Milestone 2 Report

Daniel Petrov, Kyrylo Shevchenko

How to run the project

TO RUN THE PROJECT ONE MUST NAVIGATE TO THE ROOT DIRECTORY OF THE PROJECT. THERE SHOULD BE A ‘DOCKER-COMPOSE.YML’ FILE AND AFTER RUNNING THE COMMAND ‘DOCKER COMPOSE UP’ A CONTAINER IS STARTED. ALL THE BUILD WORK IS DONE AUTOMATICALLY BY DOCKER AND ONLY THE NECESSARY PORTS ARE BEING EXPOSED.

Tech Stack

The application itself contains 2 parts - backend and frontend.

For the backend part we used Java, together with the Java Spring Boot framework.

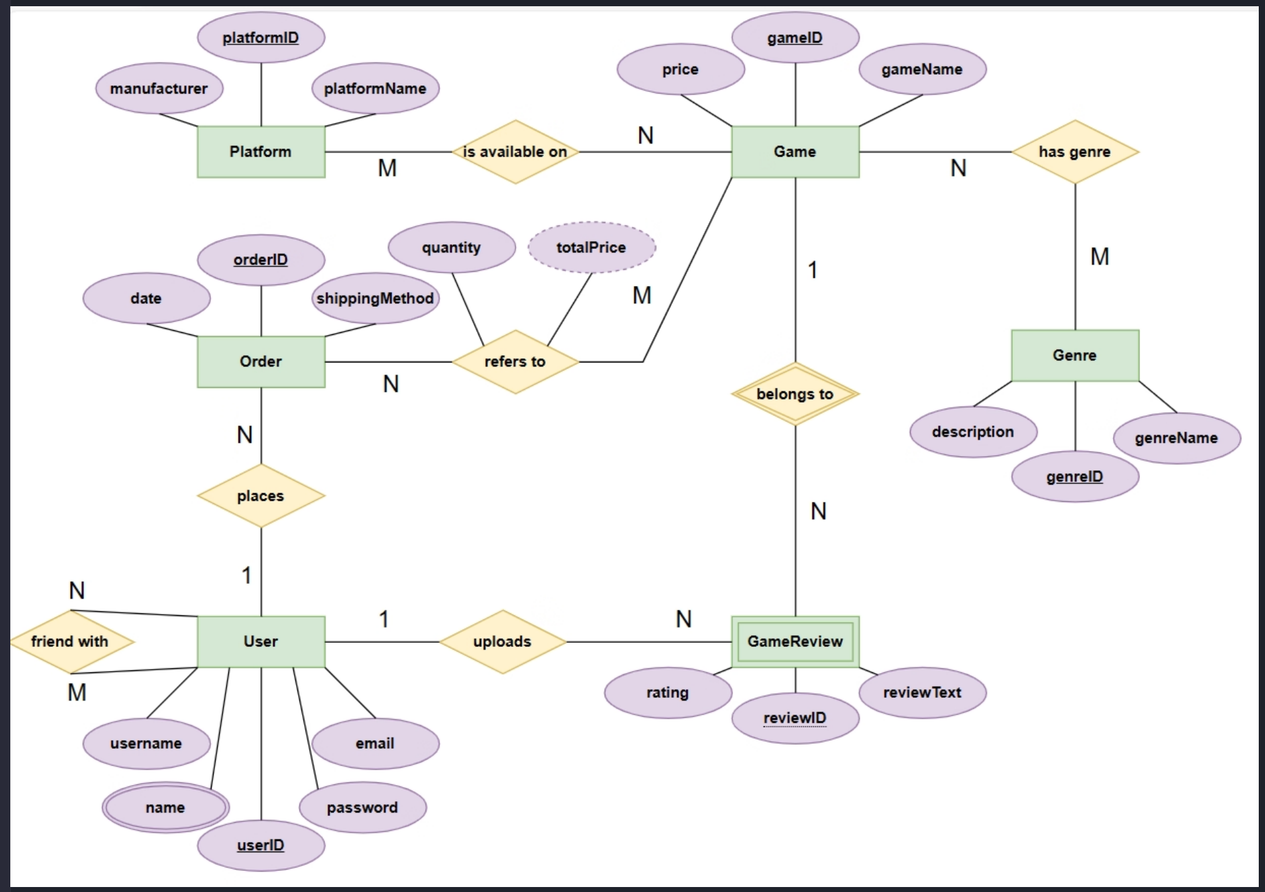
Frontend was built using a simple HTML/CSS/JS stack with bootstrap for the design and jquery library to fetch the api endpoints and send requests to the backend.

For the secure access (HTTPS) on the frontend we used a self-signed certificate generated with the help of keytool - a utility tool provided by Java for managing keys and storing them in a keystore.

