UNIVERSITÀ
DI PARMA

RELAZIONE FINALE
TECNOLOGIE INTERNET

Michele Bandini

Sommario

Introduzione 3
Modalità di gioco 3
Scalata 3
Battaglia 3
Come avviare una partita 4
Schermata di gioco 6
Comunicazione P2P 7
Tecnologie utilizzate 8
Backend 8
Frontend 8

Figura 1 definizione e parola 3
Figura 2 GuessWord Home page 4
Figura 3 Impostazioni partita 5
Figura 4 Lobby vista dai partecipanti 6
Figura 5 Schermata di gioco 6
Figura 6 Risposta esatta 7
Figura 7 Risposta errata 7

Introduzione

Guess Word è un gioco dove l'obbiettivo è quello di indovinare una parola a partire dalla sua definizione e dalla sua iniziale. Più il tempo passa, più lettere della parola verranno scoperte, rendendo più facile l'indovinamento. **GuessWord** può essere giocato sia in singolo che in multigiocatore. L'obbiettivo del gioco è avere più punti possibili al termine di tutti i round.

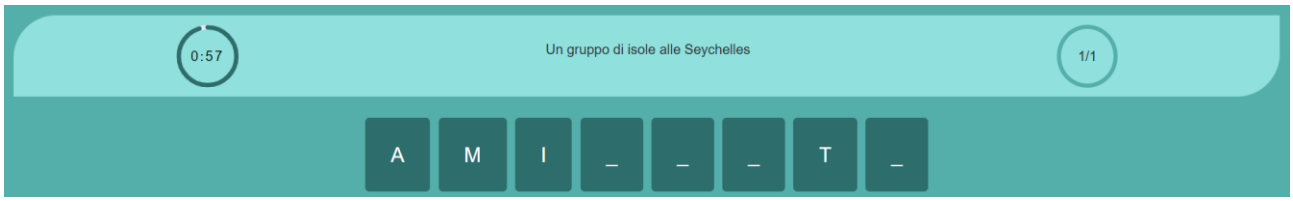

FIGURA 1 DEFINIZIONE E PAROLA

Modalità di gioco

In **GuessWord** sono presenti due modalità di gioco chiamate *Scalata* e *Battaglia*.

Scalata

Nella modalità scalata viene deciso in fase di preparazione della partita un numero di round e la durata di questi. Per ogni parola indovinata il giocatore guadagna un punto. Si può passare alla parola successiva solo se si ha indovinato quella precedente. Al cambio round vengono cambiate le parole misteriose, così da permettere a un giocatore bloccato su una parola del round precedente di "svincolarsi" da essa.

Al termine di tutti i round avrà vinto il giocatore che ha indovinato più parole. È ammesso il pareggio.

Battaglia

Nella modalità *Battaglia* viene prestabilito il numero di round, il tempo massimo per indovinare una parola e il numero di parole da indovinare per round. La durata del round è quindi data dal tempo della singola parola moltiplicato per il numero delle parole. Tutti i giocatori vedono contemporaneamente la stessa coppia definizione parola. Quando si indovina, il punteggio sarà dato dalla formula:

$$int\ Punteggio\ =\ wordTimeLeft\ /\ 100$$

Dove *wordTimeLeft* è il tempo rimasto in millisecondi per indovinare la parola.

Se un giocatore allo scadere del tempo non è riuscito a indovinare la parola riceverà zero punti.

Come avviare una partita

Per giocare a **GuessWord** non è necessaria nessun tipo di iscrizione. Basta selezionare l'avatar che si preferisce, inserire un nome e creare una nuova partita. Se invece si vuole partecipare ad una partita creata da un altro giocatore sarà necessario inserire il codice della stanza associata ad essa.

