Tris!
Applicazioni e Servizi Web

Andrea Sperandio - 0900065041
andrea.sperandio4@studio.unibo.it

Giugno 2022

Tris! è una Single-Page Application che permette ai propri utenti di sfidarsi online all'omonimo gioco, conosciuto anche con il nome di Tic-tac-toe

1 Introduzione

Tris nasce come gioco carta-penna che si svolge tra 2 giocatori su una griglia quadrata 3×3. Ogni partecipante ha associato un simbolo (solitamente 'O' e 'X') e, a turno, sceglie una cella vuota della griglia in cui disegnarlo. Il giocatore che riesce a disporre tre dei propri simboli in linea retta orizzontale, verticale o diagonale vince la partita. Se la griglia viene riempita senza che nessuno dei giocatori sia riuscito a completare una linea retta di tre simboli, il gioco finisce in parità.[17]

L'obiettivo di questo progetto è quello di realizzare una versione online del gioco Tris che permetta agli utenti di sfidarsi da remoto. L'altro punto chiave è quello di mettere a disposizione dei suoi utilizzatori una schermata di statistiche con lo scopo di analizzare l'andamento delle partite, utili, ad esempio, per scoprire se è più probabile che vinca il giocatore che fa la prima mossa o lo sfidante.

2 Requisiti

Il sistema prevede la realizzazione di un'applicazione web con le seguenti funzionalità:

- creazione di stanze in cui sfidarsi a Tris
  - scelta del nome della stanza
  - scelta del giocatore che farà la prima mossa
- consultazione stanze aperte
  - numero di stanze attualmente disponibili, in cui è possibile entrare
  - nome della stanza e giocatore iniziale per ciascuna stanza

  - ingresso in una stanza
- gestione della partita
  - visualizzazione stato della partita
  - possibilità di abbandonare la sfida
  - avvio della partita
  - chat per scambiare messaggi con l'avversario
  - rivincita a partita conclusa, con cambio del giocatore iniziale
- visualizzazione di informazioni in tempo reale
  - numero di giocatori online
  - numero di partite in corso
  - numero di partite giocate in tutto
  - eventuali stanze disponibili
- visualizzazione di dati statistici
  - numero di partite giocate
  - numero di partite cancellate
  - numero medio di mosse per partita
  - numero medio di rivincite per partita
  - giocatore più vincente tra chi fa la prima mossa e l'avversario
  - posizione più vincente (per righe, colonne o diagonali)

Il sistema deve permettere ad un utente di entrare in una stanza tramite link e deve mantenere uno stato di consistenza interna in caso di azioni inaspettate da parte dei giocatori, come la chiusura della tab del browser (o del browser stesso) a partita in corso o la perdita della connessione ad internet. Deve essere possibile muoversi tra le pagine dell'applicazione sia direttamente (tramite url), che attraverso i link presenti nelle pagine stesse, che tramite i bottoni di navigazione *Indietro* e *Avanti* messi a disposizione dal browser. L'applicazione, infine, deve essere accessibile da ogni tipo di dispositivo, desktop, tablet e mobile, a prescindere dal sistema operativo o dal browser utilizzato e deve risultare semplice da usare, intuitiva e offrire un'esperienza di gioco gradevole.

3 Design

La metodologia seguita per il design dell'applicazione è quella Agile, con l'obiettivo di rilasci frequenti ed incrementali di nuove versioni con funzionalità aggiuntive, tenendo a mente gli aspetti critici del sistema ed i bisogni utente. Inizialmente quindi, scelta l'architettura generale del sistema, si è sviluppato il core dell'applicazione: creazione ed

accesso alle stanze e gestione della partita. Successivamente, sono state aggiunte funzionalità come la chat e la visualizzazione dei dati statistici. In questo caso particolare, l'obiettivo consisteva in un'unica consegna dell'applicazione finita, per cui lo stile delle interfacce utente è stato aggiunto in un secondo momento. In un caso reale di sviluppo del sistema con consegne di più versioni incrementali e funzionanti, invece, anche la presentazione grafica delle interfacce utente avrebbe costituito un'attività fondamentale da integrare ad ogni sprint.

