Progetto di Ingegneria Informatica
Simple Virtual Tabletop Enviroment

Simone Cervini
907374
Anno Accademico 2022/2023
Tutor: prof. Giovanni Agosta

Sommario

1. Specifiche

1.1 Obiettivo

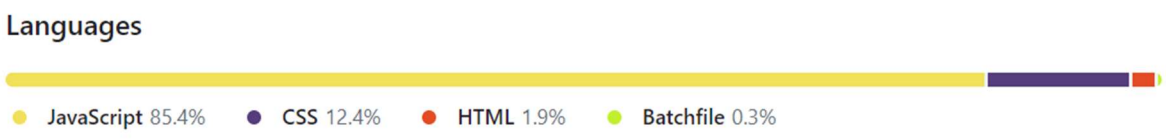
Lo scopo del progetto è sviluppare una piattaforma open source VTT basata su tecnologie moderne all'interno del browser.

L'ambiente web deve risultare semplice e intuitivo per permettere l'utilizzo immediato e non rilegato ad un solo sistema di gioco.

Repository pubblica: https://github.com/Cervini/projectvtt

1.2 Linguaggi e Framework

Il progetto è stato scritto in JavaScript, CSS e HTML.
In aggiunta due file batch permettono l'avvio della piattaforma con più rapidità.



Comprende un ambiente server sviluppato con il framework back-end Express JS (di Node JS), e un ambiente front-end sviluppato con il framework React JS.

Documentazioni ufficiali dei framework:

- Express JS: [https://expressjs.com/]
- Node JS: [https://react.dev/]

1.3 Back end

La componente back end ha l'unico scopo di permette la comunicazione tra client e la sua funzione è semplicemente quella di ricevere ed inviare messaggi.

Il server non memorizza nulla se non i codici delle stanze e i socket che permettono la comunicazione con i client.

L'unica libreria non nativa di Express (Node JS) è socket.io [https://socket.io/] che permette una comunicazione bidirezionale e a bassa latenza tra il server e i client, inoltre implementa la riconnessione automatica da parte del client in disconnessione improvvisa.

1.4 Front end

La quasi totalità delle operazioni che permettono il funzionamento della piattaforma viene eseguita dal client.

Il client si occupa di:

- Creare la stanza alla quale si connetteranno i giocatori
- Modificare la stanza (mappa di gioco, token e movimenti)
- Comunicare al server tutti i cambiamenti in tempo reale

Tutte le modifiche e i file caricati da locale (mappe ed eventuali token) vengono memorizzati sul client e il server si limita a trasferirli agli altri giocatori.

Le librerie non native di React JS utilizzate sono:

- react-rnd
- socket.io-client
- react-icons

Nel dettaglio:

react-rnd [https://www.npmjs.com/package/react-rnd] permette di implementare con semplicità le funzionalità di *drag and drop* e di *resizing* agli elementi presenti sul DOM, è stato utilizzato per permettere un veloce ed immediato movimento dei token sulla mappa e la modifica della loro dimensione.

socket.io-client [https://www.npmjs.com/package/socket.io-client] permette di implementare lato client le funzionalità della libreria socket.io utilizzata nel back end, implementa con facilità e gestisce la connessione con il server.

react-icons [https://www.npmjs.com/package/react-icons] permette l'utilizzo di semplici icone vettoriali.

2. Implementazione

2.1 Front end

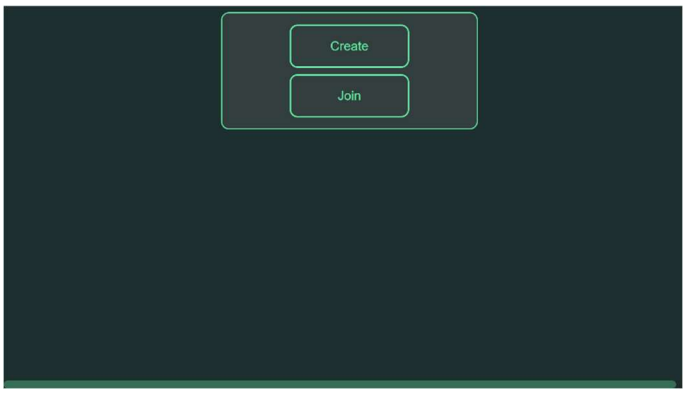
2.1.1. Table

Table è l'istanza che contiene e definisce tutti gli elementi che permettono di visualizzare e descrivono la stanza.

