Design Document for Airarret

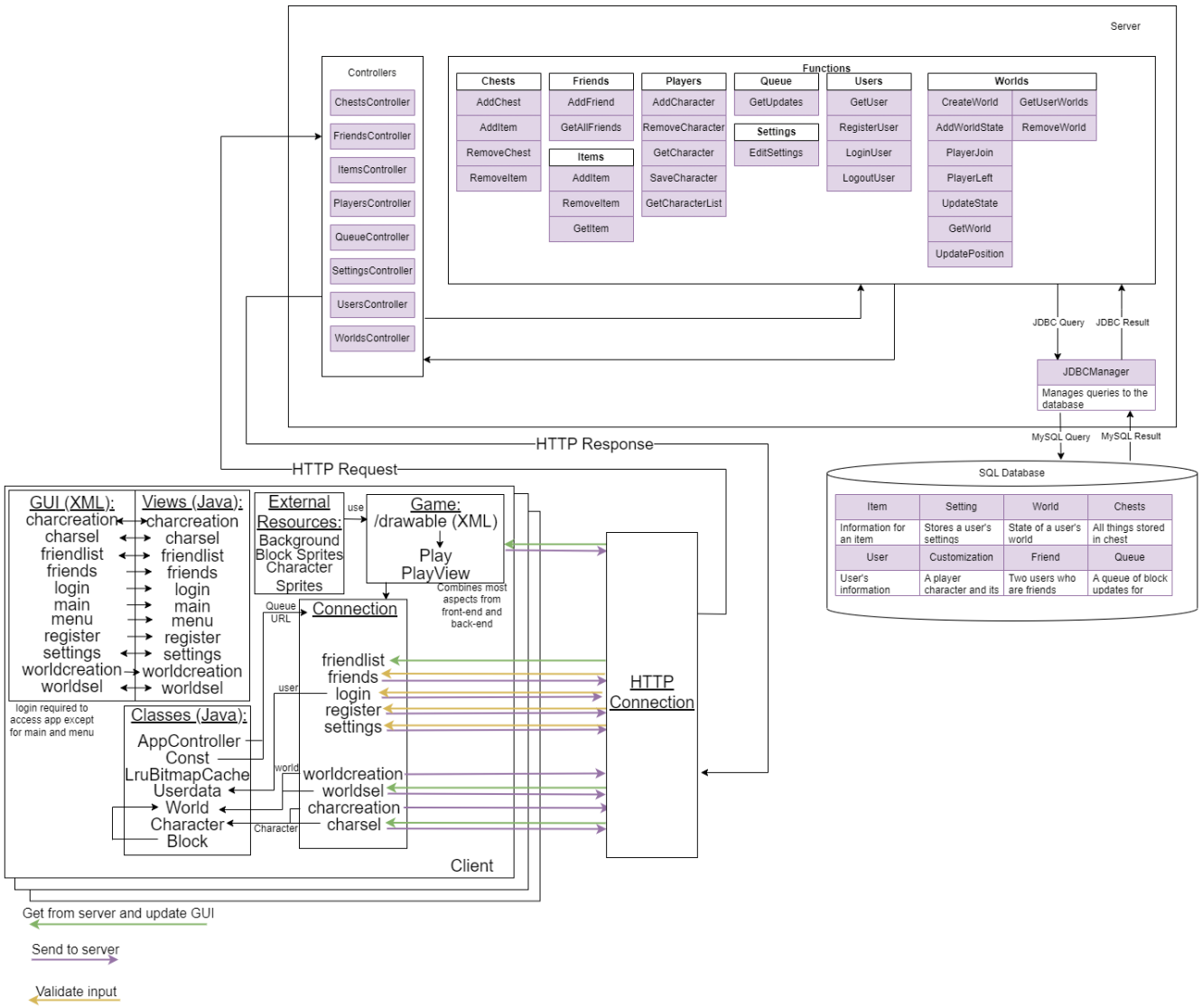
Group 2 AN 8

Adam Riffel: 25% contribution

Dillon Gesy: 25% contribution

Shishir Sharma: 25% contribution

Gabe Owen: 25% contribution


Server
Controllers
ChestsController
FriendsController
ItemsController
PlayersController
QueueController
SettingsController
UsersController
WorldsController
Functions
Chests
AddChest
AddItem
RemoveChest
RemoveItem
Friends
AddFriend
GetAllFriends
Items
AddItem
RemoveItem
GetItem
Players
AddCharacter
RemoveCharacter
GetCharacter
SaveCharacter
GetCharacterList
Queue
GetUpdates
Settings
EditSettings
Users
GetUser
RegisterUser
LoginUser
LogoutUser
Worlds
CreateWorld
GetUserWorlds
AddWorldState
RemoveWorld
PlayerJoin
PlayerLeft
UpdateState
GetWorld
UpdatePosition
JDBC Query
JDBC Result
JDBCManager
Manages queries to the database
MySQL Query
MySQL Result
SQL Database
Item
Setting
World
Chests
Information for an item
Stores a user's settings
State of a user's world
All things stored in chest
User
Customization
Friend
Queue
User's information
A player character and its
Two users who are friends
A queue of block updates for
HTTP Response
HTTP Request
GUI (XML):
Views (Java):
charcreation
charsel
friendlist
friends
login
main
menu
register
settings
worldcreation
worldsel
login required to access app except for main and menu
External Resources:
Background Block Sprites
Character Sprites
use
Game:
/drawable (XML)
Play
PlayView
Combines most aspects from front-end and back-end
Queue URL
Connection
user
world
Character
friendlist
friends
login
register
settings
worldcreation
worldsel
charcreation
charsel
HTTP Connection
