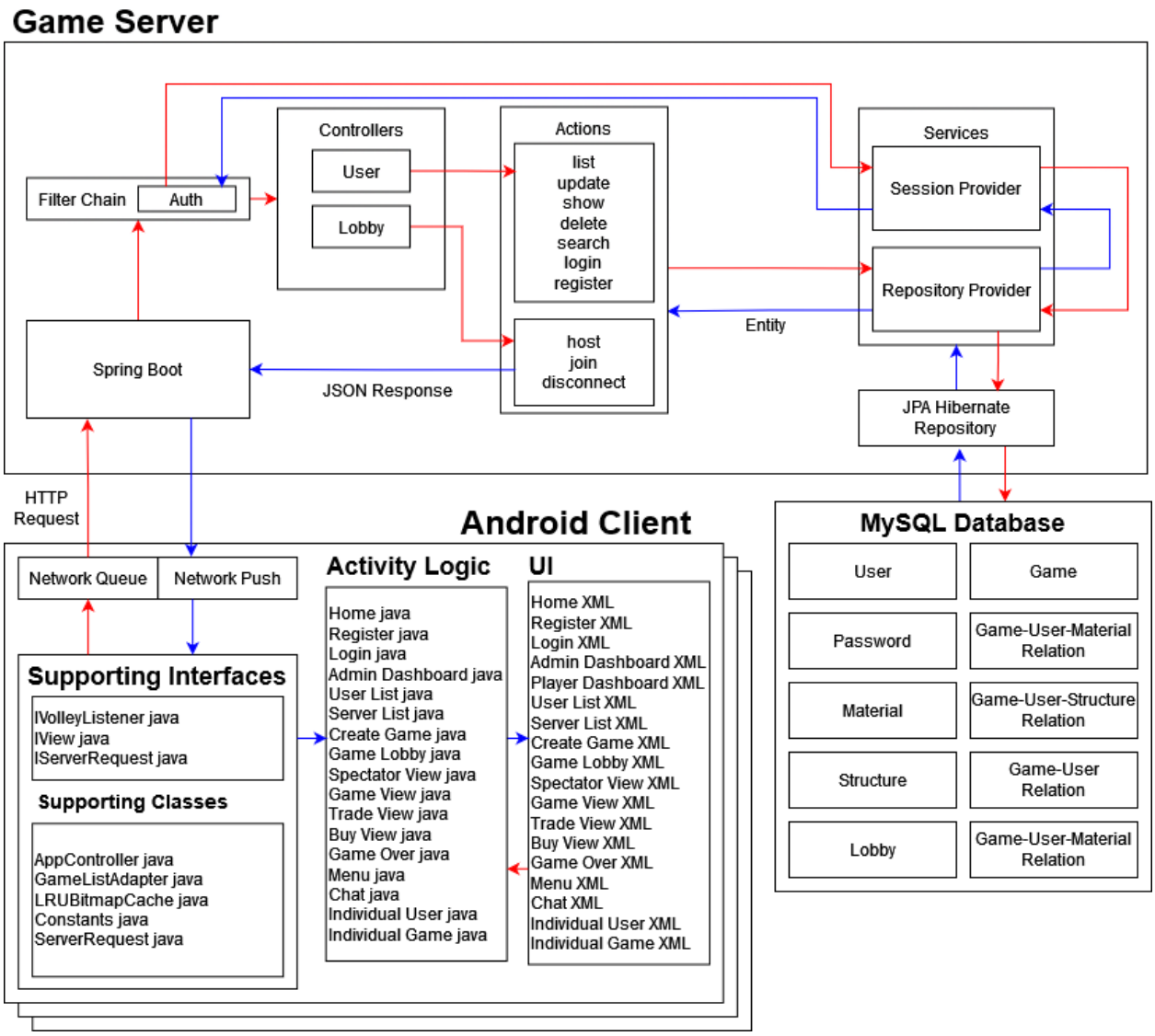
Block Diagram

1_YN_3: David Arlt, Noah Cordova, Matthew Renze, Dan Rosenhamer

King of the Rock


Game Server
Controllers
User
Lobby
Actions
list
update
show
delete
search
login
register
host
join
disconnect
Services
Session Provider
Repository Provider
Entity
Filter Chain
Auth
Spring Boot
JSON Response
JPA Hibernate Repository
HTTP Request
Android Client
Network Queue
Network Push
Supporting Interfaces
IVolleyListener java
IView java
IServerRequest java
Supporting Classes
AppController java
GameListAdapter java
LRUBitmapCache java
Constants java
ServerRequest java
Activity Logic
Home java
Register java
Login java
Admin Dashboard java
User List java
Server List java
Create Game java
Game Lobby java
Spectator View java
Game View java
