Daniele Foffano, 853953
10 giugno 2018

Report Progetto
Tecnologie e Applicazioni Web

Battaglia navale

Daniele Foffano, Alberto Veneri, Enrico Vettori

1. Introduzione

La consegna del progetto prevedeva la creazione di un'applicazione web, utilizzando server con API in stile REST e un front-end di tipo SPA.
Questa applicazione doveva consentire ad una community di utenti di giocare a battaglia navale, consentendo a ciascun utente operazioni quali: la registrazione dell'account (il login e anche la cancellazione dello stesso), la ricerca di partite disponibili, la visualizzazione di una classifica degli utenti con numero di partite vinte/perse e ovviamente il giocare la partita contro un altro giocatore.

2. Sviluppo e spiegazioni

2.1. API REST e SPA

Come sottolineato nell'introduzione, in questo progetto abbiamo utilizzato delle API in stile REST e un front-end SPA.

- REST (Representational State Transfer) è un particolare tipo di architettura software, che prevede la suddivisione delle funzionalità e dell'applicazione in risorse web. Ogni risorsa deve essere unica e indirizzabile attraverso la sintassi dei link ipertestuali. Pone anche dei vincoli riguardo al rapporto tra client e server, di seguito i più importanti:
  innanzitutto devono essere due entità ben distinte, con i propri ruoli (es.: il client non si occuperà del salvataggio delle informazioni, e il server non si occuperà della veste grafica), e la comunicazione tra le due entità è stateless, ovvero nessuno stato di una sessione client verrà memorizzato all'interno del server.
- SPA (Single-Page Application) è una tipologia di applicazione che può essere consultata e utilizzata rimanendo sempre sulla stessa pagina (esempio famoso: Facebook. Esempio meno famoso: la nostra applicazione web su battaglia navale). Questo significa che tutto il codice necessario alla visualizzazione della

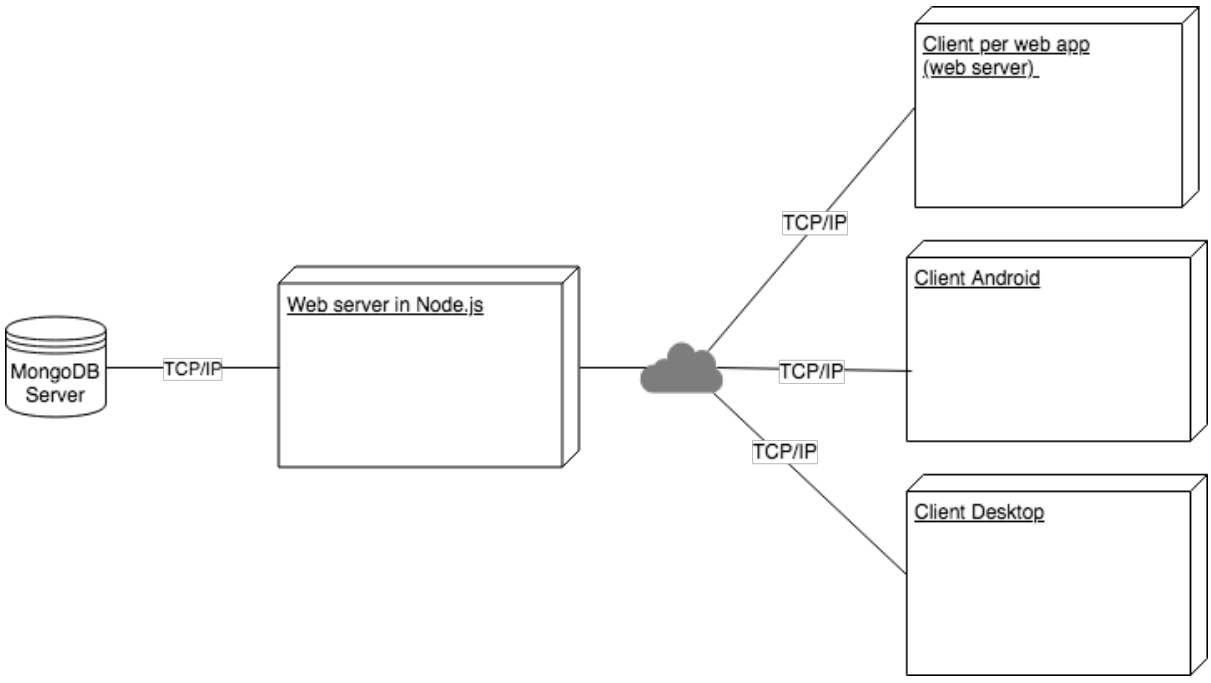
pagina è recuperato attraverso una sola operazione di caricamento, mentre invece le risorse vengono caricate dinamicamente su richiesta dell'utente (o comunque solamente quando necessario) portando quindi ad una maggiore efficienza dell'applicazione.

2.2. Tecnologie utilizzate

Prima di descrivere nel dettaglio il codice del progetto, è bene soffermarsi sulle tecnologie utilizzate: abbiamo utilizzato lo "stack" MEAN, ovvero MongoDB, Express.js, Angular e Node.js.

