

Architettura

La versione online di questa documentazione è disponibile presso la [Wiki del repository di GitHub](#).

Modello dati

La struttura dei dati organizzati nel database è rappresentata graficamente tramite [questo schema](#).

API

Premessa: il body delle risposte degli endpoints è sempre un oggetto di tipo [infrastructure.net.HttpMessage<T>](#).

Users

- /users
 - [POST](#)
 - **Autenticazione richiesta:** no
 - **Request.body:** [application.DTOs.ISignupRequestDto](#) (le informazioni necessarie alla registrazione)
 - **Response.body:** `infrastructure.net.HttpMessage<boolean>` (l'esito della richiesta di registrazione)
 - **Descrizione:** Crea un nuovo utente con le caratteristiche specificate e restituisce l'esito della registrazione.
- /users/rankings
 - [GET](#)
 - **Autenticazione richiesta:** no
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IUserRanking[]>` (l'elenco dei rankings)
 - **Descrizione:** Restituisce la classifica dei 10 giocatori migliori, valutati in base a percentuale di vittoria e numero di partite vinte.
- /users/:userId
 - [GET](#)
 - **Autenticazione richiesta:** no
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IUserProfile>` (le informazioni del profilo)
 - **Descrizione:** Restituisce tutte le informazioni sull'utente: dati del profilo e valori di ranking.
 - [DELETE](#)

- **Autenticazione richiesta:** si
- **Response.body:** infrastructure.net.HttpMessage<boolean> (l'esito della richiesta di eliminazione account)
- **Eventi:**
 - UserDeleted
 - **Addressee:** utente il cui account è stato eliminato
 - PendingMatchesChanged
 - **Addressee:** pubblico
 - MatchCanceled
 - **Addressee:** eventuale utente che era in partita con l'utente eliminato
- **Descrizione:** Elimina l'utente specificato. Questa operazione può essere eseguito solo dall'amministratore. Se presente, la partita in attesa aperta dal giocatore viene chiusa o, nel caso l'utente fosse in partita, questa viene annullata e non conteggiata nello storico delle partite. Le partite già concluse rimarranno invece nello storico.
Tutti messaggi scambiati con l'utente eliminato vengono eliminati.
- /users/:userId/powers
 - [GET](#)
 - **Autenticazione richiesta:** si
 - **Response.body:** infrastructure.net.HttpMessage<[application.DTOs.IUserPowers](#)> (le azioni di moderazione disponibili)
 - **Descrizione:** Restituisce un oggetto che descrive quali azioni di moderazione l'utente è autorizzato ad eseguire.
- /users/:userId/ban
 - [POST](#)
 - **Autenticazione richiesta:** si
 - **Request.body:** [application.DTOs.IUserBanRequest](#) (trasporta la durata del ban)
 - **Response.body:** infrastructure.net.HttpMessage<Date> (la data di scadenza del ban)
 - **Eventi:**
 - UserBanned
 - **Addressee:** utente il cui account è stato bannato
 - PendingMatchesChanged
 - **Addressee:** pubblico
 - MatchCanceled
 - **Addressee:** eventuale utente che era in partita con l'utente bannato
 - **Descrizione:** Modifica o rimuove la data di scadenza del ban per l'utente specificato. Questa operazione può essere eseguita solo dagli utenti con grado [infrastructure.identity.Roles.Moderator](#) o superiore.
Solo gli utenti con grado [infrastructure.identity.Roles.Administrator](#) possono applicare ban permanenti.
Solo gli utenti con grado [infrastructure.identity.Roles.Administrator](#) possono applicare ban ad altri utenti con grado [infrastructure.identity.Roles.Administrator](#).

