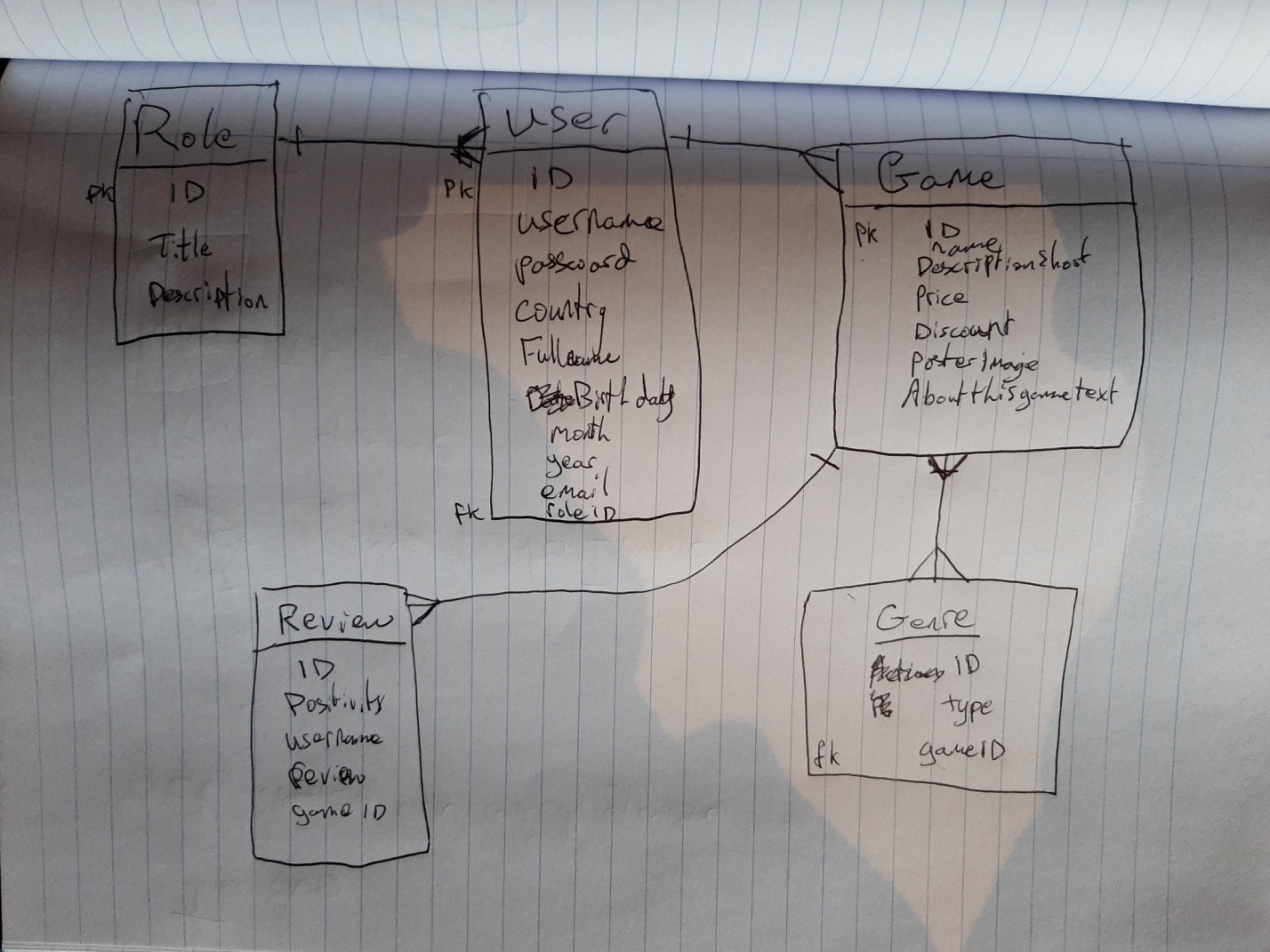
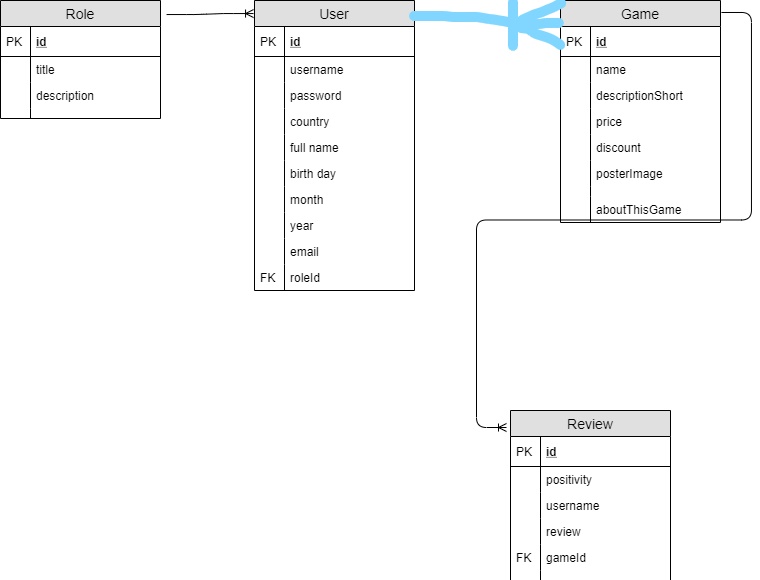
# ERD -> structure

