
UNIVERSITÀ
DI PARMA

# FINAL RELATION

# INTERNET TECHNOLOGIES


Michael Bandini

## Summary

# Introduction

Guess Word is a game where the goal is to guess a word starting from its definition and its initial. The more time passes, the more letters of the word will be discovered, making guessing easier. **GuessWord** can be played in both single and multiplayer. The object of the game is to have as many points as possible at the end of all rounds.


FIGURE 1 DEFINITION AND WORD

# Game mode

In **GuessWord** there are two game modes called *Climb* and *Battle.*

## Climbing

In climbing mode, the number of rounds and their duration are decided during the preparation of the game. For each word guessed the player earns one point. You can move on to the next word only if you have guessed the previous one. When the round changes, the mysterious words are changed, so as to allow a player stuck on a word from the previous round to "break free" from it.

At the end of all rounds, the player who guessed the most words wins. Draw is allowed.

## Battle In

*Battle* mode , you set the number of rounds, the maximum time to guess a word, and the number of words to guess per round. The duration of the round is therefore given by the time of the single word multiplied by the number of words. All players see the same word definition pair at the same time. When guessing, the score will be given by the formula:

$$= \qquad / 100$$

Where *wordTimeLeft* is the time left in milliseconds to guess the word.

If a player fails to guess the word after time runs out, he will receive zero points.

How to start a game No membership
is required to play **GuessWord .** Just select the avatar you prefer, enter a name and create a new game. If, on the other hand, you want to participate in a game created by another player, you will need to enter the code of the room associated with it.