L'architettura del sistema scelta è quella di un'applicazione a 3 livelli, che comporta la realizzazione di 3 componenti principali in comunicazione tra loro:

- applicazione lato client
- applicazione lato server
- database

Il design dell'applicazione prevede l'utilizzo di un Database per la memorizzazione dei dati e l'indagine statistica a posteriori: qui persistono le informazioni relative alle stanze, come nome, giocatore iniziale, mosse effettuate sulla griglia di gioco. L'applicazione viene poi divisa tra client e server. Al server è delegata:

- la logica del gioco, è il server a determinare il termine della partita e l'eventuale vincitore a mano a mano che gli sfidanti compiono mosse
- la gestione delle connessioni, i clients infatti non comunicano direttamente tra loro, ma scambiano informazioni con il server, che le trasmette agli endpoint interessati
- la trasmissione dei dati di gioco e di partita in tempo reale
- l'interfacciamento con il Database, con conseguente inserimento, aggiornamento e analisi dei dati

Il client, invece, realizza lo strato di presentazione dell'applicazione, che riceve i dati dal server, anche in tempo reale, li mostra all'utente umano e reagisce ai suoi input, eventualmente comunicando con il server. Rappresenta quindi la porta dell'applicazione per l'utente finale ed incorpora la logica per gestire gli input utente e quelli provenienti dal server.

Le interfacce utente, codificate lato client, si occupano di mostrare in tempo reale dati ed informazioni all'utente e di reagire alle sue azioni. Sono progettate tenendo a mente il principio di discoverability, intesto come l'abilità di scoprire in autonomia quali operazioni si possono fare, e le 10 euristiche di usabilità di Nielsen [3] al fine di ottimizzare la user experience e, di conseguenza, la soddisfazione dell'utente. L'applicazione è dotata di interfacce grafiche responsive, in modo da essere fruibile da ogni tipo di dispositivo desktop, tablet o mobile. Prima dello sviluppo, sono stati realizzati i seguenti Mockup, approvati dal committente:

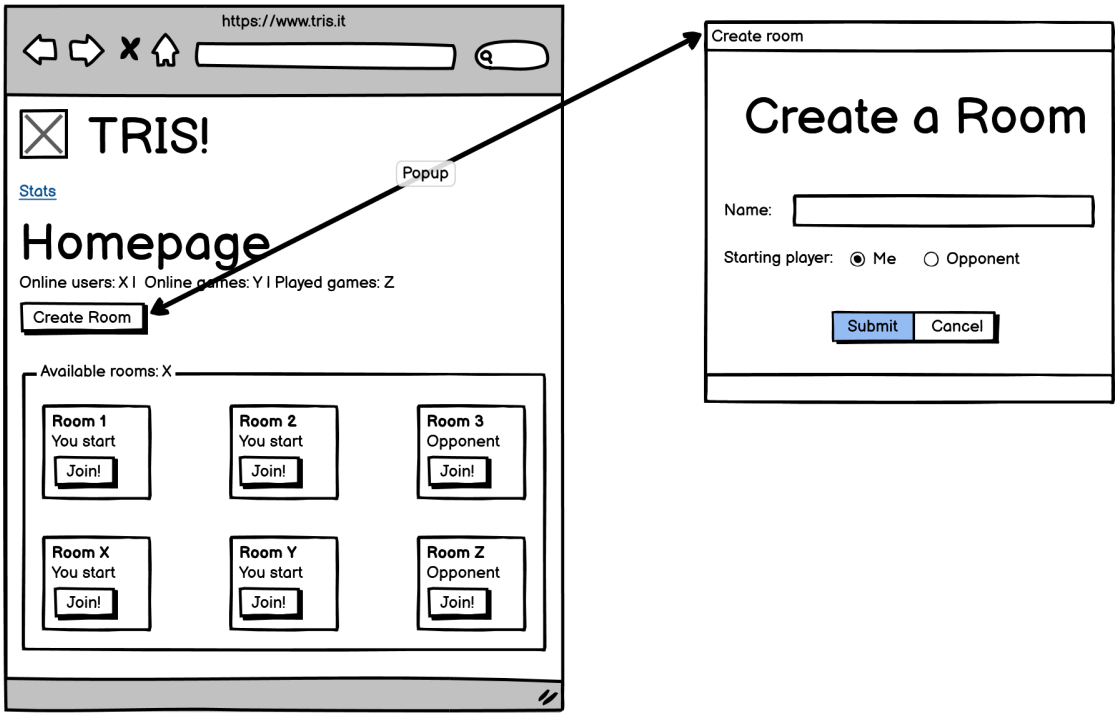

Figura 1: Mockup Homepage

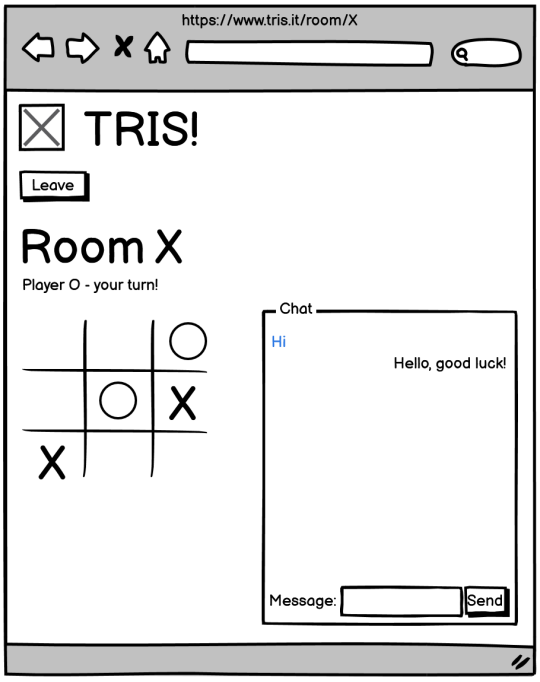

Figura 2: Mockup Roompage

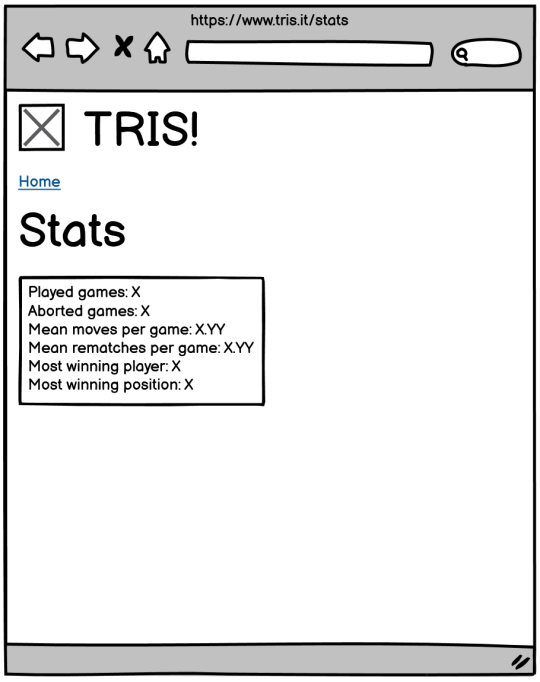

Figura 3: Mockup Statspage

La Homepage 1 mostra alcuni dati in tempo reale, permette di creare una stanza ed elenca le stanze disponibili, in attesa di un giocatore per iniziare la partita. La Roompage 2 rappresenta la stanza che ospita la sfida vera e propria, è munita di chat e di grafica conforme al gioco del Tris. Infine, la Statspage 3 è dedicata alla consultazione dei dati statistici aggregati per analisi e curiosità.

4 Tecnologie

Le tecnologie adottate per lo sviluppo del progetto proposto corrispondono a quelle che compongono il signature stack MERN: MongoDB + Express.js + React.js + Node.js [8].

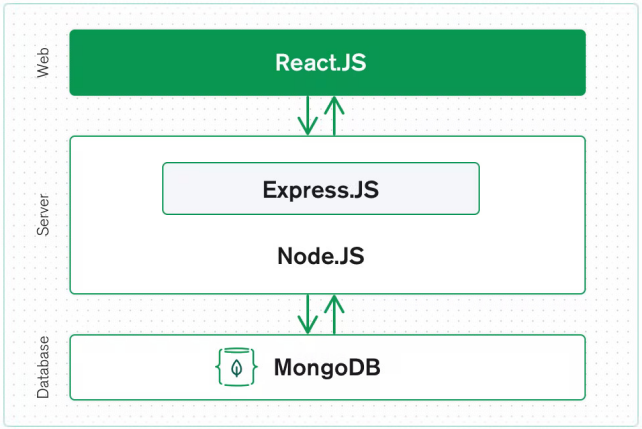

Figura 4: Stack MERN

L'applicazione è costituita da un'architettura a 3 livelli: frontend (client), backend (server) e database, come già detto in 3.