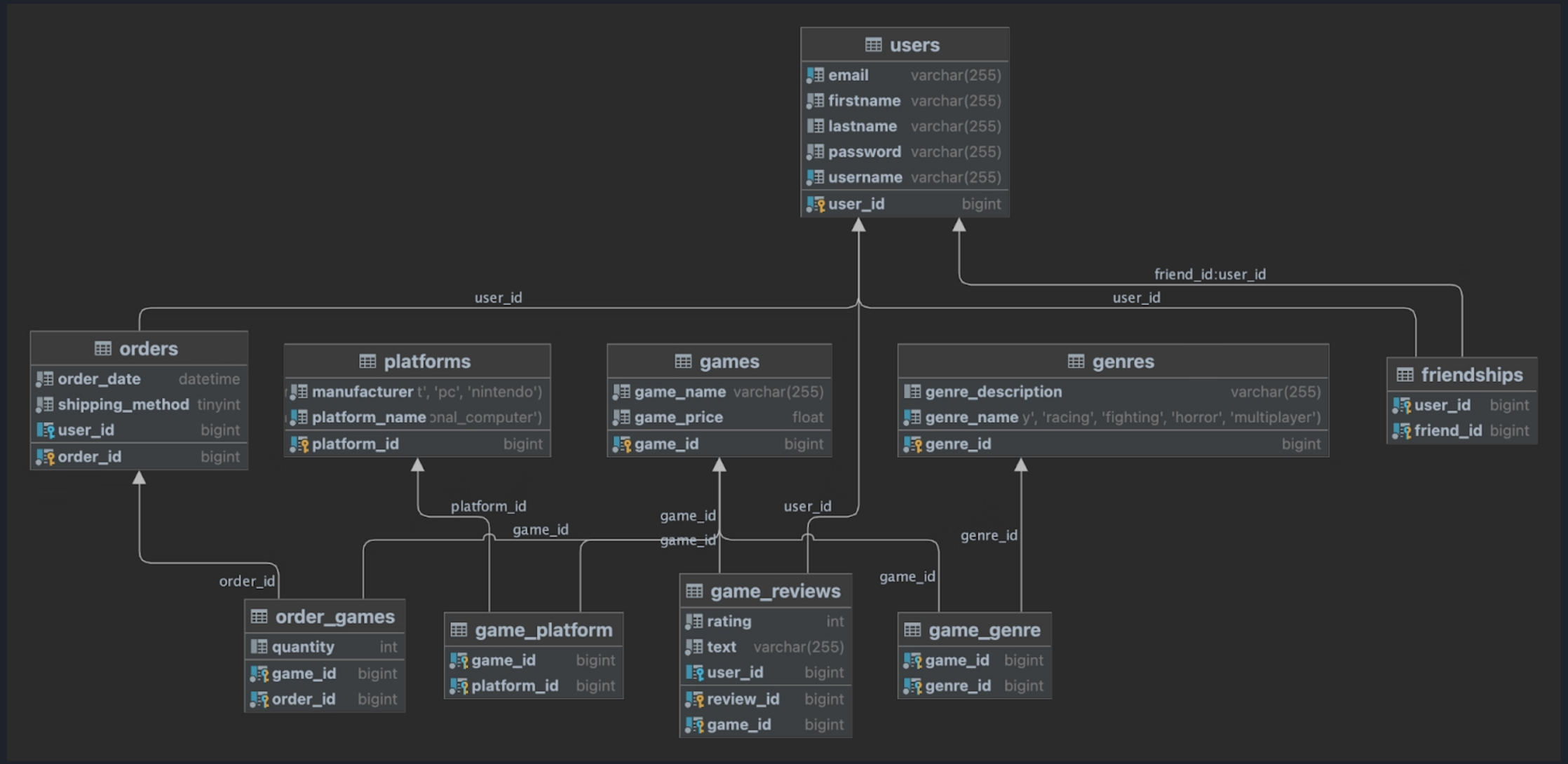
Everything regarding build-pipeline was implemented using docker and docker-compose files.