Trade View java
Buy View java
Game Over java
Menu java
Chat java
Individual User java
Individual Game java
UI
Home XML
Register XML
Login XML
Admin Dashboard XML
Player Dashboard XML
User List XML
Server List XML
Create Game XML
Game Lobby XML
Spectator View XML
Game View XML
Trade View XML
Buy View XML
Game Over XML
Menu XML
Chat XML
Individual User XML
Individual Game XML
MySQL Database
User
Game
Password
Game-User-Material Relation
Material
Game-User-Structure Relation
Structure
Game-User Relation
Lobby
Game-User-Material Relation

Design Descriptions

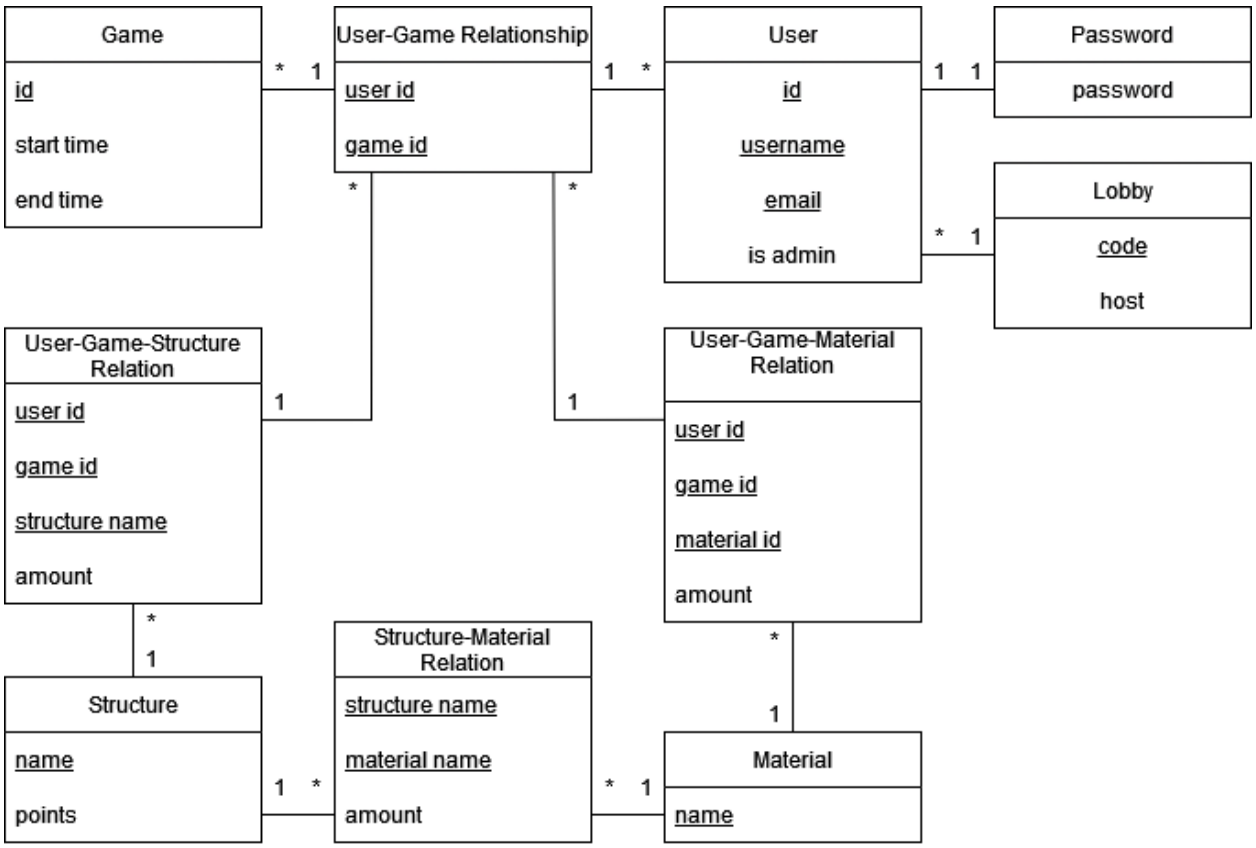
On the client side, each client contains modules categorized into View, Logic, and Connection and are supported by the GUI, activity, logic, and communications classes.