- Frontend: React è il framework javascript lato client che permette di integrare il codice HTML di presentazione dell'applicazione con i dati ricevuti dal server[12]. Utilizza un modello a componenti stateful integrati tra loro e guidati dai dati, che tengono aggiornate le view in base agli ultimi dati ricevuti, realizzando così una consistenza tra ciò che vede l'utente e lo stato interno dell'applicazione. Questa, lato client, effettua richieste HTTP all'applicazione lato server grazie alla libreria Axios (client HTTP basato su promise [2]) e riceve automaticamente dati in real time in push dal server grazie al meccanismo delle socket, implementato attraverso la libreria Socket.IO [15]. Infine, per gestire al meglio il routing all'interno della Single Page Application Tris lato client, React è stato integrato con la libreria React Router[13].

- Backend: Node.js è una piattaforma software Javascript, asincrona e ad eventi, utilizzata lato server per sviluppare un server web HTTP [10]. Nell'arco di questo progetto è stata integrata tramite il modulo Express, che permette di gestire il routing degli URL e le richieste/risposte HTTP. Per dialogare con il Dabatabase MongoDB si è scelta la libreria Mongoose, che permette di modellare i dati applicativi tramite uno schema, aiutando lo sviluppatore con tipizzazione, validazioni e costruzione delle query [9].
- Database: MongoDB è un database di tipo documentale (non relazionale, no-SQL), utilizzato per memorizzare i dati relativi alle partite. É uno strumento multi-piattaforma che si integra perfettamente con React ed Express+Node perché memorizza e trasmette i documenti come oggetti JSON (JavaScript Object Notation).

La scelta dello stack MERN rende totalmente fluido lo scambio di dati tra client, server e database, non richiedendo alcun tipo di conversione o cambio di codifica. Inoltre, lo sviluppo si basa unicamente sul linguaggio di programmazione Javascript e sulla notazione JSON, oltre che sui linguaggi base del web (HTML + CSS). Possibili alternative tecnologiche a React ugualmente valide sono Angular (stack MEAN) e Vue (stack MEVN), la scelta in questo caso è ricaduta su React poiché l'applicazione non richiede tutte le funzionalità all-in-one offerte da Angular, ma rapidità di esecuzione e di sviluppo [11]. Al fine di produrre un'applicazione responsive e fluida è stato fatto uso di grid [6] e flexbox [4] lato CSS, oltre che di media query [16]. Inoltre, per facilitare, velocizzare e meglio organizzare lo stile grafico dell'applicazione lato client, si è utilizzato il preprocessore SASS (Syntactically Awesome Style Sheets) con la sintassi SCSS (Sassy CSS), che permette di estendere direttamente il codice CSS con direttive più potenti [14].

5 Codice

All'avvio dell'applicazione, ad ogni utente viene assegnato un id univoco, memorizzato nella sessionStorage del browser. Questo significa che l'id rimarrà invariato durante la navigazione di una sessione, ma ad ogni sessione ne verrà generato uno differente: al ricaricamento della pagina l'id viene mantenuto, alla chiusura della tab del browser, invece, questa informazione viene persa. Il dato serve ad identificare l'utente e permette il corretto svolgimento delle partite, grazie all'associazione delle mosse ad un determinato giocatore. Generato o recuperato l'id dalla sessionStorage, questo viene passato al server durante l'apertura di una socket bidirezionale, che consente a quest'ultimo di inviare gli aggiornamenti in push al client, ossia all'applicazione lato utente. Questa connessione rimane attiva per tutta la durata della sessione e viene chiusa al termine della stessa o al ricaricamento della pagina da parte dell'utente (nonostante la sessione non termini). Il funzionamento attuale dell'applicazione prevede l'annullamento di una partita in corso se un utente si disconnette dal server, quindi se chiude o ricarica la pagina o incorre in un problema di connessione di rete. Alla connessione al server, l'utente riceve dati aggiornati sulle stanze presenti, sulle partite in corso e su quelle giocate. Alla disconnesione, invece,

viene rimosso dalla stanza, se presente. In entrambi i casi, tutti gli utenti attivi ricevono un aggiornamento in tempo reale del numero di giocatori connessi. La maggior parte dei dati che l'applicazione scambia con il server sono trasmessi ed aggiornati in tempo reale, la pagina dedicata alle statistiche, invece, recupera le informazioni al caricamento, ma occorre un refresh manuale per visualizzare i dati più aggiornati. Questo è un comportamento voluto per evitare di sovraccaricare il server ed il database, sia in termini di elaborazioni, che di banda utilizzata.