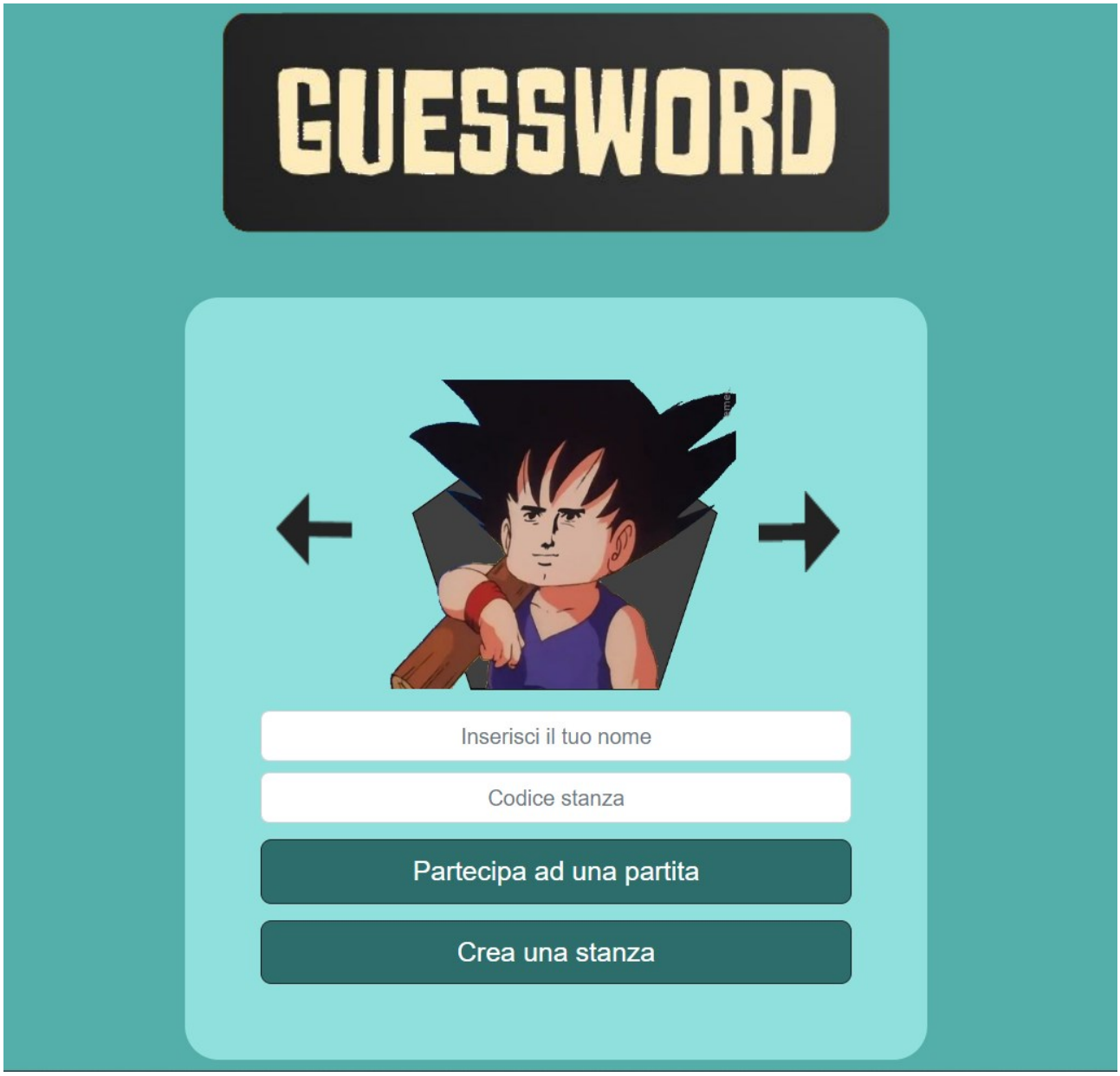

FIGURA 2 GUESSWORD HOME PAGE

Nel caso in cui avessimo premuto il bottone *Crea una stanza* ci si presentarà la schermata di configurazione della partita, in questa è possibile selezionare la modalità di gioco, impostare il numero di round e la loro durata. Solo chi ha creato la stanza potrà vedere la schermata seguente.