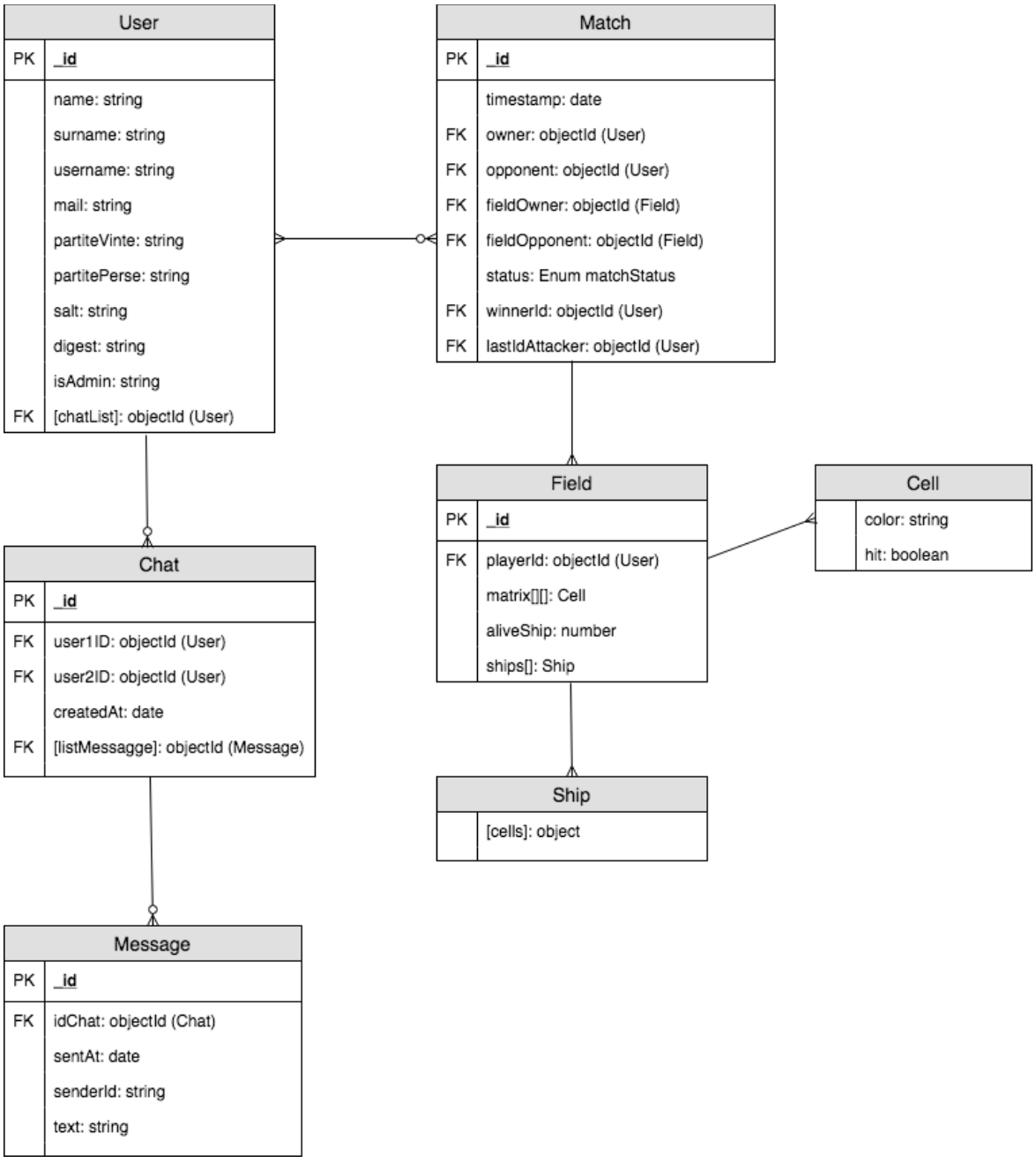
- MongoDB: è un database NoSQL, di tipo documentale. Il vantaggio nel suo utilizzo sta non soltanto nella sua semplicità e efficienza, ma anche nel fatto che i documenti memorizzati e ritornati dalle query sono tutti in formato JSON, che calza a pennello con l'utilizzo del linguaggio javascript.
- Express.js: è un framework progettato per supportare la gestione di applicazioni web su Node.js. Il suo compito sarà quello di gestire le chiamate agli endpoint.
- Angular: è un framework per la componente front-end di una web application, nello specifico applicazioni di tipo SPA (Single-Page Application).
- Node.js: è un ambiente I/O event-driven. Nel processare le richieste, attenderà una notifica che segnali la disponibilità della risorsa da restituire. Le richieste non saranno bloccanti, infatti, pur rimanendo in attesa delle risorse, Node.js servirà altre richieste.

Qui sotto una illustrazione di come avviene la comunicazione tra le varie entità dell'applicazione.



2.3. MongoDB: Modello dei dati

Di seguito una schematizzazione delle collezioni di MongoDB, delle classi utilizzate e delle loro relazioni:



Le collezioni di MongoDB sono 5:

- User: contiene tutte le informazioni dell'utente.
- Chat: contiene lo scambio di messaggi tra due utenti distinti (è unica per ogni coppia di utenti).
- Message: contiene un messaggio appartenente alla chat idChat, spedito dall'utente senderId
- Match: contiene tutte le informazioni sul match, compresi i due utenti partecipanti ed i rispettivi campi di gioco.
- Field: contiene le informazioni sul campo di gioco dell'utente playerId, quindi la disposizione delle sue navi, il suo campo allo stato attuale e il numero di navi ancora intatte.

Abbiamo inoltre utilizzato due classi, per meglio definire le meccaniche di gioco e la struttura del campo:

- Ship: rappresenta la nave, contiene le coordinate delle celle su cui è posizionata.
- Cell: rappresenta la cella del campo, contiene il colore con cui deve essere raffigurata ed una proprietà aggiuntiva che indica se sia stata già colpita o meno.

Si noti che, sebbene nello schema vengano indicati alcuni campi delle collezioni come Foreign Key, questo concetto non è propriamente corretto per un database NoSQL come MongoDB: sarebbe più corretto definirle come references ad altri documenti all'interno del database.

2.4. Express: API

Come già sottolineato, il compito di Express sarà quello di gestire gli endpoint della nostra applicazione; qui di seguito le API REST che abbiamo utilizzato per ciascuno di essi.

Endpoint	**Metodo**	**Attributi**
/	Get	
Descrizione	Ritorna la lista degli endpoint principali in formato JSON	

Endpoint	**Metodo**	**Attributi**
/renew	Get	
Descrizione	Rinnova il JWT dell'utente loggato	

Endpoint	Metodo	Attributi
/login	Get	
Descrizione	Vengono inviati i dati dell'utente e verificate le credenziali, viene salvato anche un JWT per l'autenticazione.	

Endpoint	Metodo	Attributi
/login[?remindMe=True]	Get	remindMe
Descrizione	Vengono passati i dati dell'utente e viene salvato un JWT per l'autenticazione. Se il campo remindMe è settato a true, l'utente verrà ricordato.	

Endpoint	Metodo	Attributi
/users	Post	Email, password, username, nome e cognome
Descrizione	Crea un nuovo utente con gli attributi inseriti in fase di registrazione. Ritorna errore se manca la password o se il nome o l'email sono già state usate da altri.	

Endpoint	Metodo	Attributi
/users/:username	Get	
Descrizione	Ritorna in formato JSON le informazioni dell'utente :username, se non esiste ritorna errore	

Endpoint	Metodo	Attributi
/users/:username	Put	email, password, username, nome, cognome, ruolo
Descrizione	Aggiorna l'utente username con i dati passati come attributi. Un utente può essere modificato soltanto da sé stesso o da un utente admin. Il ruolo di admin può essere dato o tolto ad altri utenti solo da un admin.	

Endpoint	Metodo	Attributi
/users/:username	Delete	
Descrizione	Elimina l'utente username. Oltre all'utente stesso, soltanto gli admin possono eliminare un utente.	

Endpoint	Metodo	Attributi
/users/:username/matches	Get	
Descrizione	Ritorna in formato JSON una lista dei match dell'utente.	

Endpoint	Metodo	Attributi
/chats	Get	
Descrizione	Ritorna la lista di tutte le chat dell'utente autenticato	

Endpoint	Metodo	Attributi
/chats	Post	Id_destinatario
Descrizione	Crea una nuova chat tra l'utente autenticato e l'utente id_destinatario. La chat è unica per ogni coppia di utenti.	

Endpoint	Metodo	Attributi
/chats/:id	Get	
Descrizione	Restituisce la chat avente l'id specificato	

Endpoint	Metodo	Attributi
/chats/:id	Post	Messaggio
Descrizione	Aggiunge il messaggio alla chat con l'id specificato	

Endpoint	Metodo	Attributi
/chats/:id	Delete	
Descrizione	Cancella la chat con l'id specificato	

Endpoint	Metodo	Attributi
/scoreboard	Get	limit, type
Descrizione	Ritorna un file JSON contenente una lista degli utenti, che possono essere ordinati decrescentemente in base al totale delle partite, al numero delle partite vinte o di quelle perse. C'è anche la possibilità di limitare la lista attraverso l'attributo limit.	

Endpoint	Metodo	Attributi
/matches	Get	
Descrizione	Ritorna la lista dei match in stato di attesa sotto forma di file JSON.	

