My project

1. Distributed Bomberman Game Management System
1.1. Project description and requirements
This project has two main objectives: firstly, to enrich the students' knowledge and familiarity with the models and algorithms used to manage distributed systems; and secondly, to deepen the students' knowledge and practice of some of the tools used to develop and manage distributed systems.
In particular, this project proposes the development of a system to manage the creation and execution of various distributed multiplayer games similar to the traditional Bomberman.
Bomberman is a game that was released by Hudson Soft in 1983 in which players control a character (Bomberman) who strategically places bombs throughout a maze to destroy obstacles and enemies. The main objective involves eliminating all the enemies or finding the way out of the maze, while avoiding being hit by the bombs themselves or by the explosions of bombs placed by other players or the computer.
An example of the GUI adapted for Java can be seen in Figure 1.
Figure 1 – Example of a screenshot of Bomberman.

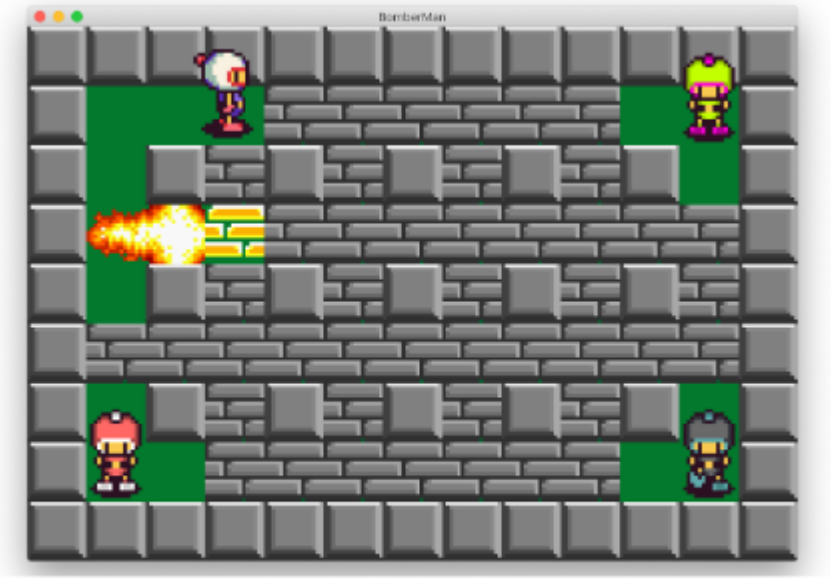


Figure 1 – Example of a *screenshot* of Bomberman.

The proposed distributed system should serve as a broker for managing and synchronizing
several multiplayer Bomberman games running simultaneously, according to the requirements listed below:
[R1] Each player must use a local GUI to contact a remote GameFactory in order to register with a username and password. They will then use the same service to login to the system with the credentials defined in the registration. They must use JWT for authentication;

[R2] After players have logged in, they access the system's GameSession. In this area they can: o Get/view all BombermanGame games ever created;
o Get/view the BombermanGame games that are already running;
o Get/view a list of references for all BombermanGame games created that are still accepting players (i.e. that are not yet running);
o Get the reference (proxy) for a given BombermanGame that accepts players; o Create new BombermanGame games. Each game created must have a unique identifier. When creating a new game, it should be possible to set the number of players (2 to 4); [R3] Through the proxy for an already created BombermanGame, the player can perform a set of actions:
o Each player can join (attach) only one BombermanGame at a time. Each game must require the presence of n players (clients) to start (2 to 4 players);
o When a player joins a BombermanGame, a local instance of the game (GUI) should be created so that this player can play and view the actions of the other players; o After joining a game, each player's actions (e.g., position, movement, score, etc.) must be synchronized with the BombermanGame instance on the server, which in turn synchronizes with the various players registered in that game. An implementation must be created using the Observer standard with RMI and another implementation using the Publish/Subscribe standard with RabbitMQ. NB: implementation in both technologies is mandatory;
[R4] The system should include at least 3 server instances to ensure fault tolerance and load distribution. The information on registered and connected users, as well as the status of the different BombermanGames, must be replicated and synchronized between the servers. The system should automatically manage the persistence of the BombermanGames status. They could create a Front Server that re-routes players (in Round Robin) to one of the 3 backend services. If a server fails then the players connected to that server must use another server/replica of the game in order to continue playing. A server that recovers its operation after a crash must be able to synchronize its status from the local information and that of the other servers. [R5] The system should have its architecture described using UML diagrams: o Class diagrams identifying the remote interfaces and their implementation classes; o Message sequence diagrams for system usage scenarios.
As described, the main objective will be to develop the game in a distributed way. The system should facilitate the organization, coordination and persistence of data relating to the various groups of players who have competitive access to the same resources. Figure 2 illustrates a simplified schematic of the distributed system instances.