FIGURA 3 IMPOSTAZIONI PARTITA

A questo punto sarà possibile avviare una partita in solitaria o, condividendo il codice della stanza con altri giocatori, sarà possibile iniziare una partita in Multiplayer. Solo chi ha creato la stanza potrà avviare la partita.

Per i partecipanti la schermata sarà la seguente

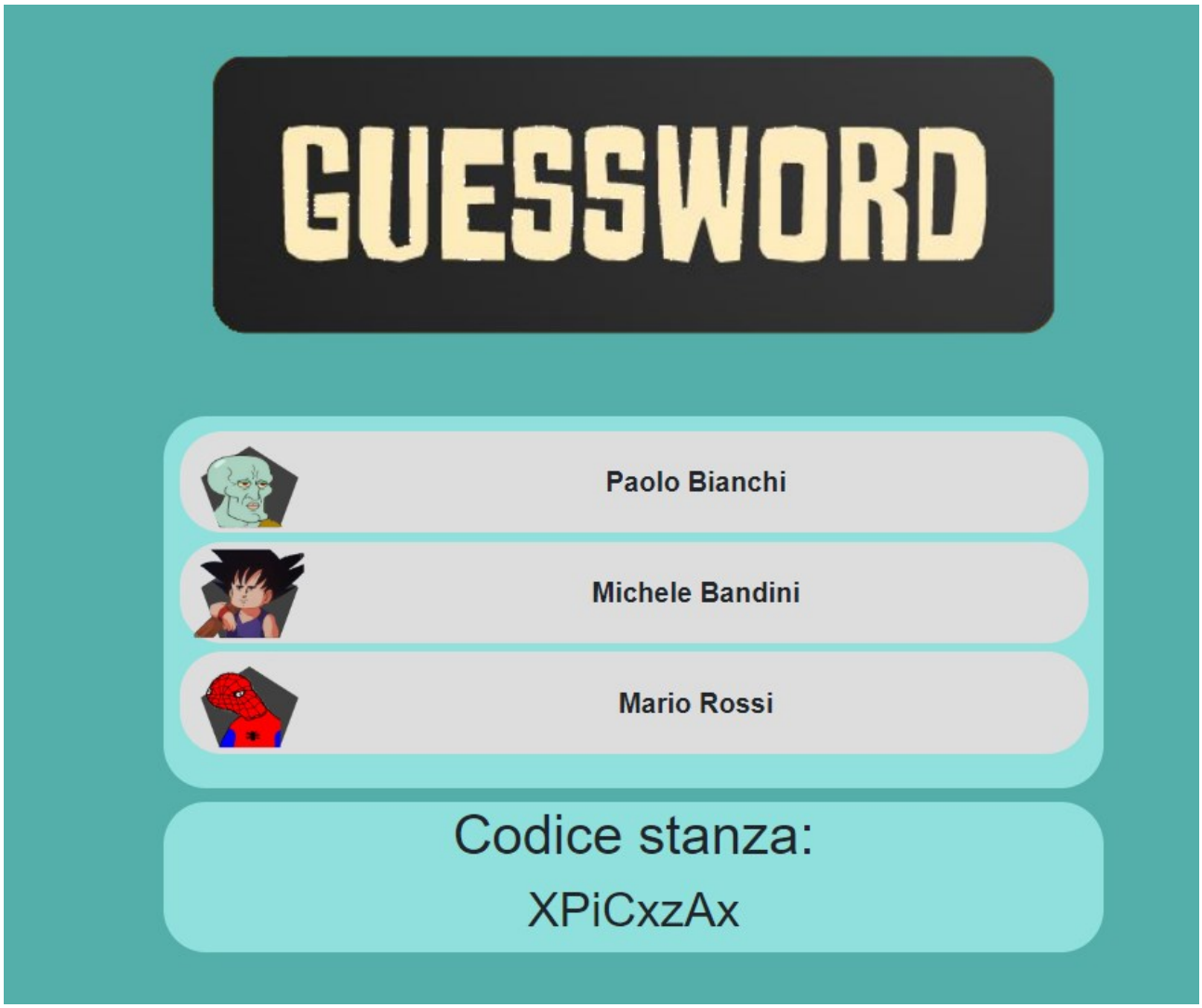

FIGURA 4 LOBBY VISTA DAI PARTECIPANTI

Schermata di gioco

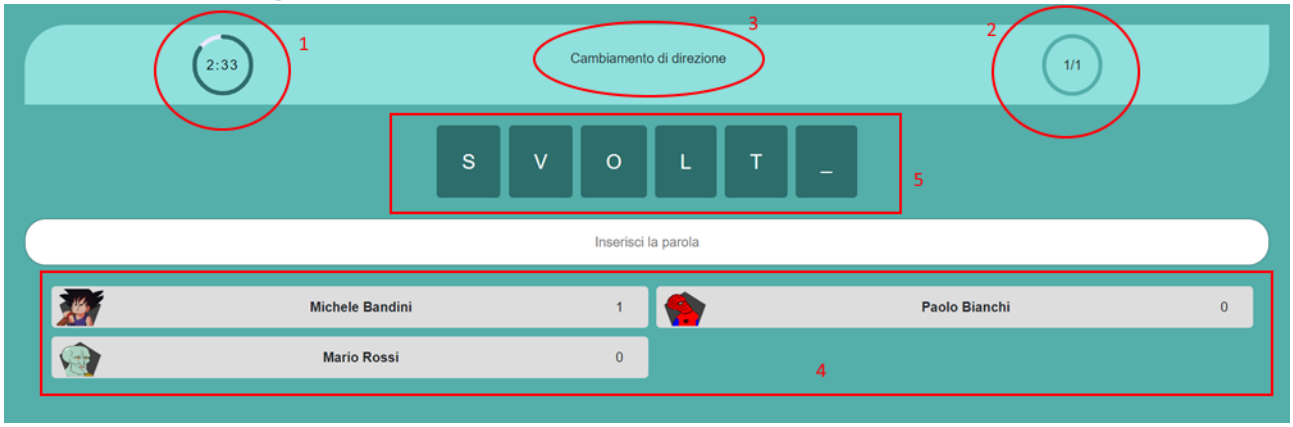

FIGURA 5 SCHERMATA DI GIOCO