6 Test

In fase di sviluppo, all'aggiunta di ogni nuova funzionalità è stato effettuato un walk-through completo del sistema per verificare il corretto funzionamento di tutte le sue componenti. Ovviamente, per il corretto mantenimento futuro dell'applicazione sarebbe necessario creare dei regression tests automatici in grado di monitorare lo stato del sistema ed impedire la pubblicazione di nuove versioni che, introducendo nuove funzionalità o correzioni a bug, causino errori in altre parti dell'applicazione, precedentemente funzionanti.

Oltre ai tests effettuati dallo sviluppatore, è stato realizzato uno usability test con co-discovery learning, come documentato dal video allegato. In particolare, due utenti reali si sono ritrovati per la prima volta davanti all'applicazione finita e, insieme, ne hanno esaminato, prima liberamente e poi guidati e orientati a tasks, funzionalità ed usabilità, con l'obiettivo di valutarne le potenzialità, la facilità d'uso e, sopratutto, il livello di raggiungimento dello scopo principale lato esperienza utente: la possibilità di giocare delle partite a Tris da remoto. Successivamente, è stato chiesto ai partecipanti di compilare un report, motivando le risposte, per valutare l'usabilità dell'applicazione. Di seguito ne vengono riportati i risultati:

Aspetto	Voto Marialisa	Voto Andrea	Voto Medio	Commento Marialisa	Commento Andrea
Visibilità dello stato del sistema	5	5	5	-	-
Corrispondenza tra sistema e mondo reale	5	5	5	-	C'è anche la faccina quando perdo
Controllo e libertà	4	4	4	Per vedere le statistiche devo uscire dal gioco e tornare indietro	Il tasto delle statistiche non è immediato da vedere
Consistenza e standard	5	5	5	-	-

Prevenzione dell'errore	5	5	5	Non ti è concesso sbagliare	-
Riconoscimento anziché ricordo	5	4	4.5	-	Interfaccia minimale: pochi tasti e non puoi sbagliare, ma era meglio se ci fossero stati suggerimenti perché è molto schematico
Flessibilità ed efficienza d'uso	-	-	-	Non sono previste differenziazioni a seconda dell'esperienza utente maturata	-
Design ed estetica minimalista	4	5	4.5	Non c'era nulla di più del contenuto o qualcosa che potrebbe distrarre. Avrei aggiunto qualcosa per distinguerlo da prodotti simili e renderlo più carino	Meglio di tanti altri giochi che si trovano al telefono, che sono pieni di banner pubblicitari ed è difficile giocarci
Aiuto all'utente	5	5	5	È talmente difficile sbagliare in questo gioco che non abbiamo visto nulla. Dopo aver visto, a posteriori, cosa succede se l'utente cerca di uscire a metà partita, allora posso dare una valutazione	-

Documentazione	1	3	2	Assente	Come detto in *Riconoscimento anziché ricordo*, avrei avuto bisogno di una pagina con le regole di gioco e di suggerimenti durante la navigazione
Facilità d'uso complessiva	5	4	4.5	Conoscevo già il gioco ed il sistema è molto facile da usare	Non ricordavo bene le regole del gioco, avrei gradito poter consultare le istruzioni
Soddisfazione utente	5	5	5	Oltre alla facilità di gioco, l'aggiunta della chat ha reso più gradito il gioco perché fornisce la possibilità di comunicare con la persona con cui stai giocando	Ho gradito molto la chat

A seguito del test di usabilità, si può concludere che il prodotto risulta intuitivo, semplice, minimale e facilmente utilizzabile. Gli utenti sono soddisfatti delle caratteristiche del prodotto, ma suggeriscono alcuni importanti miglioramenti, tra cui:

- maggiore visiblità della navigation bar
- pagina dedicata alle regole del gioco
- maggiori indicazioni testuali su cosa si può fare, come e quando
- suggerimenti in popup durante la navigazione per facilitare l'utente inesperto

Ovviamente, il campione utilizzato per i suddetti test risulta troppo limitato per poter trarre conclusioni affidabili e sarebbe necessario sottoporre allo stesso test anche altre coppie di candidati, magari con età e conoscenze informatiche differenti.