The Android View of the screens is composed of GUI xml classes and java activity classes. When the program runs, the screens are initialized and an AppController is instantiated to maintain our main thread. Any time a user presses a button, the action is added to the thread by the AppController and processed according to the Android Logic defined in the activity and logic classes. Thus, the user is directly interacting with the screen views which all implement the class IView. Upon every user action, an instance of a server request implementing IServerRequest is created and sent to the server via http and our specifications of the Android Connection. The connection is established upon every action, or opened using websockets when applicable, and server requests that are made from every action are sent to the server using http protocol. These are pushed to the network by the AppController and received by the server. For incoming communication, the same connection is used between the client side http and the server. Incoming responses, or messages via websocket, are picked up by listeners implementing IVolleyListener and processed according to the logic and activity classes.

On the server side, the database tables hold all of information relating to users, specifically their information (username/password), their login status, and the role they have in relation to the game. In addition to users the database tables also hold the information regarding two major game components, structures and materials. The last group of information that the database tables hold is the game and the game-user relationship that determines what state each game is in and what state each player of each game is in.

The data of game creation and in-game user progression is designed to flow from the http based front end into the Spring Boot server interface where it is then handled by controllers, then through actions and service requests and finally into the database. Upon reaching the database, a JSON object is created and sent back through the JPA Hibernate repository, into the repository provider service, back to actions, sent to the spring boot interface, which will send the JSON response body back to the front end.

Database Tables



Game: Information on a single instance of a Game.

User-Game Relation: Information attached to a specific User and a specific Game.

User: Information about a User, including login credentials and app roles.

Password: Passwords for Users.

Lobby: Information about the pre-game waiting area, including a join code and the hosting User.