Endpoint	Metodo	Attributi
/matches	Post	
Descrizione	Crea un nuovo match il cui owner sarà l'utente autenticato. Un utente non può creare un nuovo match se è già partecipante (o creatore) di un altro match.	

Endpoint	Metodo	Attributi
/matches/:id_match	Get	Type
Descrizione	Restituisce i dati del match :id_match in formato JSON. Se il tipo specificato è fullInfo (type=fullInfo) allora vengono ritornate anche le informazioni dei campi reference del match (i due utenti e i campi). Nello specifico il campo dell'utente contro cui sta combattendo l'utente autenticato viene ritornato con tutte le caselle in forma "anonima", fatta eccezione per quelle da lui già colpite.	

Endpoint	Metodo	Attributi
/matches/:id_match/join	Put	
Descrizione	Aggiorna il match identificato dall'id specificato, aggiungendo come secondo partecipante (opponent) l'utente autenticato.	

Endpoint	Metodo	Attributi
/matches/:id_match/board	Put	La disposizione delle navi in un file JSON con la seguente struttura: { "positioning" : { "ships" : [[{"x" : Number, "y" : Number}, {"x" : Number "y" : Number}], …]}} Non è possibile inserire navi se il match di cui si fa parte non è in stato di building (ovvero se non ci sono due utenti all'interno del match). Qualora ci fossero delle irregolarità nella disposizione delle navi (navi adiacenti, troppe navi di uno stesso tipo, navi fuori dal campo, navi sovrapposte, navi troppo corte o troppo lunghe) viene ritornato un errore.
Descrizione	Modifica il match identificato dall'id specificato, creando un nuovo campo per l'utente autenticato e inserendo le navi nelle posizioni da lui scelte	

Endpoint	Metodo	Attributi
/matches/:id_match	Put	Position
Descrizione	Modifica il match “colpendo” la cella in posizione position (che avrà un formato di tipo { “position” : {“x” : Number, “y” : Number}}. La cella del campo colpita assumerà un nuovo colore in base al suo contenuto, ovvero se il colpo è finito in acqua o su una delle navi. Se una nave viene distrutta a causa di questo colpo, vengono aggiornati i colori di tutte le celle della nave. L’utente non può sparare due volte alla stessa cella. L’utente non può sparare due volte di fila, si torna con l’avversario (il primo a sparare è estratto a sorte).	

2.5. Angular: realizzazione del client web

Angular viene utilizzato per strutturare il web client dell’applicazione: ecco una descrizione dei component, dei servizi e delle routes implementate.

Component	Funzione
Index.html	File principale html, importa i file e i moduli necessari.
NavBar-Component	Il suo compito è quello di mostrare la navbar del sito. La navbar cambia a seconda che l’utente sia autenticato o meno.
App-Component	Unisce la navbar alla pagina contenente tutto il resto. Al suo interno è implementato un modal-dialog che notifica il ricevimento di un messaggio.
UserSignup-Component	Utilizzato per la registrazione dell’utente. Richiede tutti i campi che devono essere compilati: nome, cognome, username, password, conferma della password. Password e conferma della password devono essere identiche, altrimenti notifica l’utente della discrepanza tra le due.
UserLogin-Component	Utilizzato per l’accesso all’applicazione da parte di un utente. Richiede l’inserimento del nome utente e della relativa password. Qualora vi fossero dati mancanti o password errata, ritorna il relativo errore, notificando l’utente. Attraverso il pulsante “Registrati” vi è la possibilità di essere reindirizzati alla pagina di registrazione di un nuovo utente.
UserInfo-Component	Utilizzato per visualizzare i dati di un profilo utente. Se il profilo visualizzato è lo stesso dell’utente autenticato, vi è la possibilità di modificarne anche i dati.

Component	Funzione
Match-Component	Utilizzato per portare avanti la partita tra i due giocatori. Il giocatore che può attaccare può selezionare la cella del campo avversario che vuole colpire. Il colore cambia se colpisce/affonda una nave oppure se il colpo va in acqua. Non è possibile sparare su una cella già colpita oppure sparare per due turni di fila. Viene visualizzato anche il proprio campo, oltre a quello dell'avversario, con rappresentati i colpi sparati finora dal nemico. Viene anche presentata una legenda che consente al giocatore di capire il significato dei colori delle caselle.
List-Matches-Component	Utilizzato per presentare all'utente una lista dei match in attesa di un giocatore. Vi è anche la possibilità di creare il proprio match. Non è possibile creare una nuova partita se si fa già parte di un'altra.
Match-Builder-Component	Consente all'utente di posizionare le navi all'interno del campo con un sistema di drag and drop. C'è inoltre la possibilità di ruotarle mediante l'apposito pulsante.
List-Chat-Component	Consente di visualizzare tutte le chat con altri utenti. Vi è la possibilità di accedere ad una chat o di cancellarla.
Chat-Component	Consente di visualizzare i messaggi della chat selezionata. Consente di inviare un nuovo messaggio.
Players-Component	Consente di vedere gli utenti registrati nel gioco. Si possono anche ricercare tramite barra di ricerca inserendo delle informazioni per restringere il campo.
NotFound-Component	Utilizzato per la visualizzazione di una Page Not Found, qualora fosse inserito un URL errato.
Scoreboard-Component	Consente di visualizzare la scoreboard degli utenti, elencandoli in ordine decrescente per numero di partite totali, vinte o perse. Vi è anche la possibilità di impostare un limite degli utenti visualizzati

I servizi implementati sono i seguenti:

Service	Funzione
SocketIO-Service	Consente una connessione al server tramite socket.io-client. Il client può quindi ricevere le informazioni dal parte del server rimanendo in ascolto sulla connessione stabilita.
User-Service	Consente di gestire le informazioni dell'utente (retireve, update, login, etc). Inoltre fornisce delle funzioni per estrarre le informazioni dal JWT.
Utilities-Service	Consente di creare gli header (authorization, cache-control e Content-Type) e i parametri della connessione. Fornisce anche una funzione per verificare l'autenticazione di un utente, attraverso il token passato in input.
Match-Service	Consente la gestione delle informazioni dei match, interfacciandosi col server. Questo riguarda l'inserimento delle navi, il mostrare i campi agli utenti in gioco, ma anche la gestione dei colpi di ciascun utente.