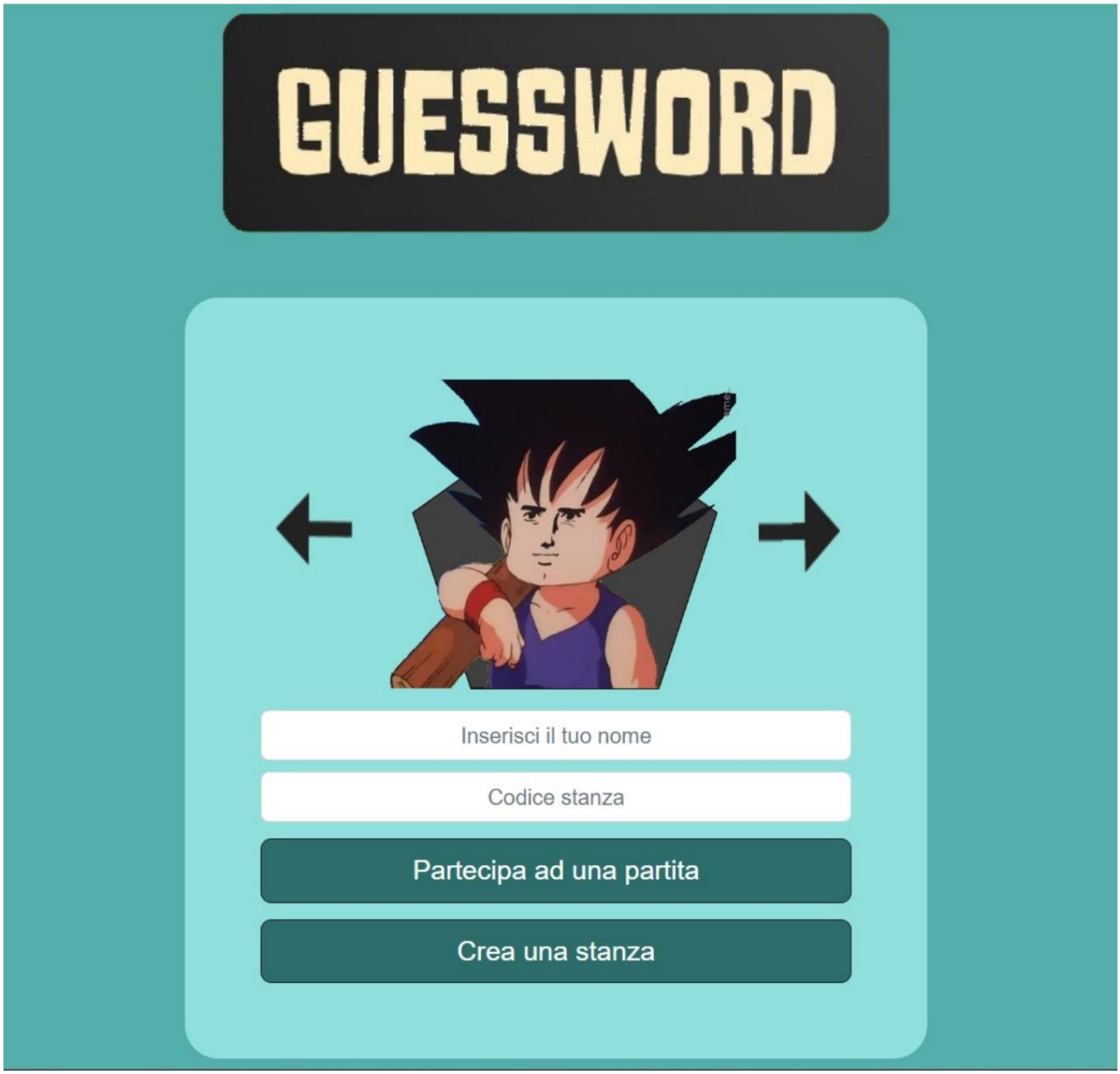


FIGURE 2 GUESSWORD HOME PAGE

In case we had pressed the *Create a room* button , the game configuration screen will appear, in which it is possible to select the game mode, set the number of rounds and their duration. Only the creator of the room will be able to see the following screen.


FIGURE 3 MATCH SETTINGS

At this point it will be possible to start a solo game or, by sharing the room code with other players, it will be possible to start a Multiplayer game. Only the creator of the room will be able to start the game.