7 Deployment

Al fine di poter utilizzare l'applicazione in ambiente di test, occorre innanzitutto scaricarne i sorgenti da Github, dove è presente anche un file README contenente queste stesse istruzioni. Inoltre, è necessario aver installato e configurato Node.js, che viene distribuito già equipaggato del suo package manager (npm). Fatto ciò è possibile avviare l'applicazione:

- lato client:
  1. spostarsi nella cartella *client*
  2. eseguire il comando *npm install* per installare tutte le dipendenze
  3. eseguire il comando *npm run build* per generare la cartella build contenente i sorgenti dell'applicazione ottimizzati
- lato server:
  1. spostarsi nella cartella *server*
  2. eseguire il comando *npm install* per installare tutte le dipendenze
  3. eseguire il comando *node index.js* per avviare il server

L'applicazione è ora accessibile all'indirizzo http://localhost:4001, raggiungibile, ad esempio, tramite browser. Terminata questa prima configurazione, per avviarla nuovamente basterà semplicemente lanciare il comando *node index.js* all'interno della cartella *server* e visitare l'indirizzo appena indicato.

Al fine di pubblicare l'applicazione online è necessario associare al server web un indirizzo IP statico e pubblico, raggiungibile quindi anche dall'esterno.

8 Conclusioni

I due obiettivi principali del progetto, ossia la realizzazione del gioco Tris online e la visualizzazione delle statistiche, sono stati soddisfatti. Oltre ai miglioramenti necessari emersi durante il test di usabilità, l'applicazione potrebbe essere estesa con nuove funzionalità, ad esempio:

- assegnazione di un nome utente
- creazione profilo personale per visualizzazione statistiche per utente
- visualizzazione del punteggio in caso di rivincite
- creazione di stanze protette da password
- miglioramento robustezza dell'applicazione, in modo da non annullare una partita in corso se un utente ricarica la pagina o incorre in un problema di perdita momentanea della connessione di rete e da gestire correttamente la navigazione da browser attraverso i tasti *Indietro* e *Avanti*

- possibilità di sfidare il computer
- suggerimento all'utente della migliore prossima mossa
- analisi più approfondite nella pagina dedicata alle statistiche
- gestione della visualizzazione interattiva di una vecchia partita grazie allo storico delle mosse effettuate

Le tecnologie utilizzate sono risultate ideali ed adatte per raggiungere gli scopi del progetto. Anche la scelta di design di strutturare l'applicazione come una Single-page application con architettura a 3 livelli si è dimostrata corretta ed ha permesso di soddisfare i requisiti utente tramite un modello semplice da gestire, manutenere ed espandere.

Riferimenti bibliografici

[1] Create React App. Deployment. `https://create-react-app.dev/docs/deployment/`.

[2] Axios. Getting started. `https://axios-http.com/docs/intro`.

[3] Alessandro De Cenzo. Le 10 euristiche di nielsen. `https://blankgrowth.agency/blog/digital-marketing/valutazione-euristica/`.

[4] Chris Coyier. A complete guide to flexbox. `https://css-tricks.com/snippets/css/a-guide-to-flexbox/`.

[5] Express.js. Express. `https://expressjs.com/`.

[6] Chris House. A complete guide to grid. `https://css-tricks.com/snippets/css/complete-guide-grid/`.

[7] MongoDB. Introduction to mongodb. `https://www.mongodb.com/docs/manual/introduction/`.

[8] MongoDB. Mern stack: Full-stack web application development. `https://www.mongodb.com/mern-stack`.

[9] Mongoose.js. mongoose. `https://mongoosejs.com/`.

[10] Node.js. About node.js®. `https://nodejs.org/en/about/`.

[11] Aris Pattakos. Angular vs react vs vue 2022. `https://athemes.com/guides/angular-vs-react-vs-vue/`.

[12] React. Tutorial: Intro to react. `https://reactjs.org/tutorial/tutorial.html`.

[13] React Router. React router v6 is here. `https://reactrouter.com/`.

[14] SASS. Sass basics. https://sass-lang.com/guide.

[15] Socket.IO. Introduction. https://socket.io/docs/v4.

[16] W3Schools. Css @media rule. https://www.w3schools.com/cssref/css3_pr_mediaquery.asp.

[17] Wikipedia. Tris (gioco). https://it.wikipedia.org/wiki/Tris_(gioco).