Di seguito le routes implementate:

Path	Component
login	User-Login.Component
user	User-Info.Component
user/:username	User-Info.Component
user/:username/matches	List-Matches.Component
signup	User-Signup.Component
players	Players.Component
match	List-Matches.Component
match/:id	Match.Component
match/:id/board	Match-Builder.Component
chats	List-Chats.Component
chats/:id	Chat.Component
scoreboard	Scoreboard.Component
**	Notfound.Component

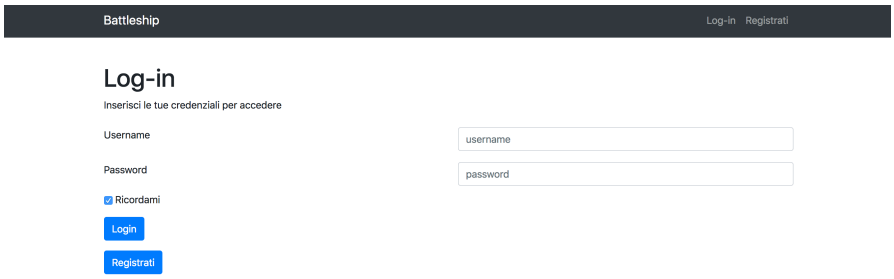
3. Funzionamento

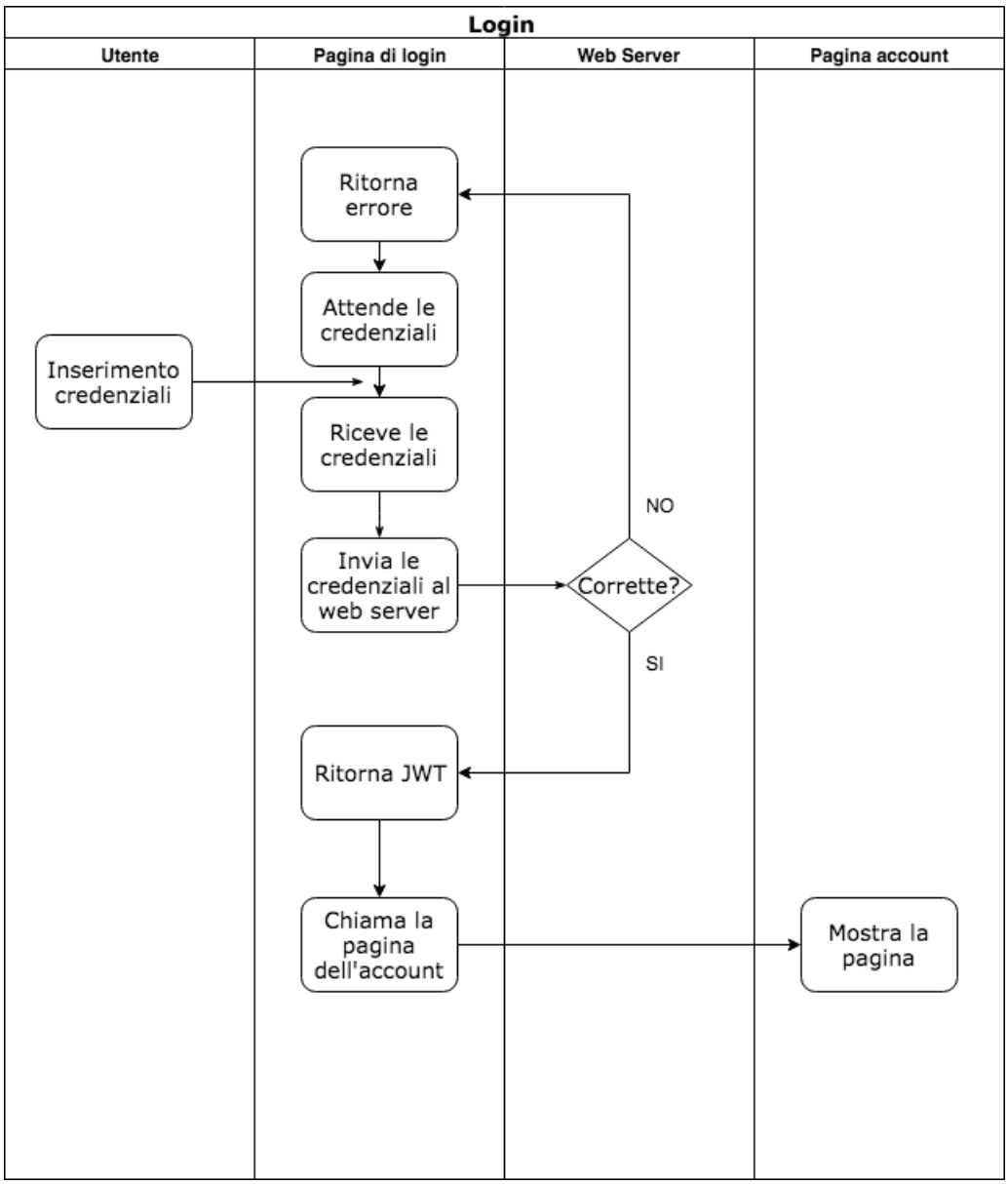
3.1. Login

Come prima cosa l'utente deve autenticarsi, se non lo ha già fatto.
In questa schermata può inserire le credenziali di accesso che, se corrette, gli garantiranno il permesso di accedere alla schermata successiva dell'applicazione (ovvero la sua pagina personale).

In caso di credenziali corrette, il server web ritorna un JWT contenente le informazioni dell'utente: gli consentirà di accedere alle altre pagine dell'applicazione.

In caso di credenziali errate, l'utente non potrà ovviamente accedere e gli verrà notificato l'errore.