- `/users/:userId/role`
 - [POST](#)
 - **Autenticazione richiesta:** si
 - **Request.body:** [application.DTOs.IRoleAssignmentRequestDto](#) (trasporta il nuovo ruolo da assegnare)
 - **Response.body:** `infrastructure.net.HttpMessage<Infrastructure.Identity.UserRole>` (il nuovo ruolo assegnato)
 - **Eventi:**
 - `UserRoleUpdated`
 - **Addressee:** utente il cui ruolo è stato cambiato
 - **Body:** [infrastructure.identity.UserRole](#) (il nuovo ruolo assegnato)
 - **Descrizione:** Modifica il ruolo dell'utente specificato. Questa operazione può essere eseguita solo dagli utenti con grado [infrastructure.identity.Roles.Administrator](#).
- `/users/:userId/matchHistory`
 - [GET](#)
 - **Autenticazione richiesta:** no
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IEndedMatchSummaryDto\[\]>` (la lista delle partite)
 - **Descrizione:** Restituisce la lista delle ultime 20 partite giocate dall'utente specificato.

Auth

- `/auth/login`
 - [POST](#)
 - **Autenticazione richiesta:** no
 - **Response.body:** `infrastructure.net.HttpMessage<string>` (JWT di accesso)
 - **Descrizione:** Se l'utente identificato dalle credenziali fornite non è bannato fornisce il JWT di accesso, altrimenti restituisce `null`.
Le tecniche utilizzate per l'invio delle credenziali e la validazione delle richieste sono descritte [qui](#).

Game

- `/game/playables`
 - [GET](#)
 - **Autenticazione richiesta:** si
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IPlayablesDto>`

- **Descrizione:** Fornisce le informazioni relative alle partite in attesa disponibili, alla possibilità del giocatore di crearne una, gli ID delle eventuali partite in attesa creata dall'utente e partita a cui sta partecipando
- /game/pendingMatches
 - [POST](#)
 - **Autenticazione richiesta:** sì
 - **Response.body:** infrastructure.net.HttpMessage<string> (l'ID della partita in attesa creata)
 - **Eventi:**
 - PendingMatchesChanged
 - **Addressee:** pubblico
 - **Descrizione:** Crea una nuova partita in attesa assegnata all'utente identificato tramite il JWT contenuto nella richiesta.
- /game/pendingMatches/:pendingMatchId
 - [DELETE](#)
 - **Autenticazione richiesta:** sì
 - **Response.body:** infrastructure.net.HttpMessage<boolean> (l'esito dell'operazione)
 - **Eventi:**
 - PendingMatchesChanged
 - **Addressee:** pubblico
 - **Descrizione:** Elimina la partita con l'ID specificato creata dall'utente identificato dal JWT contenuto nella richiesta.
 - [PUT](#)
 - **Autenticazione richiesta:** sì
 - **Response.body:** infrastructure.net.HttpMessage<string> (l'ID della partita creata)
 - **Eventi:**
 - PendingMatchesChanged
 - **Addressee:** pubblico
 - PendingMatchJoined
 - **Addressee:** giocatore che ha creato la partita in attesa
 - **Body:** string (ID della partita creata)
 - **Descrizione:** L'utente identificato dal JWT contenuto nella richiesta si unisce alla partita in attesa specificata che diventa quindi una partita vera e propria.
- /game/matches/:matchId
 - [GET](#)
 - **Autenticazione richiesta:** sì
 - **Response.body:** infrastructure.net.HttpMessage<[application.DTOs.IMatchDto](#)> (informazioni sulla partita)
 - **Descrizione:** Restituisce tutte le informazioni relative alla partita a cui l'utente loggato sta partecipando, e solamente le informazioni visibili dal punto di vista di quel giocatore.
- /game/matches/:matchId/config
 - [POST](#)
 - **Autenticazione richiesta:** sì

- **Request.body:** [infrastructure.game.IShipPlacement](#) (posizione e orientamento di ogni singola nave)
 - **Response.body:** `infrastructure.net.HttpMessage<boolean>` (l'esito della configurazione)
 - **Descrizione:** Invia l'elenco dei posizionamenti e degli orientamenti di ogni singola nave, e restituisce l'esito della configurazione.
- `/game/matches/:matchId/singleShot`
 - [POST](#)
 - **Autenticazione richiesta:** sì
 - **Request.body:** [infrastructure.game.ISingleShotMatchAction](#) (informazioni sull'attacco)
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IAttackResultDto>` (il risultato dell'attacco)
 - **Descrizione:** Invia la coordinata da attaccare e restituisce un oggetto che rappresenta l'esito dell'attacco, contenente le celle nemiche coinvolte e i relativi cambiamenti.

