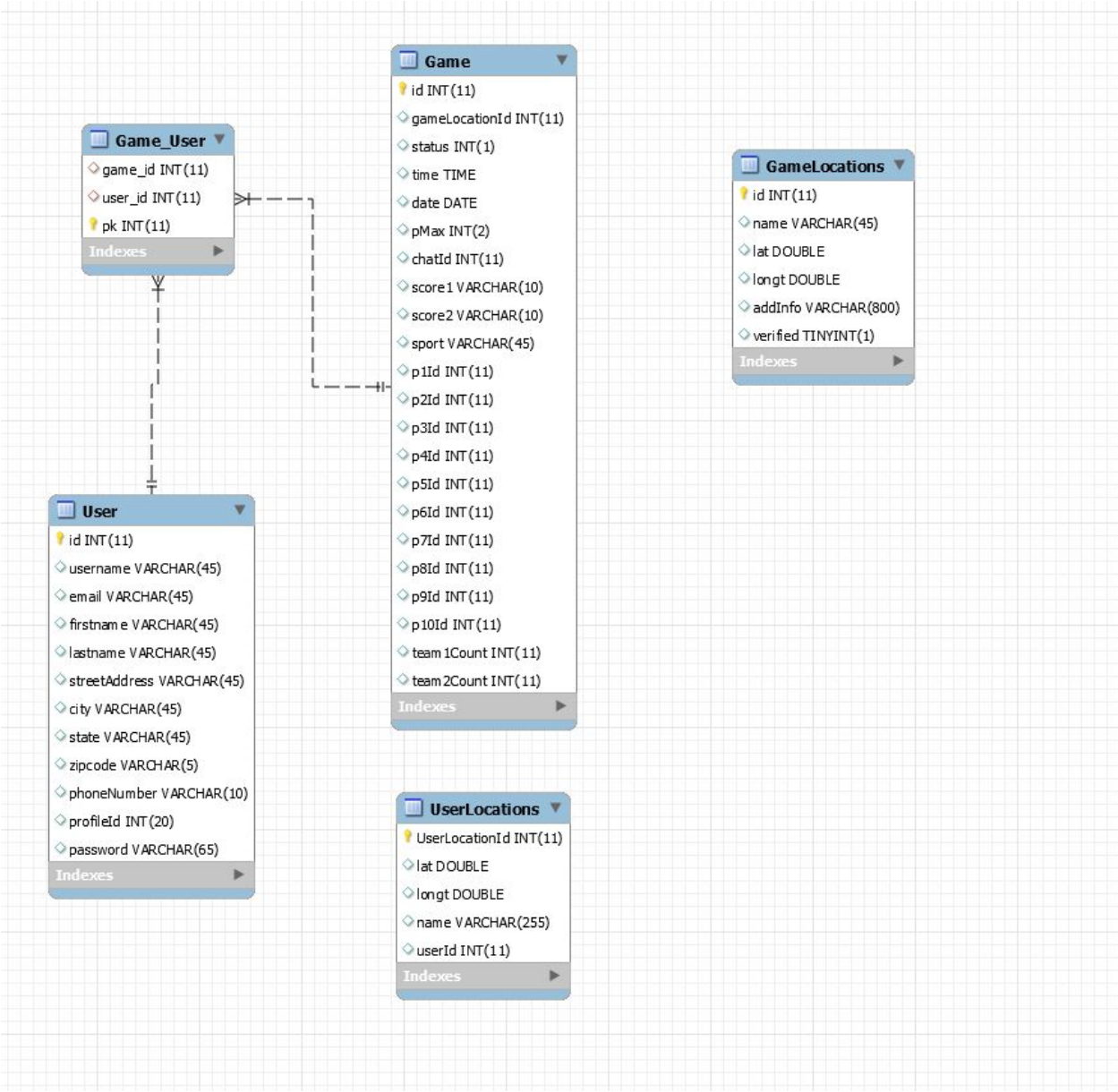
Spring MVC Framework -

Spring boot's Web Starter library is used to help configure our application based on our needs and with minimal understanding of lower level services/processes. Combined with Hibernate mapping we can simplify our server's web endpoints, focus on request handling/feedback, and better organize our classes.

Spring Data API -

Spring's Data starter library is used to help us store data and interface with our MySQL database using hibernate mappings, objects to represent our tables, and generated sql queries. We use a JDBC driver to easily configure our database connection.

MySQL Database Tables and Fields



One-to-Many Relationships:
User to UserLocations (unidirectional)

Many-to-Many Relationships:
Game to User (mapped by join table: Game_User)