This was the ERD I had at the start but wasn’t able to implement it all. The second is the one that relates to my database.





My tables are Role, User, Game, Review and Genre.

Role has id(pk), title and description. User has id(pk), username, password, country, full name, birth day, month, year, email, role id(fk). A role is given to a user, eg manager or user or admin. One user can have one role. One role can have many users.

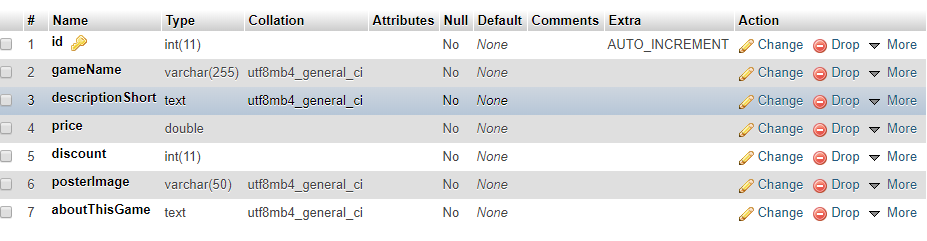
Game has id(pk), name, descriptionShort, price, discount, posterImage, and aboutThisGame. Review has id(pk), positivity, username, review, gameId(fk). The review foreign key relates to the game id. One game can have many reviews. One review can only have one game.

One game can have many genres and one genre can have many games.(not implemented)

# phpmyadmin table structure

Games table

id for the primary key (int, auto increment so its unique)



gameName to store the name of any game.

descriptionShort to store any short relevant info about the game.

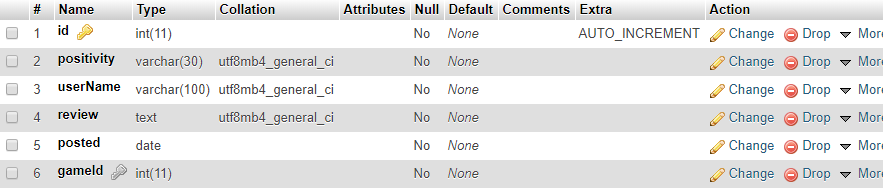
price to store the price as a double so it shows the .00 after the number.

discount to store the discount %.

poster image to store the actual image that will be displayed.

aboutThisGame stores the text for the description.

Reviews Table.



Id for the primary key for unique identifier.

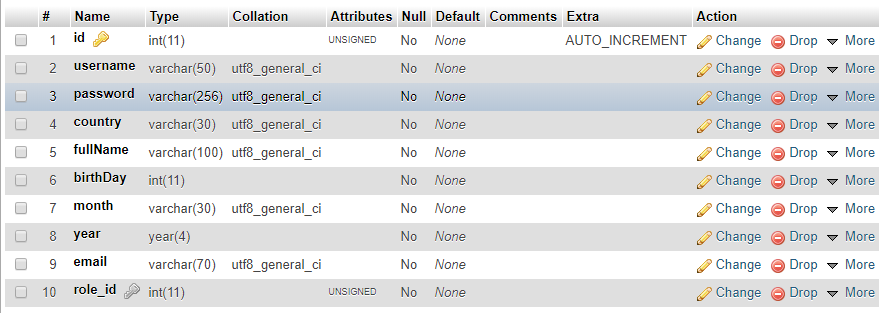
positivity was originally a enum but I couldn’t get that to work with php so I just changed it to a varchar and had a drop down list for the values it could be.

userName to store the username of the person who wrote the review.

posted is to store the date the review was posted.

gameId to store the game id of the game the review is related to. (fk)

Users table



id for pk.

username to store the username.

password to store the users password.

country to store their country.