Flow diagram per il login di un utente

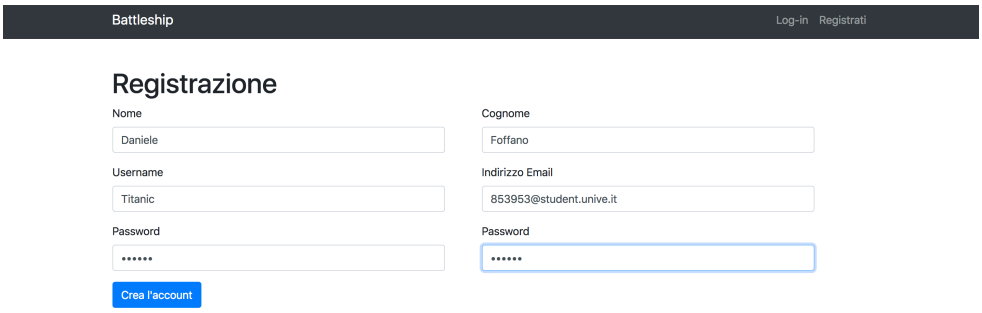
3.2. Registrazione

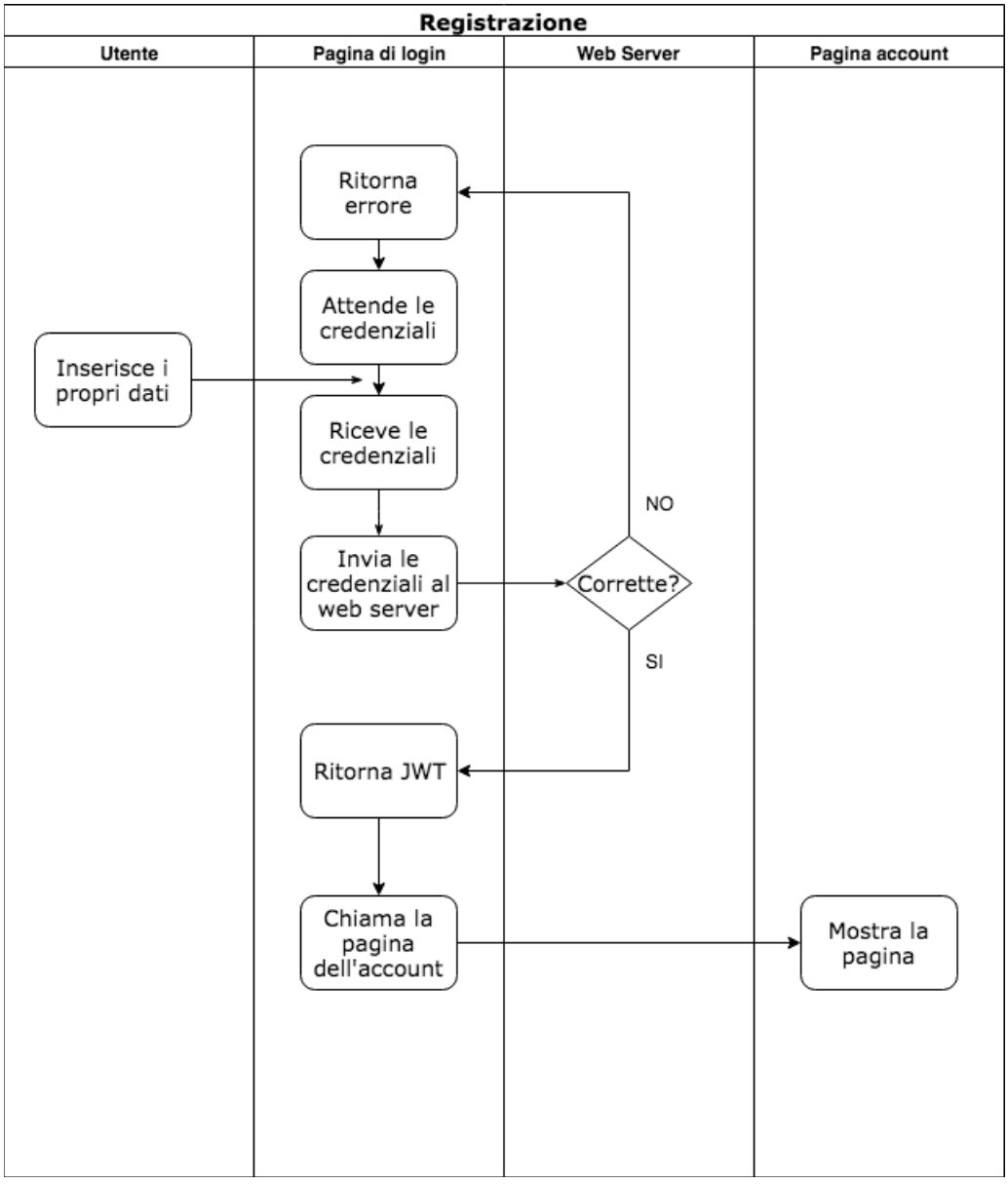
Qualora l'utente non fosse già registrato, nella pagina di login è possibile premere sul pulsante "registrati" per accedere alla pagina in cui si potrà creare un nuovo account.

Verranno richieste tutte le informazioni necessarie: nome, cognome, indirizzo email, username, password e conferma della password.

Se il nome utente o la mail sono già state utilizzate, l'account non verrà creato e sarà ritornato un errore dal web server per avvisare l'utente.

Anche qualora le due password non dovessero coincidere l'account non verrà registrato e l'utente sarà avvisato della discrepanza.





Flow diagram per la registrazione di un utente

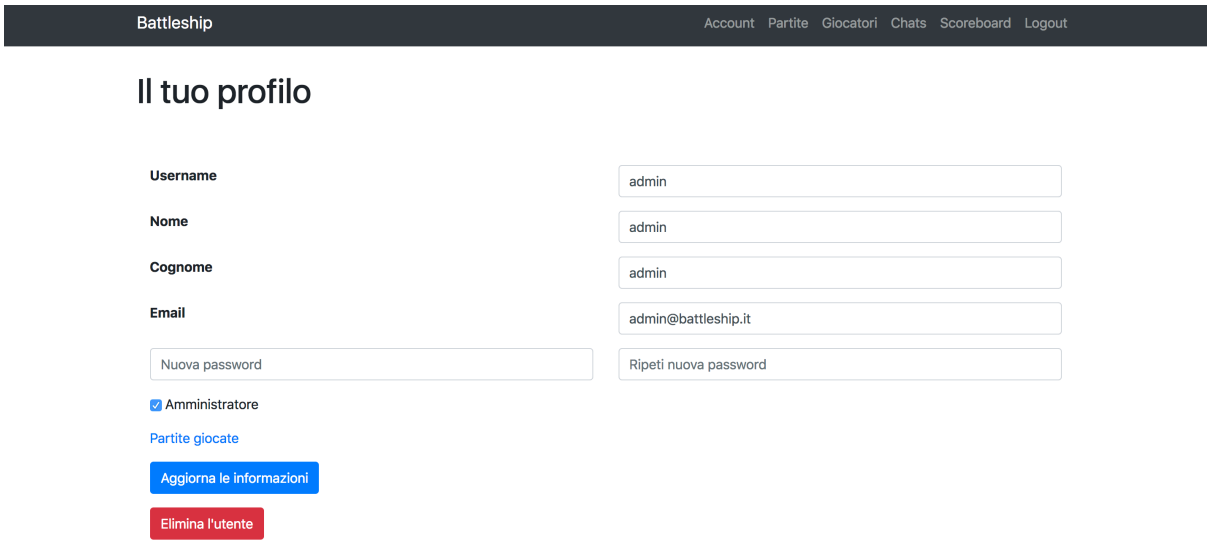
3.3. Pagina personale

Nella pagina personale sarà possibile visualizzare i dati del proprio account, e sarà possibile eliminarlo o modificarne i dati.

L'operazione di modifica può essere effettuata solamente dall'account proprietario, quella di cancellazione sia dal proprietario che da un utente admin.

Vi è anche la possibilità di nominare un utente admin (o verificare che già lo sia): questo privilegio spetta solamente ad un utente già admin.