Chat

- `/chat`
 - [GET](#)
 - **Autenticazione richiesta:** sì
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IChatDto\[\]>` (la lista delle chat)
 - **Descrizione:** Restituisce la lista degli utenti con cui l'utente specificato può parlare, e per gli utenti con cui ha già parlato il relativo storico dei messaggi scambiati.
- `/chat/:userId`
 - [GET](#)
 - **Autenticazione richiesta:** sì
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IChatMessageDto>` (la lista dei messaggi)
 - **Descrizione:** Restituisce la lista dei messaggi scambiati con l'utente specificato.
 - [POST](#)
 - **Autenticazione richiesta:** sì
 - **Request.body:** [application.DTOs.INewMessage](#) (il messaggio da inviare)
 - **Response.body:** `infrastructure.net.HttpMessage<application.DTOs.IChatMessageDto\[\]>` (il messaggio inviato e il relativo timestamp)
 - **Eventi:**
 - `YouGotANewMessage`
 - **Addressee:** giocatore a cui è indirizzato il messaggio
 - **Body:** [application.DTOs.IChatMessageDto](#)
 - **Descrizione:** Invia un messaggio all'utente specificato e, in caso la consegna vada a buon fine, restituisce il messaggio inviato con il timestamp della consegna.

Autenticazione

L'autenticazione si basa su [Basic Access Authentication](#) al momento del login, tramite cui si ottiene un [JSON Web Token](#) che viene utilizzato per validare le richieste successive.

Le tecniche utilizzate non si occupano della protezione delle credenziali, che è invece delegata al protocollo HTTPS.

Basic Access Authentication

La fase di login si effettua inviando all'endpoint [/auth/login](#) una richiesta HTTP di tipo POST contenente l'header `Authorization` con le credenziali di accesso in formato [Basic](#): la stringa `Basic <username>:<password>` codificata in base 64. [passport-http.BasicStrategy](#) si occupa del parsing del campo `Authorization`, estraendo username e password per poi validarli tramite una [funzione dedicata](#).

Questa funzione può passare ad express un oggetto come errore oppure, nel caso le credenziali siano valide, confermare la validazione fornendo un oggetto contenente i dati dell'utente che si vuole mettere a disposizione della funzione dell'endpoint, la quale potrà accedervi tramite il campo `user` dell'oggetto `Request` di express.

Nel caso le credenziali siano valide l'endpoint restituisce un JWT generato utilizzando [jsonwebtoken](#), valido per 7 giorni, contenente il payload [UserJWTPayload](#).

JSON Web Token

Il JWT viene salvato localmente dal client, per essere poi inserito in tutte le richieste che richiedono l'autenticazione.

Il token viene inviato inserendo nel campo `Authorization` della richiesta una stringa in formato: `Bearer <token>`.

La validazione del token è effettuata tramite [express-jwt](#), un middleware che si occupa di effettuare il parsing del valore nell'header `Authorization`, decodificare il token, estrarne il payload e metterlo a disposizione della funzione dell'endpoint tramite il campo `user` dell'oggetto `Request`.

`express-jwt` mette a disposizione la possibilità di specificare una [funzione per la revoca del token](#), utile quando il server ha necessità di invalidare il token prima della sua scadenza. Questa funzionalità viene utilizzata per invalidare il token degli utenti bannati, i quali riceveranno una risposta con codice `401` pur avendo fornito un token valido.