Lo stato di Table è così definito:

`socket`, contiene l'oggetto socket connesso al server
`role`, indica il ruolo del giocatore connesso
`code`, contiene il codice della stanza alla quale si è connessi
`map`, contiene l'URL dell'immagine della mappa
`appear`, flag che permette di visualizzare la finestra con le azioni
`usermenu`, flag che permette di visualizzare il menu con le opzioni di modifica dell'username
`tokens`, array che contiene tutti gli URL delle immagini utilizzate per i token
`tmenu`, flag che contiene l'identificativo dell'ultimo token selezionato o 'false'
`grid`, flag che permette di visualizzare la griglia
`username`, stringa contenente l'username del giocatore connesso
`messages`, array contenente tutti i messaggi inviati nella chat dalla connessione
`newMessage`, stringa che contiene il messaggio digitato prima dell'invio
`height`, altezza in pixel dell'immagine della mappa
`width`, larghezza in pixel dell'immagine della mappa
`cellSize`, dimensione in pixel delle caselle della griglia

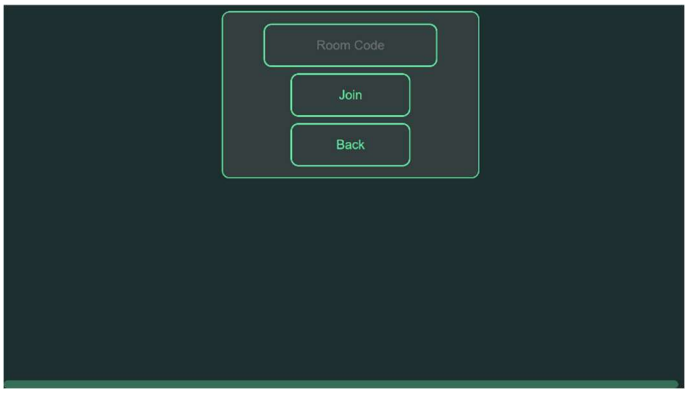
2.1.2. User Interface



Schermata home: questa schermata accoglie gi utenti e permette di creare una nuova stanza (e quindi di generare un nuovo codice) o di entrare in una stanza già creata inserendo il codice di tale stanza.

Selezionando 'Create' viene inviato il comando al server che restituisce un nuovo codice univoco per la stanza.

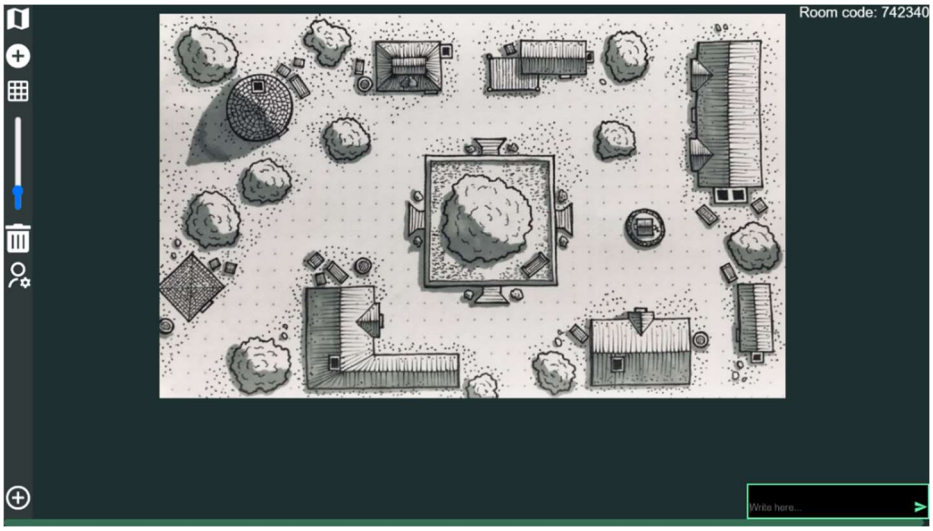
Selezionando Join si viene portati alla schermata che permette l'inserimento del codice della stanza alla quale si vuole accedere.