1) In modalità scalata indica il tempo rimanente alla fine del round, mentre in modalità battaglia indica il tempo rimasto per indovinare la parola
2) Numero del round corrente a sinistra e numero di round totali sulla destra
3) Definizione della parola da indovinare

4) Classifica in tempo reale
5) Parola da indovinare

Per inserire la parola basterà scriverla nella text box e premere invio. Se la parola è stata indovinata verrà colorata di verde.

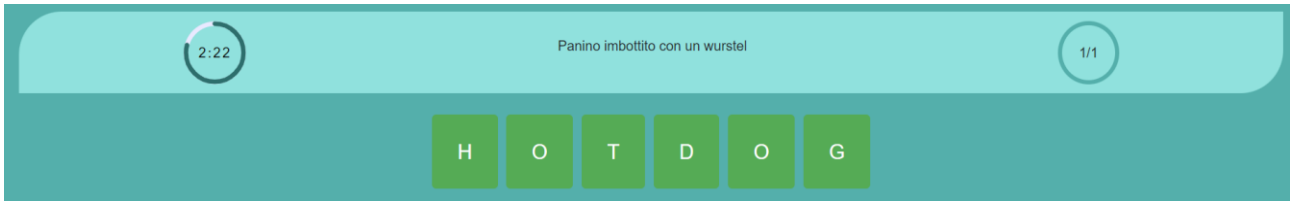

FIGURA 6 RISPOSTA ESATTA

Se invece è stata data una risposta errata verrà questa verrà colorata di rosso.