For participants the screen will be as follows

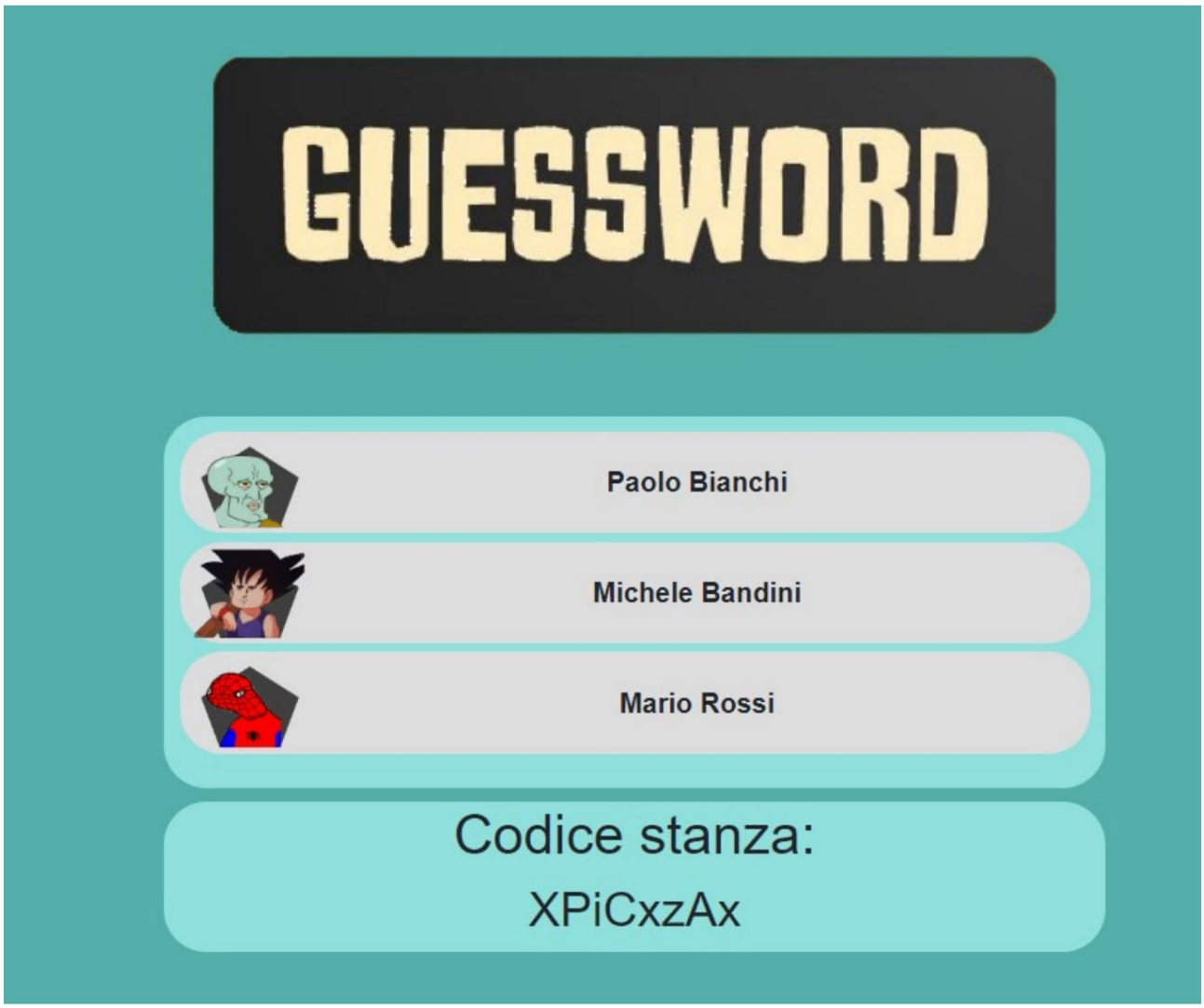


FIGURE 4 LOBBY SEEN BY PARTICIPANTS

## Game screen

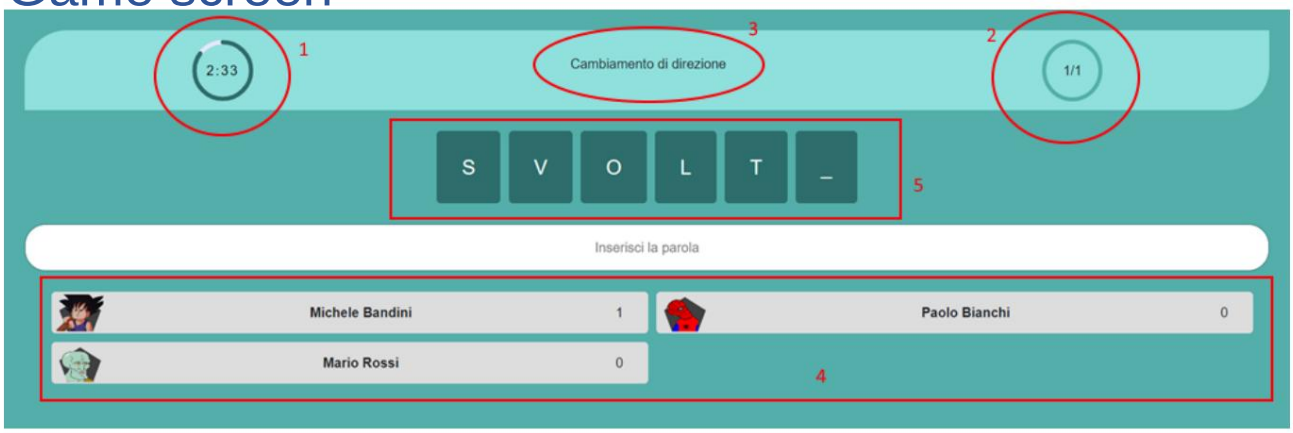


FIGURE 5 GAME SCREEN

1) In climb mode indicates the time remaining at the end of the round while in climb mode battle indicates the time left to guess the word
2) Current round number on the left and total round number on the right
3) Definition of the word to guess

4) Real-time leaderboard
5) Word to guess

To enter the word, just write it in the text box and press enter. If the word has been guessed it will be colored green.