Schermata di accesso: in questa schermata è possibile inserire il codice di una stanza per farci l'accesso.

Quando un nuovo utente entra nella stanza il server chiede al client che ha creato la stanza di inviare tutti le informazioni utili per conoscere lo stato attuale della stanza, quali: mappa di sfondo, immagini dei token, loro posizioni e dimensioni, e dimensioni delle celle della griglia.

Visualizzazione del tavolo di gioco: nella schermata di gioco sono presenti quattro elementi. A sinistra le **azioni**, nel centro e sullo sfondo la **mappa di gioco**, in altro a destra il **codice stanza** e in basso a destra la **chat**.



Il menu delle **azioni**: questo menu permette all'utente di svolgere determinate azioni (le azioni possibili sono definite dal ruolo quale DM o Player), le azioni possibili sono: cambiare mappa di gioco, aggiungere un token alla mappa dii gioco, rendere visibile o invisibile la griglia, selezionare la dimensione delle celle della griglia, eliminare tutti i cambiamenti e pulire la stanza, e aprire il menu utente che permette di modificare il nome visibile nella chat.

2.1.3 Azioni

Cambiamento della mappa: l'azione permette all'utente di caricare un'immagine dal proprio computer e di impostarla come mappa di gioco per tutti gli utenti connessi alla stanza.

Il file viene usato per costruire un oggetto Blob e inviato al server che procede con l'invio dell'oggetto a tutti gli altri utenti connessi, quando i client ricevono l'oggetto Blob impostano come `map` nello stato l'URL dell'oggetto, modificando così la mappa di gioco visualizzata.

Aggiunta di un token: l'azione permette l'aggiunta di un token (personaggio o mostro, in quanto identici dal punto di vista del software) alla mappa di gioco.

Esattamente come per la mappa l'immagine caricata viene usata per costruire un oggetto Blob, poi inviato agli altri utenti. Per la visualizzazione il token si presenta sopra la mappa di gioco e grazie alla libreria react-rnd può essere spostato e ridimensionato con il puntatore.

Visualizzazione della griglia: l'azione permette di visualizzare la griglia se invisibile o viceversa.

Al click dell'icona il flag `grid` viene modificato, il che ha impatto sul render della griglia che avviene solo quando il flag è positivo. La griglia è costruita come un insieme di div quadrati che presentano un bordo grigio e che si sovrappongono allo spazio coperto dalla mappa.

Dimensione delle celle: l'azione permette di modificare la dimensione delle celle della griglia.

All'interazione con lo slider la dimensione delle celle nello stato viene modificata, questo ha come reazione l'aggiornamento del render della griglia, che viene modificata cambiando il numero di celle e le loro dimensioni (visto che lo spazio da riempire, ovvero l'immagine della mappa, resta sempre lo stesso all'aumentare delle dimensioni, diminuirà il numero).

Reset della stanza: l'azione permette di eliminare tutte le modifiche fatte alla stanza e di tornare allo stato iniziale della stanza.

Menu utente: l'azione permette di aprire il menu utente.

Come per la visualizzazione della griglia l'azione modifica lo stato che, come conseguenza re*nderizza*, il menu o all'opposto, lo nasconde.

2.2 Back End

Come già presentato le funzionalità del back end sono molto semplici, divide i socket per stanza in modo da mandare le comunicazioni di una stanza solo ai membri della stessa e invia i messaggi ricevuti alle stanze o agli utenti.

Alla connessione il server fornisce un codice univoco se il client sta creando una stanza e, una volta fornito al client la stanza viene creata.

Se il client invece si sta connettendo ad una stanza già creata il server lo inserisce tra i client della stanza e manda un comando di aggiornamento forzato al client *DM* della stanza, il quale risponde con tutte le informazioni necessarie per poter ricostruire la visualizzazione della stanza, il server poi invia queste informazioni al nuovo client che può visualizzare la stanza.