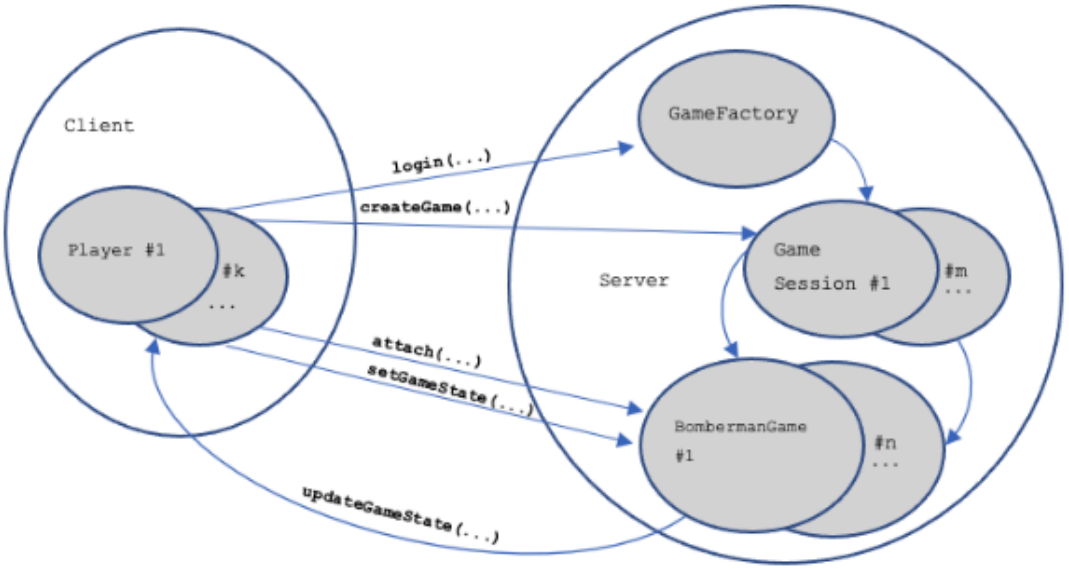


Figure 2 - Simplified diagram of distributed system instances.

1.2. Tecnology

There are various design patterns that you should use to organize the structure and behaviour of the distributed application you are requesting. For example, the observer design pattern is widely used to synchronize several observers subscribing to the same subject. Alternatively, the publish/subscribe pattern allows several producers to communicate asynchronously with several consumers via Message Queues. The factory method design pattern is also a suitable solution for creating sessions according to user profiles. Another very useful pattern is the visitor design pattern, which allows you to remotely send/execute operations on a predefined data structure (e.g. a file system tree). There are many other patterns that can be used or combined in the creation, organization and implementation of the system.

1.3. Explanation of the details of the Bomberman game made available

In the Bomberman game available, the player (client) controls a character with the aim of defeating the enemies. To do this, you have to place bombs along the way to destroy walls and eliminate enemies, while avoiding your own bombs and those of your opponents.

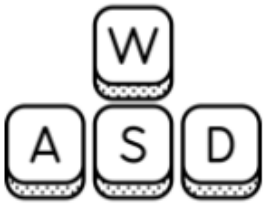
The steps to run the game provided are as follows:

1. Create a new project based on the code provided in Canvas 1
2. Open a Terminal and enter the "src" folder: cd src
3. Compile the game via terminal: javac *.java
4. Run the Server class: java Server
5. Open a new Terminal instance
6. Run the Client class for each player you want to join the game:
java Client
7. Move your character with the keys W, S, D, A
8. Place bombs with the key Space

9. Beat your opponents

Keys

To play, use the W, S, D, A keys to move and the Space key to place bombs on the map.



Movimentar

Colocar bombas

Important note:
You will have to adapt the game so that the requirements can be implemented. It is recommended that they create a new package within the class project and change the path for the images (absolute path) so that the game starts without problems..