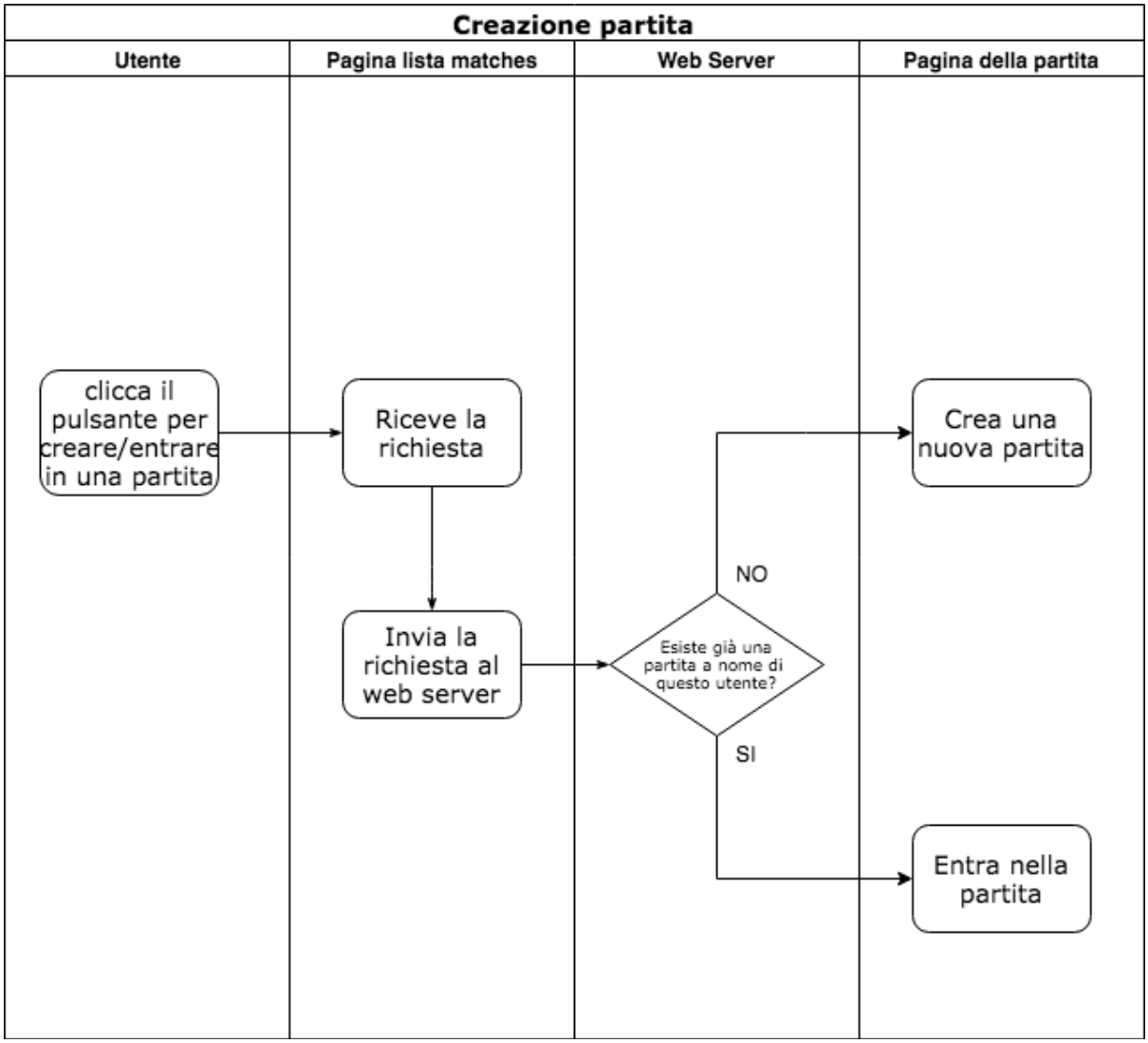
Se si visualizza la pagina di un altro utente, vi è anche la possibilità di iniziare una chat con lui (o riprenderne una già avviata in passato).



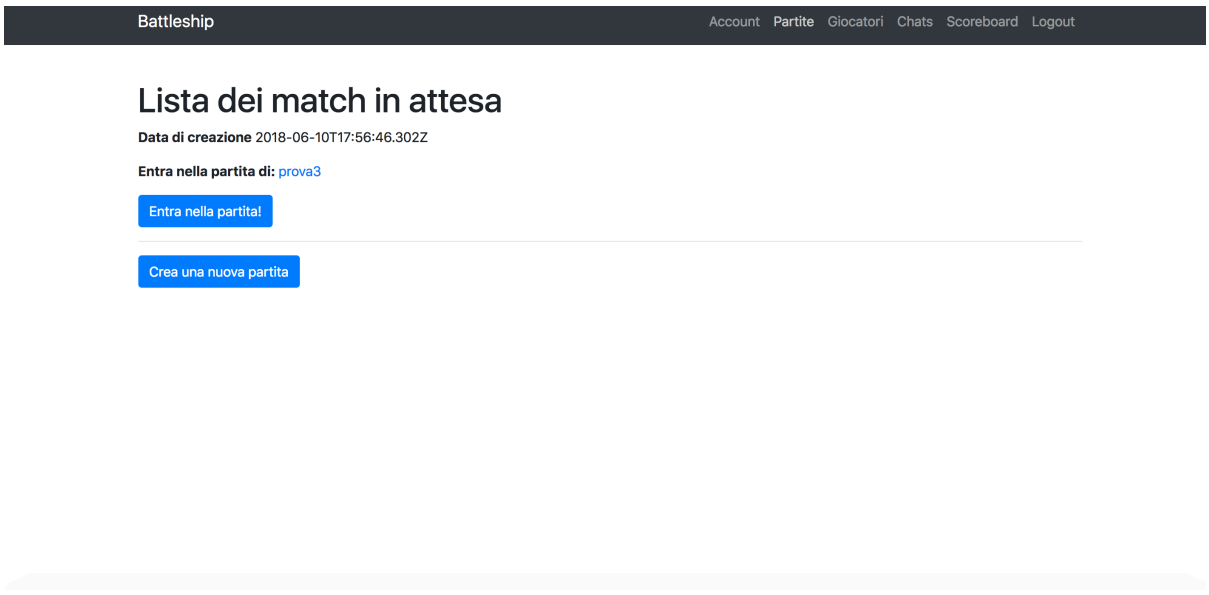
3.4. Pagina per visualizzare i match in attesa

Andando nella pagina contenente la lista dei match in attesa, sarà possibile visualizzare i match creati dagli altri utenti a cui manca un secondo giocatore. Sempre in questa pagina vi sarà infatti la possibilità di prendere parte ad una di queste partite cliccando sull'apposito pulsante.
Un'altra opportunità sarà quella di creare la propria partita, rimanendo in attesa che si aggiunga qualcuno ad essa.

Si può creare/aggiungersi ad una partita soltanto se non si hanno già partite in sospeso, sia come owner che come utente ospite.



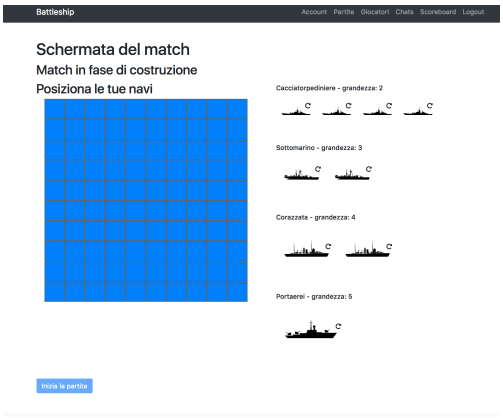
Flow diagram per la creazione di una partita

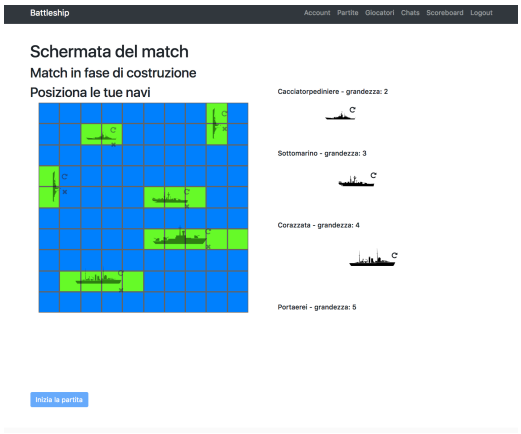


3.5. Pagina per il posizionamento delle navi

Una volta che entrambi i giocatori sono entrati in una partita, il match passa nello stato di building, ed è il momento per entrambi di disporre sul campo le proprie navi. Tramite una semplice operazione di drag and drop si può posizionare la flotta sul campo, avendo anche la possibilità di girare qualsiasi nave in verticale o orizzontale.