3. Installazione

3.1 Windows

Per utilizzare la piattaforma in locale è necessario installare Node JS [https://nodejs.org/en]

Una volta installato Node JS per visualizzare la piattaforma funzionante è necessario:

- scaricare la repository ed estrarla
- eseguire launch_backend.bat

- eseguire launch_frontend.bat

In alternativa è possibile eseguire i seguenti comandi nella cartella backend:

```
npm install
npm run dev
```

E successivamente eseguire i seguenti comandi nella cartella frontend:

```
npm install
npm start
```

La piattaforma viene avviata sulla porta 3000 e il server sulla porta 8080.
Per visualizzare la piattaforma è necessario raggiungere da un qualunque browser l'indirizzo http://localhost:3000/

3.2 Linux

Per utilizzare la piattaforma in locale è necessario installare NodeJS e npm, con i seguenti comandi

```
sudo apt-get update
sudo apt-get install nodejs
sudo apt-get install npm
```

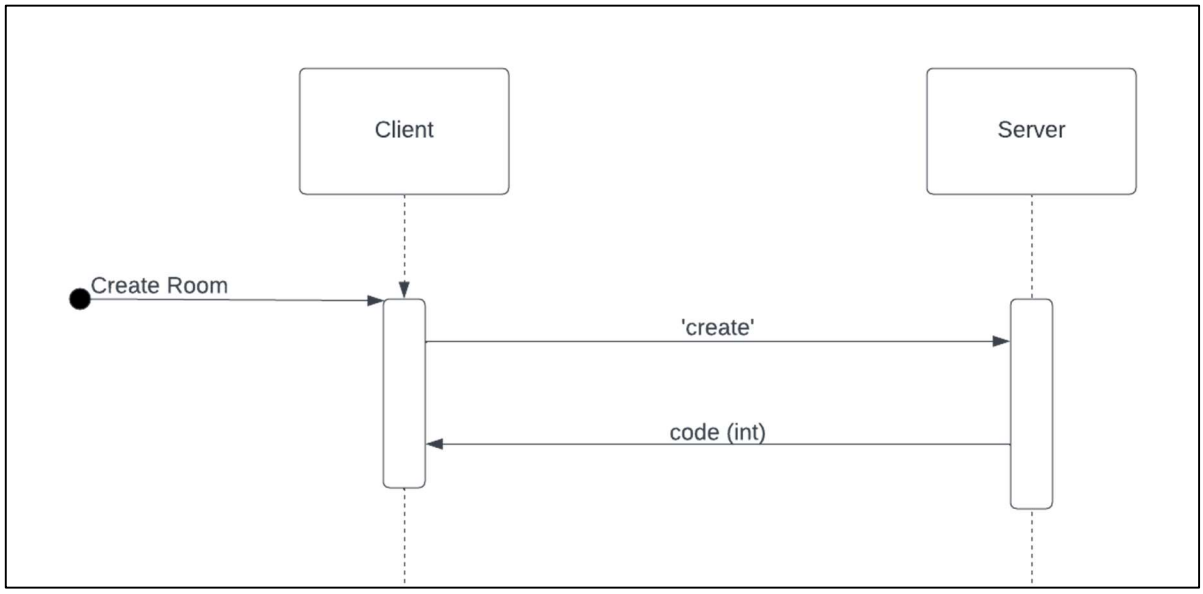
Successivamente per lanciare il server usare il seguente comando (nella directory 'backend'):