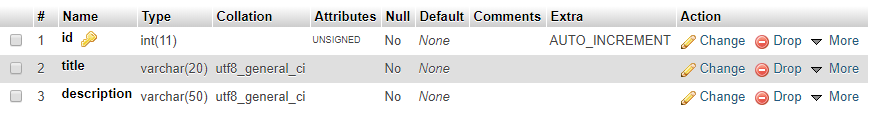
fullName to store their full name

birthday, month, and year to get their DOB.

email to store their email.

role\_id to store the role that this user relates to.

Roles Table

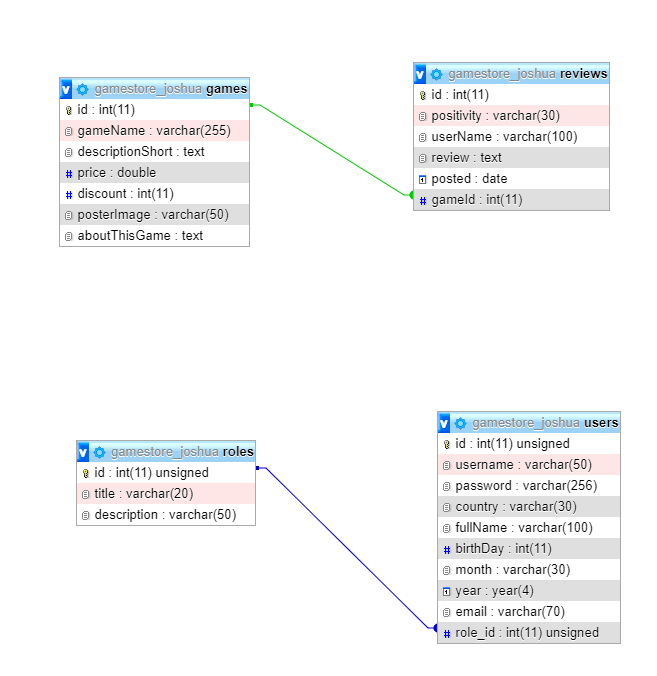


id to store primary key.

title to store the title they have.

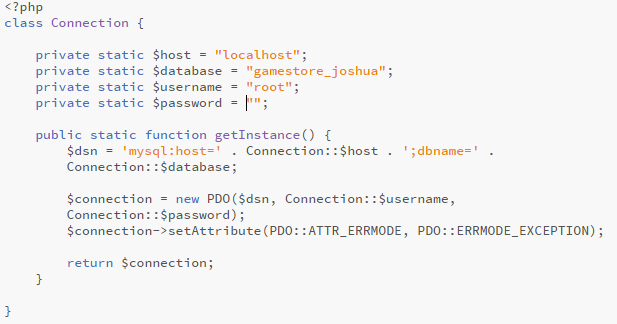
description to explain what the title can do.

# Designer tab

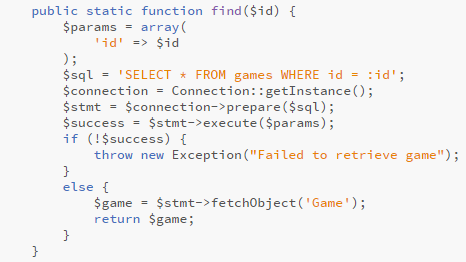


# Explanation of code

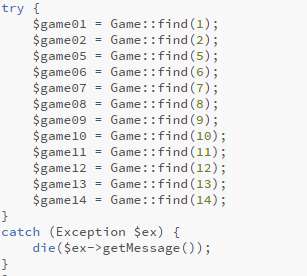
• DB connection using PHP and describe the following operations:

Using the connection class which has the database name the host name the username and the password then uses the getinstance() function to connect.

* A SELECT query with at least one WHERE condition

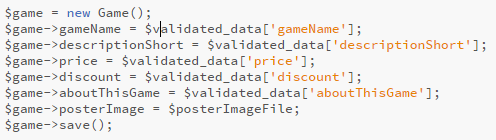


With this select I can get all the data from a specific game. In my index.php I use this to get data about specific games by using their game id as shown in the picture below. The find function takes in a parameter (the id) and uses the select \* from games where id = :id <- being the id that was taken in. if it succeeds then it fetches the object and all the info from the database and returns that data to $game.



* An INSERT command

The games\_create.php form takes in user input to create a game, then sends the data to the games\_store.php where it is validated and inserted into the database



This is creating a new game and filling the data fields with the validated data. The save() function at the end puts the values into an array with database values = the inputted values. If there is no id assigned to the object it’s creating then it lets the primary key auto increment to whatever id it should be and it inserts all the values using INSERT INTO games() values().



* An UPDATE command with at least one WHERE condition

There is an update in the above picture. The edit form sends the data(including the id) to the update form on submit it goes through the update validation and then it is saved (save()) the save function skips the first if because the id is not null and uses the update games sql which updates the id of whatever id was passed in.