Changes made from Milestone 1

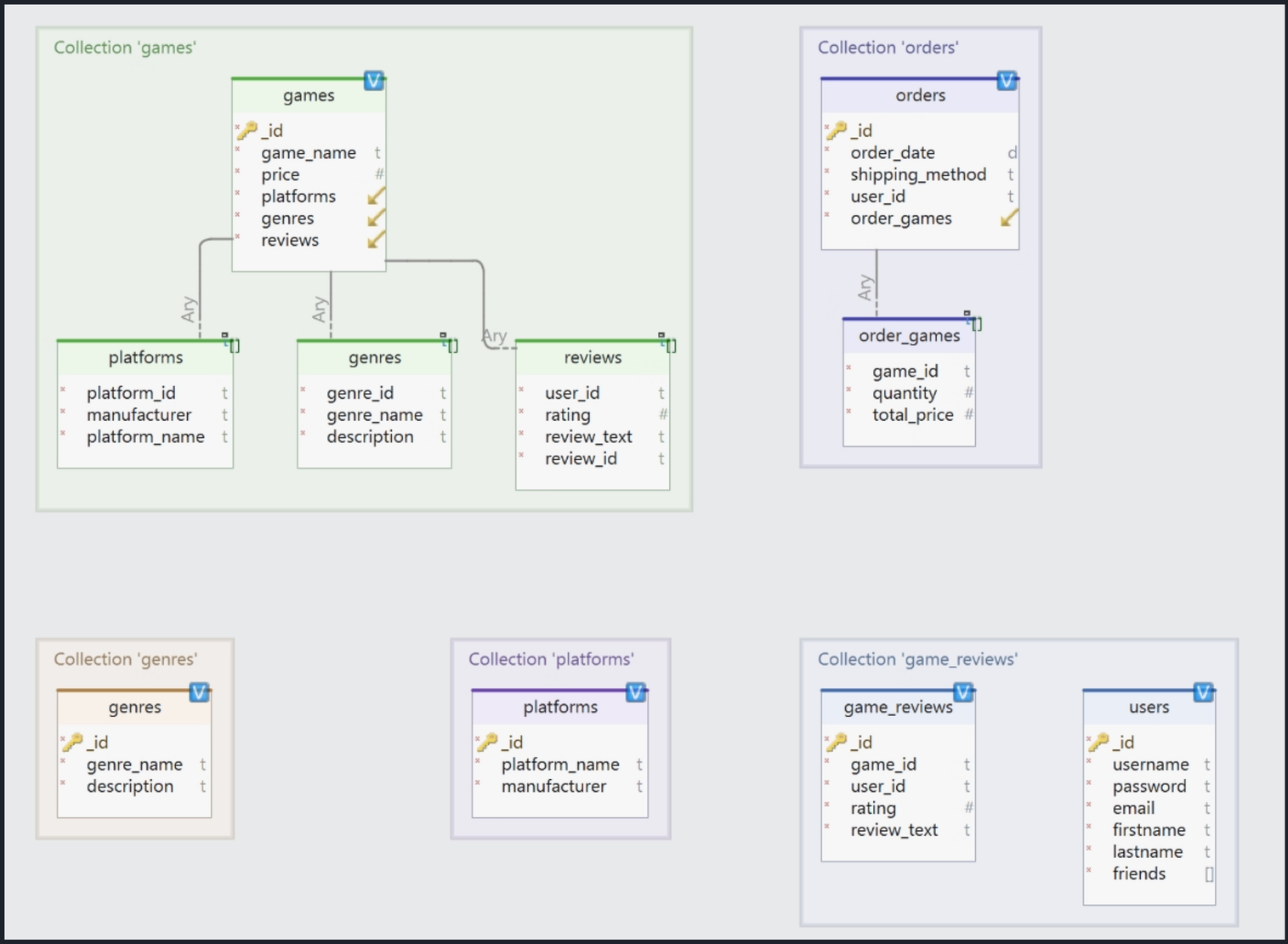
- The GameDetails entity was removed
- Replaced the rating attribute in Genre with a description.
- The rating of games in a GameReview object now ranges from 1-10
- Release_year, description were removed from Game - developer



Physical Design



NoSQL Design



Conclusion

A LOT OF TABLE DATA IS ALREADY EMBEDDED IN MONGODB, FOR EXAMPLE USER ENTITY. RELATIONSHIPS (LIKE FRIENDS, REVIEWS, ORDERS) ARE TYPICALLY STORED IN SEPARATE TABLES AND LINKED USING FOREIGN KEYS. THIS NORMALIZES THE DATA BUT REQUIRES JOIN OPERATIONS TO AGGREGATE DATA FROM MULTIPLE TABLES. SAME GOES FOR THE ORDER TABLE, MONGODB DOCUMENT SIMPLIFIES THE READABILITY OF THE DATA AND IMPROVES THE EFFICIENCY OF THE DATABASE.

REGARDING “PURCHASING A GAME” , IN MONGODB YOU CAN SEE A REDUCED NUMBER OF QUERIES TO PERFORM THE OPERATION : 1 INSTEAD OF 2. WE USED 2 SQL INSERTIONS FOR IT.
WHEN YOU FETCH A DOCUMENT, ALL THE EMBEDDED DATA IS RETRIEVED IN A SINGLE I/O OPERATION. THIS IS PARTICULARLY BENEFICIAL FOR READ-HEAVY APPLICATIONS WHERE YOU OFTEN NEED TO ACCESS RELATED DATA TOGETHER.

Explanation of Design Choices

1. EMBEDDING VS. REFERENCING:

Reviews are embedded within games

Orders include references to games rather than embedding game details directly

2. INDEXING:

Indexes are chosen based on query patterns.

3. USERID IN GAME REVIEWS:

Including `userId` in each review within the games collection allows for directly associating reviews with users.

4. COMPOSITE KEY IN GAMEREVIEW:

In MongoDB, each document must have a unique `_id`. While we use `_id` as the primary key (default in MongoDB), it's also possible to create a composite key using `review_id` and `game_id` for relational consistency. However, MongoDB's document model typically relies on the unique `_id` field for simplicity and efficiency.

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)