FIGURA 7 RISPOSTA ERRATA

Comunicazione P2P

Per permettere di giocare in multiplayer GuessWord utilizza l'architettura P2P con un server che svolge il ruolo di broker di connessione. Nessun dato scambiato tra i peer passa attraverso il server.

I peer si scambiano informazioni attraverso pacchetti. Ogni pacchetto contiene tutte le informazioni necessarie per l'esecuzione delle operazioni associate a un determinato messaggio. Il pacchetto è chiamato "peerPacket" e ha la seguente struttura:

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

Il campo "IdMessage" è un identificatore univoco presente in ogni pacchetto inviato tra i peer. Questo campo indica al peer che lo riceve quali informazioni deve utilizzare per eseguire la funzione

associata a quell'ID. Ad esempio, se l'ID del messaggio è impostato su "3", il peer sa che deve utilizzare il contenuto del pacchetto per assegnare il campo *peerPacket.score* al peer di id *peerPacket.id*.

In questo modo, i pacchetti inviati tra i peer sono altamente efficienti e contengono solo le informazioni necessarie per svolgere la funzione associata a un determinato messaggio. Ciò rende il gioco Guessword veloce e reattivo, permettendo ai giocatori di interagire in tempo reale.

Ogni peer salva tutte le informazioni legate agli altri peer attraverso il dictionary *connectedPeerPacket* che ha come chiave l'ID del peer connesso e come valore un peerPacket completo.

La connessione con un altro peer viene salvata nel campo *connectedPeerPacket[id]*.conn e con questo oggetto è possibile inviare i messaggi attraverso il metodo *conn.send().*

Tecnologie utilizzate

Backend

Lato backend è stato utilizzato Node.js che svolge le funzionalità di server web sulla porta 3000 e si occupa di istanziare i vari peer assegnandogli un ID sulla porta 9000.

Ho utilizzato Node.js perché è facile da utilizzare, offre una vasta gamma di risorse già pronte all'uso e ha una forte comunità di sviluppatori che offre supporto e risorse per la risoluzione dei problemi. Grazie a NPM è stato veramente semplice integrare API esterne come, per esempio, PeerJS che ho utilizzato per la comunicazione P2P.

Per quanto riguarda lo storage delle definizioni e delle rispettive parole ho deciso di utilizzare un file JSON, anziché SQLite come avevo inizialmente pensato, questo perché ho incontrato problemi nell'utilizzo delle librerie SQLite con ReactJS e ho voluto velocizzare i tempi di sviluppo. Anche se SQLite sarebbe stato una soluzione più complessa e potente, avrei dovuto dedicare molto tempo alla risoluzione dei problemi di compatibilità e all'integrazione delle librerie.

Invece, utilizzando un file JSON, ho potuto gestire facilmente i dati in modo semplice e veloce, senza dover preoccuparmi di problemi di compatibilità e di configurazione. Inoltre, ho potuto utilizzare librerie già disponibili e facilmente integrabili con il mio codice, il che ha accelerato il processo di sviluppo.

Nonostante l'utilizzo di un file JSON sia meno potente rispetto a SQLite, ho deciso di optare per questa soluzione perché non ho riscontrato problemi di prestazioni nell'utilizzo dei dati e ho potuto sviluppare il progetto in modo più rapido e semplice.

Frontend

Lato frontend ho utilizzato le tecnologie HTML, CSS e JavaScript. Ho sfruttato HTML per strutturare il contenuto e fornire una base solida per la presentazione visiva delle informazioni. Ho poi utilizzato CSS per dare un'estetica accattivante e renderlo più gradevole all'occhio.

Per quanto riguarda la logica di gioco e lo scambio di informazione tra i peer, ho utilizzato JavaScript con il framework React JS per la creazione dell'interfaccia utente, poiché mi ha offerto una serie di strumenti e componenti già pronti all'uso, rendendo più efficiente e veloce lo sviluppo, inoltre ho potuto definire componenti a se stanti che ho potuto riutilizzare in parti diverse del codice.