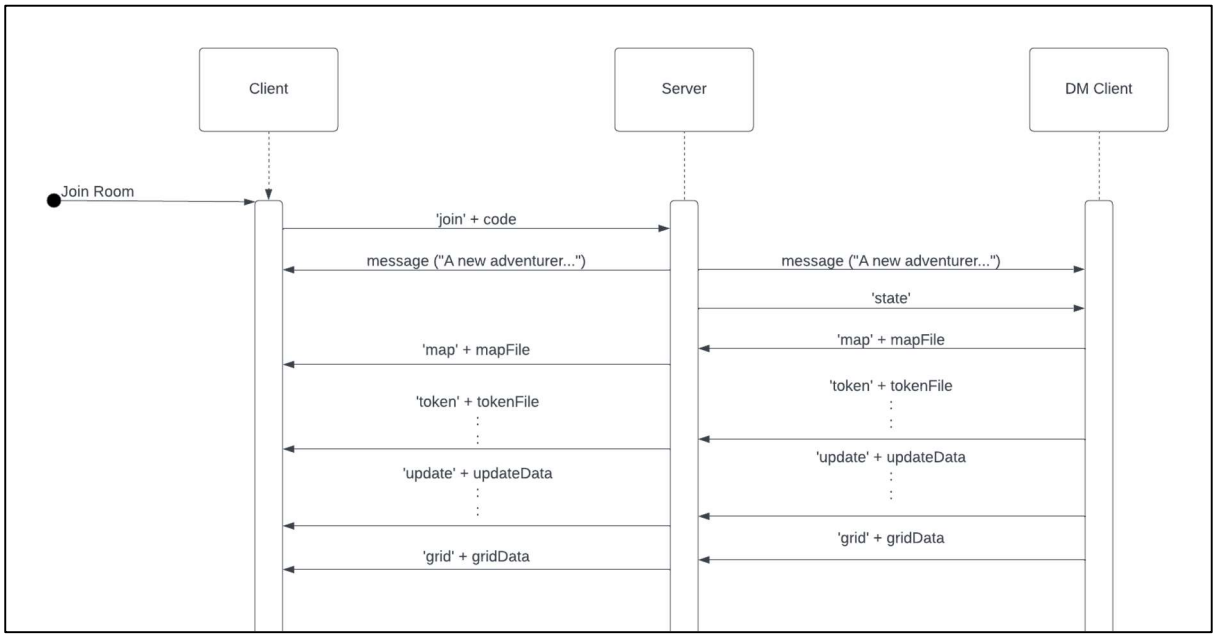
```
node server.js
```

E poi nella cartella 'forntend' utilizzare il comando:

```
npm start
```

4. Sequence Diagram

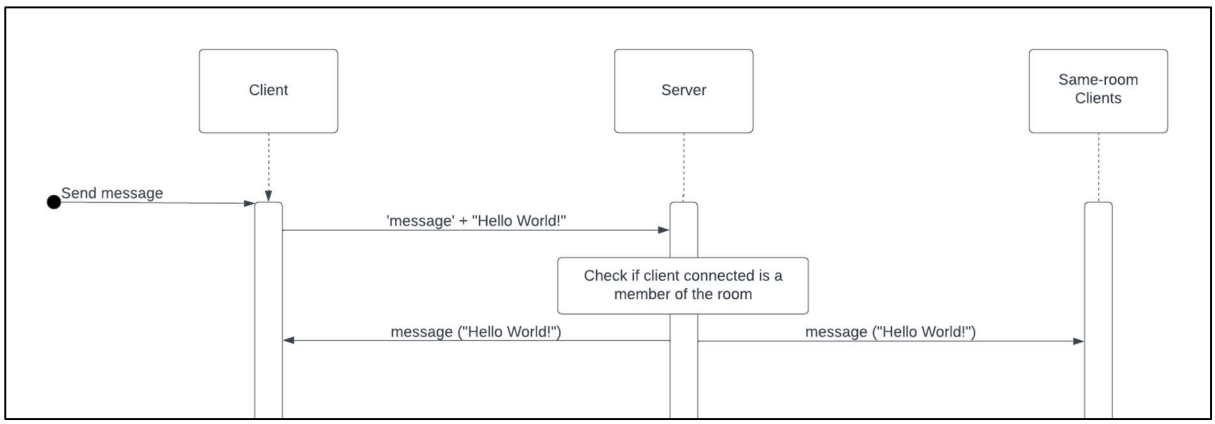




Quando un utente entra in una stanza, il server manda il comando ‘state’ al client DM che per aggiornare la scena del client appena entrato effettua tutte le azioni di aggiornamento in serie:

- aggiorna la mappa
- aggiunge i token
- aggiorna la loro posizione e dimensione
- aggiorna e definisce la visualizzazione della griglia

Per analizzare quindi il Sequence Diagram di ognuna di queste operazioni è sufficiente osservare il Sequence Diagram dell'operazione 'join'.



Quando un utente invia un messaggio il server effettua il broadcast dello stesso per permetterne la visualizzazione, per effettuare un broadcast corretto il server deve filtrare i client per inviare il messaggio solo a quelli che sono membri della stessa stanza (hanno, nel server, stesso 'room code').