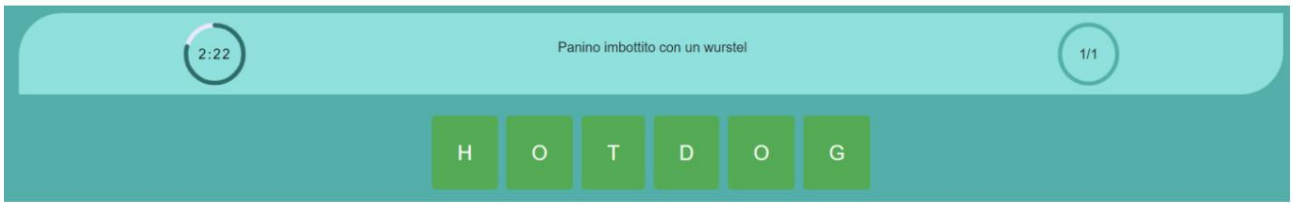


FIGURE 6 CORRECT ANSWER

If, on the other hand, an incorrect answer has been given, it will be colored red.


FIGURE 7 WRONG ANSWER

P2P communication To allow

multiplayer games, GuessWord uses the P2P architecture with a server that acts as a connection broker. No data exchanged between peers passes through the server.

Peers exchange information through packets. Each packet contains all the information needed to perform the operations associated with a given message. The packet is called "peerPacket" and has the following structure:

```
var peerPacket =
                    { 'isHost',
                    'id',
                    'idGame',
                    'idMessage',
                    'conn',
                    'guessList',
                    'name',
                    'avatarImage',
                    'score',
                    'gameMode',
                    'nTotRound',
                    'wordPerRound ',
                    'roundTime',
                    'wordMaxTime'};
```

The "IdMessage" field is a unique identifier present in each packet sent between peers. This field tells the receiving peer what information it should use to perform the function

associated with that ID. For example, if the message ID is set to "3", the peer knows that it should use the contents of the packet to assign the *peerPacket.score* field to the peer of id *peerPacket.id.*

In this way, packets sent between peers are highly efficient and contain only the information needed to perform the function associated with a given message. This makes the Guessword game fast and responsive, allowing players to interact in real time.

Each peer saves all the information related to the other peers through the *connectedPeerPacket* dictionary which has the ID of the connected peer as its key and a complete peerPacket as its value.

The connection with another peer is saved in the *connectedPeerPacket[id].conn* field and with this object it is possible to send messages through the *conn.send() method.*

## Used technologies

### Backend

Node.js was used on the backend side, which performs web server functions on port 3000 and takes care of instantiating the various peers by assigning them an ID on port 9000.

I used Node.js because it's easy to use, offers a wealth of out-of-the-box resources, and has a strong developer community offering support and troubleshooting resources.
Thanks to NPM it was really easy to integrate external APIs such as, for example, PeerJS which I used for P2P communication.

As for the storage of the definitions and the respective words, I decided to use a JSON file instead of SQLite as I had initially thought, this because I encountered problems using the SQLite libraries with ReactJS and I wanted to speed up development times. While SQLite would have been a more complex and powerful solution, I would have had to spend a lot of time troubleshooting compatibility issues and integrating libraries.

Instead, using a JSON file, I could easily manage the data quickly and easily, without having to worry about compatibility and configuration issues. Also, I was able to use libraries that were already available and easily integrated with my code, which speeded up the development process.

Although using a JSON file is less powerful than SQLite, I decided to opt for this solution because I didn't encounter any performance issues when using the data and I was able to develop the project faster and easier.

### Frontend

Frontend side I used HTML, CSS and JavaScript technologies. I leveraged HTML to structure the content and provide a solid foundation for the visual presentation of information. I then used CSS to give it an appealing aesthetic and make it more pleasing to the eye.

Regarding game logic and information exchange between peers, I used JavaScript with the React JS framework for creating the user interface, as it offered me a set of ready-to-use tools and components, making it more efficient and fast development, moreover I was able to define independent components that I was able to reuse in different parts of the code.