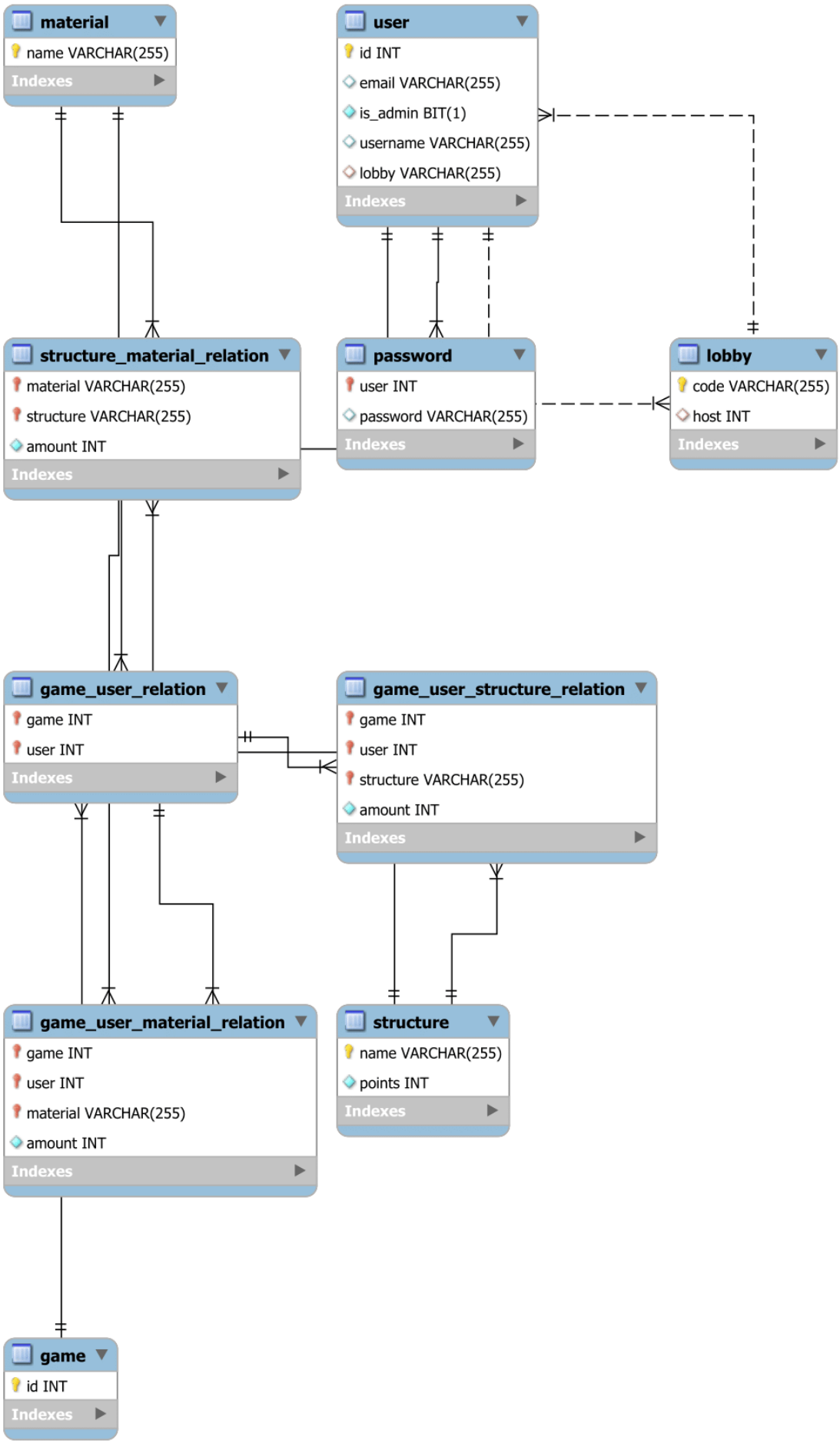
User-Game-Structure Relation: Tracks the amount of a Structure a User has in a Game.

User-Game-Material Relation: Tracks the amount of a Material a User has in a Game.

Structure: Name of a structure and its point value.

Structure-Material Relation: Defines the amount of a Material necessary to build a Structure.

Material: Name of a Material.


material
name VARCHAR(255)
Indexes
user
id INT
email VARCHAR(255)
is_admin BIT(1)
username VARCHAR(255)
lobby VARCHAR(255)
Indexes
structure_material_relation
material VARCHAR(255)
structure VARCHAR(255)
amount INT
Indexes
password
user INT
password VARCHAR(255)
Indexes
lobby
code VARCHAR(255)
host INT
Indexes
game_user_relation
game INT
user INT
Indexes
game_user_structure_relation
game INT
user INT
structure VARCHAR(255)
amount INT
Indexes
game_user_material_relation
game INT
user INT
material VARCHAR(255)
amount INT
Indexes
structure
name VARCHAR(255)
points INT
Indexes
game
id INT
Indexes