Classes (Java):
AppController
Const
LruBitmapCache
Userdata
World
Character
Block
Client
Get from server and update GUI
Send to server
Validate input

Frontend:

- Character Creation
    - Gives the user 5 options each for hair, skin, shirt, pants, and shoes
    - User must name their character and select a difficulty
    - Character object is sent and saved to the server
- Character Select
    - Retrieves list of 5 characters from the user
    - Selected character's name is sent to the server to retrieve its data (appearance and difficulty)
- World Creation
    - User must name their world, select a difficulty, and select a world type.
    - Immediately sends user to world with chosen character
    - World is sent to the server
- World Select
    - Retrieves list of 5 worlds from the user along with their IDs
    - Retrieves list of 20 worlds from user's friend's active worlds along with their IDs
    - Chosen world is immediately loaded from the server to play the game
- The Game
    - Features a procedurally, randomly generated world that is stored as a 2D char array
    - Fills the screen (canvas) with 32 by 32 pixel "Blocks" holding bitmaps every frame
    - Only the blocks that would fit on the size of the screen are drawn, with the array being shifted and the new column/row of Blocks constructed every time a change in player position occurs

Backend

Communication

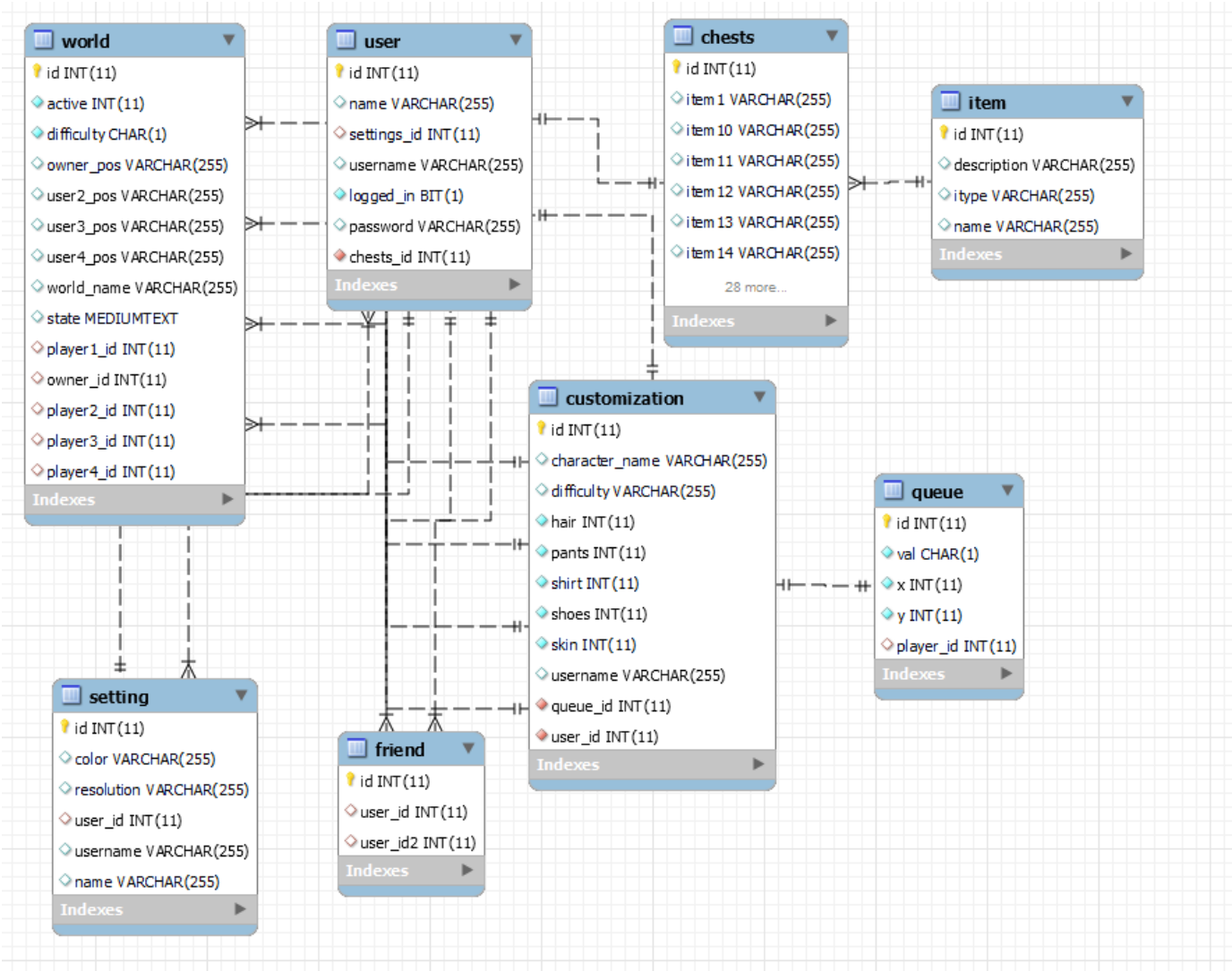
The backend connects to the frontend using Springboot's REST api to communicate with the server. Mappings are used to update the database based on the frontend requests using 3 HTTP methods:

- GET: requests information from the frontend
- POST: sends information to the database
- PUT: updates information in the database
- DELETE: deletes information in the database

Controllers

The controllers mentioned below use the HTTP mappings shown above to communicate between the frontend and the backend. Our controllers so far implemented include:

- User: multiple HTTP methods for creating a new user, logging in as an existing user, getting user information, deleting a user.
- Player: many to one relationship with user. methods for creating and customizing a new player model, editing a player model, deleting a player model and receiving any information the front end needs about the player model.
- Chest: one-to-many relationship with item. Contains methods for creating a new chest, adding an item into the chest, removing an item from the chest and deleting a chest as a whole.
- Item: Contains mappings to create items, get specific items, or delete items. Items have an id, a name, a description, and a type to designate an item.
- Setting: Contains a one-to-one relationship with User and mappings to retrieve and update information from the frontend.
- World: Contains many-to-one relationships between an owner User and several players that can join and leave a World. Contains several mappings for creating, updating, and deleting Worlds.


world
id INT(11)
active INT(11)
difficulty CHAR(1)
owner_pos VARCHAR(255)
user2_pos VARCHAR(255)
user3_pos VARCHAR(255)
user4_pos VARCHAR(255)
world_name VARCHAR(255)
state MEDIUMTEXT
player1_id INT(11)
owner_id INT(11)
player2_id INT(11)
player3_id INT(11)
player4_id INT(11)
Indexes
user
id INT(11)
name VARCHAR(255)
settings_id INT(11)
username VARCHAR(255)
logged_in BIT(1)
password VARCHAR(255)
chests_id INT(11)
Indexes
chests
id INT(11)
item 1 VARCHAR(255)
item 10 VARCHAR(255)
item 11 VARCHAR(255)
item 12 VARCHAR(255)
item 13 VARCHAR(255)
item 14 VARCHAR(255)
28 more...
Indexes
item
id INT(11)
description VARCHAR(255)
itype VARCHAR(255)
name VARCHAR(255)
Indexes
customization
id INT(11)
character_name VARCHAR(255)
difficulty VARCHAR(255)
hair INT(11)
pants INT(11)
shirt INT(11)
shoes INT(11)
skin INT(11)
username VARCHAR(255)
queue_id INT(11)
user_id INT(11)
Indexes
queue
id INT(11)
val CHAR(1)
x INT(11)
y INT(11)
player_id INT(11)
Indexes
setting
id INT(11)
color VARCHAR(255)
resolution VARCHAR(255)
user_id INT(11)
username VARCHAR(255)
name VARCHAR(255)
Indexes
friend
id INT(11)
user_id INT(11)
user_id2 INT(11)
Indexes