Una volta posizionata la nave può essere rimossa e riposizionata.
Le navi non possono essere adiacenti le une alle altre (tantomeno sovrapposte), possono comunque essere adiacenti ai margini del campo.

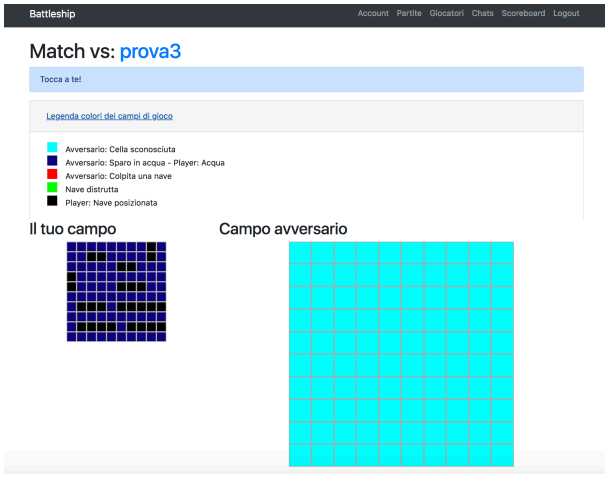


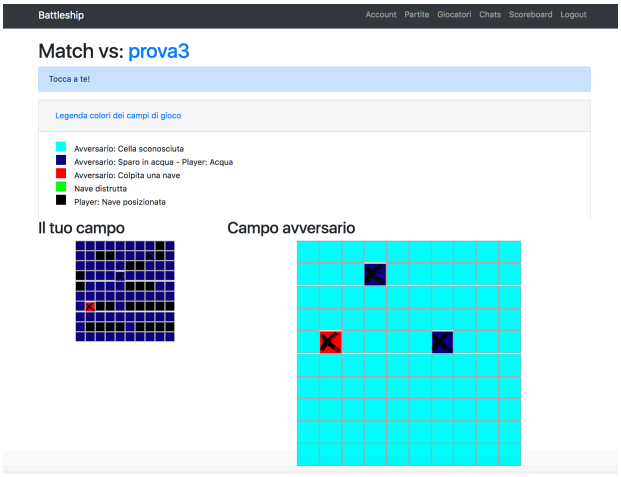


3.6. Pagina per il combattimento

Quando entrambi i giocatori avranno confermato il loro posizionamento il match passerà nello stato Active, ovvero allo scontro vero e proprio.
In questa pagina ciascun utente potrà visualizzare il proprio campo, con le proprie navi posizionate e con segnati i colpi inviati dall'avversario, e il campo avversario, dove saranno rappresentati soltanto i propri colpi sparati ed il loro risultato (acqua o nave).
Non è possibile sparare su una cella già colpita.
Non è nemmeno possibile attaccare per più turni di fila: il gioco è a turni, e il giocatore iniziale viene estratto a sorte.

La partita si conclude quando uno dei due giocatori riesce a distruggere tutte le navi avversarie.

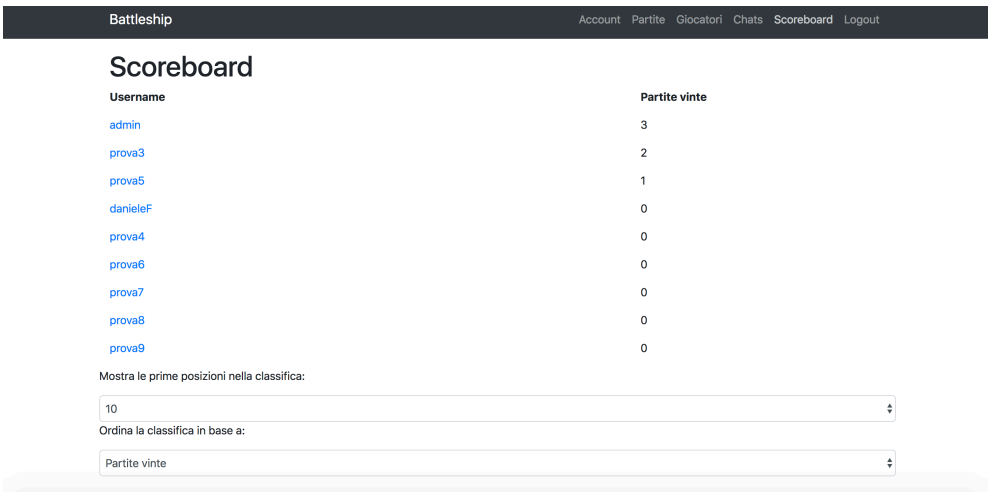




3.7. Pagina per visualizzare la scoreboard

In questa pagina sarà possibile visualizzare la classifica degli utenti.
Questi saranno ordinati in modo decrescente, sulla base di una delle tre opzioni a scelta dell'utente: partite effettuate, vinte o perse.

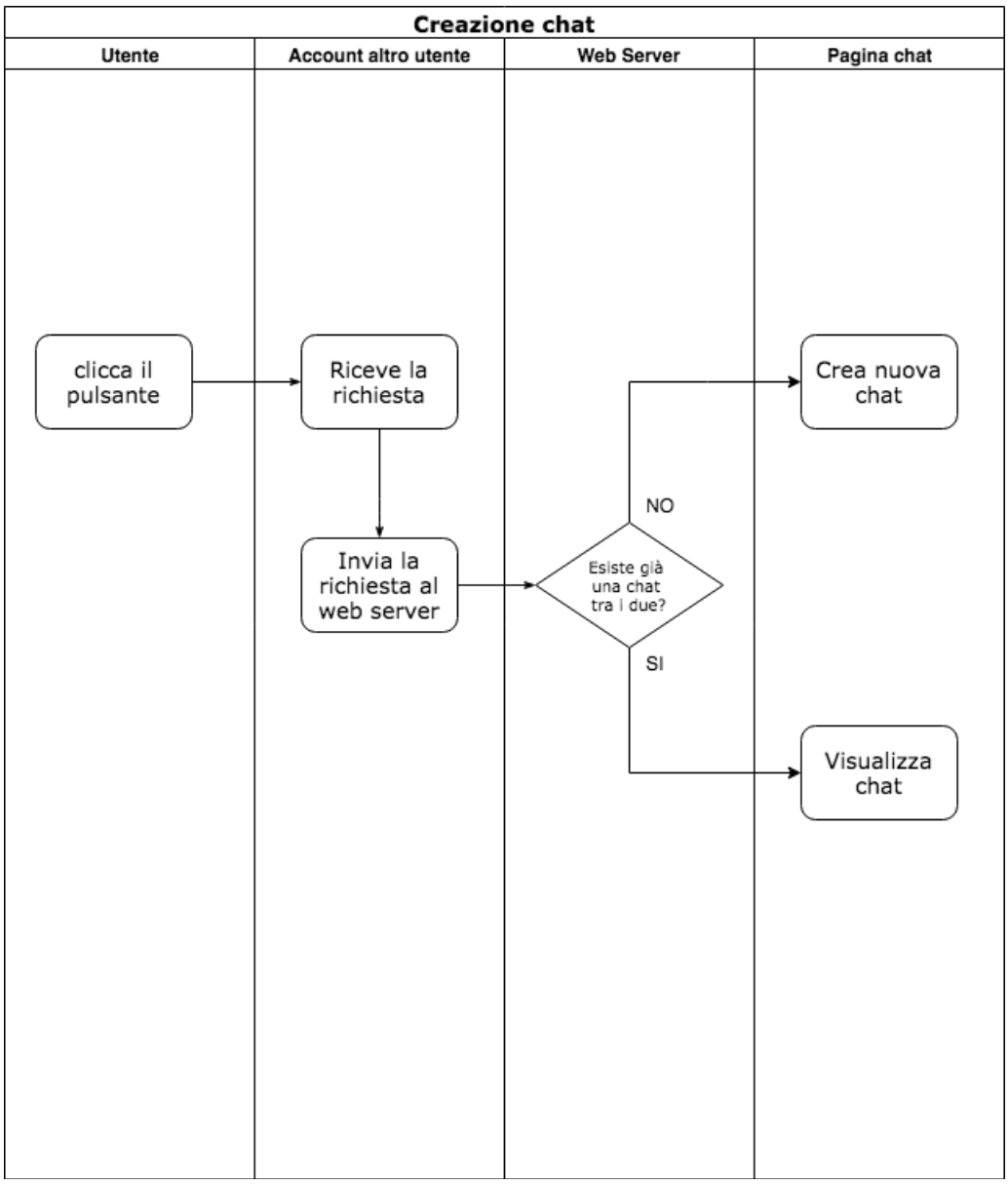
L'utente potrà inoltre scegliere il limite di utenti visualizzati.



3.8. Pagina per la visualizzazione delle proprie chat

In questa pagina sarà possibile visualizzare una lista delle proprie chat con altri utenti.

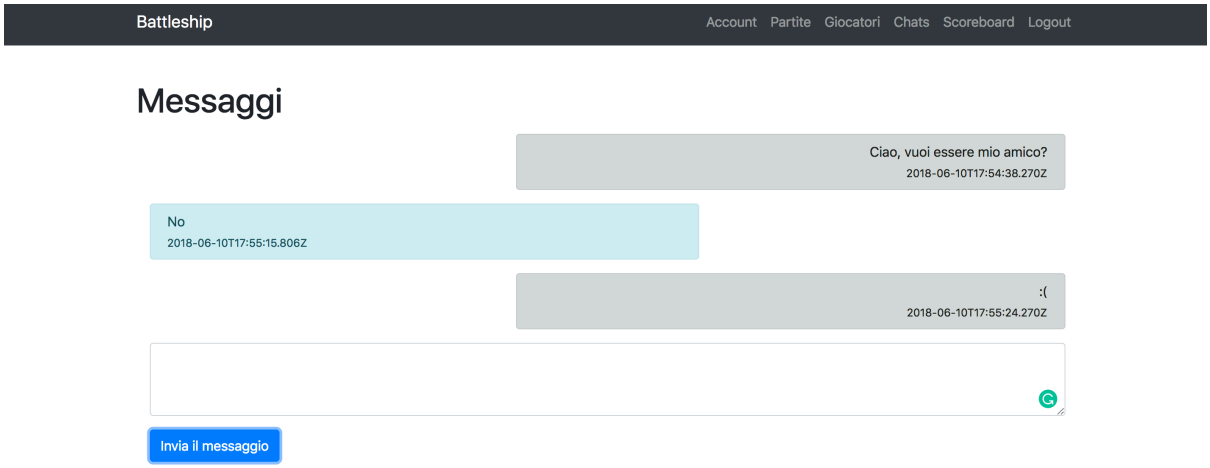
Sarà possibile aprirne una per inviare un nuovo messaggio, oppure cancellarla attraverso l'apposito pulsante.



Flow diagram per la creazione di una chat

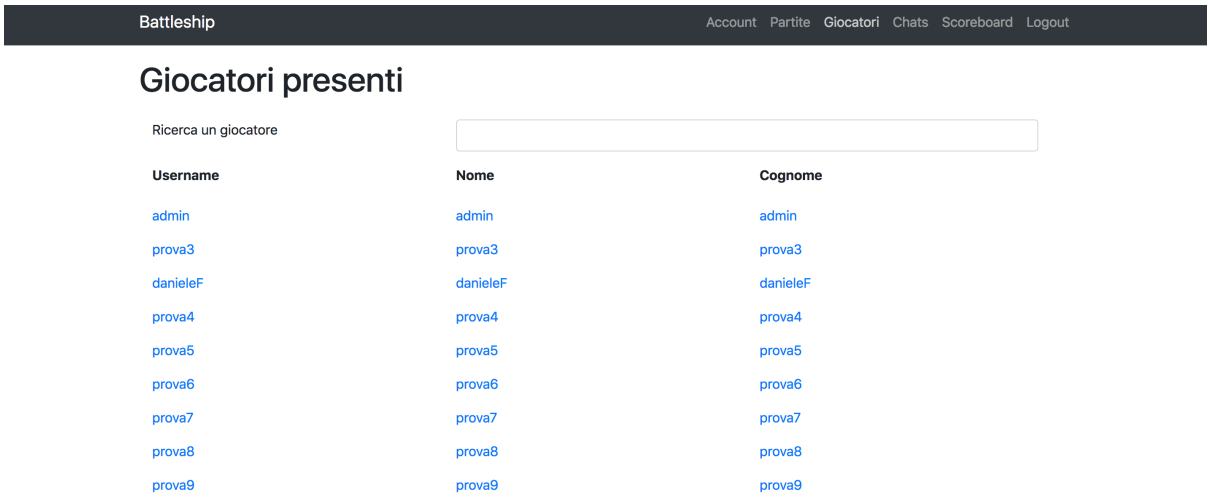
3.10. Pagina per l'invio di un messaggio

Una volta aperta/creata una conversazione sarà possibile inviare un messaggio all'altro utente, digitandolo e premendo invio.



3.11. Pagina per la ricerca di un utente

In questa pagina sarà possibile ricercare un determinato utente inserendo una parola chiave nell'apposita barra di ricerca.



4. Possibili aggiunte future

In futuro a questo progetto si potrebbe aggiungere la possibilità per un utente di giocare più partite contemporaneamente.

Una seconda aggiunta interessante potrebbe essere la possibilità di creare chat tra più di due utenti, creando dei gruppi.

Infine, utile per chi aspira a diventare un campione di battaglia navale e vuole imparare le tecniche dei migliori, sarebbe interessante l'aggiunta di una "match history" di ciascun utente, in modo che si potessero vedere le disposizioni dei campi nelle partite in cui un determinato utente ha vinto o ha perso.

5. Note finali

All'interno dell'applicazione è presente un piccolo bug quando è viene sparato un colpo all'avversario: a causa della X che compare sulla casella colpita, le celle del campo si discostano lievemente tra loro. È un problema facilmente affrontabile, ma che purtroppo a causa della mancanza di tempo non siamo riusciti a sistemare.
Non influisce tutta via sulle meccaniche di gioco, né a nostro parere rovina in maniera totale l'esperienza dell'utente: è soltanto poco gradevole visivamente