WebApp

Registrazione

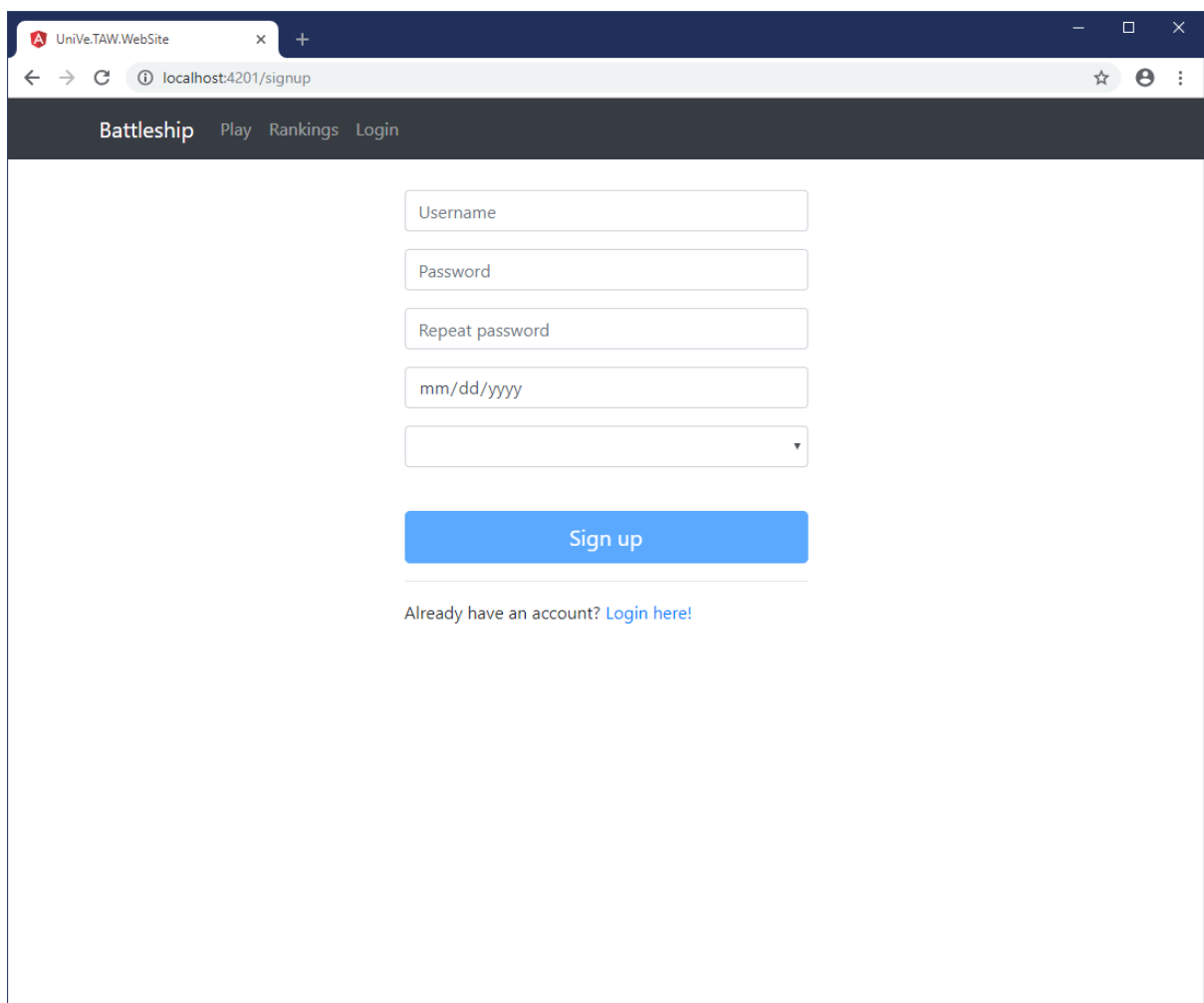
La pagina di registrazione è mappata sulla route “/signup” e visualizza il component “app/ui/identity/signup/signup.component.ts” il quale mette a disposizione il form per la registrazione, i cui campi sono:

- Username (obbligatorio)
- Password (obbligatorio)
- Ripeti password (obbligatorio)
- Data di nascita (facoltativo)
- Nazione (facoltativo)

Tramite il servizio “app/services/identity.service.ts” effettua la chiamata a “/users:POST” per eseguire la registrazione.

La richiesta di registrazione può essere inviata solo se tutti i campi obbligatori sono stati compilati e se le password inserite nei campi Password e Ripeti Password corrispondono.

Pagina di registrazione



The screenshot shows a web browser window with the address bar displaying 'localhost:4201/signup'. The page has a dark blue header with the text 'BattleShip' and navigation links 'Play', 'Rankings', and 'Login'. The main content area is white and contains a registration form with the following fields: 'Username', 'Password', 'Repeat password', a date field with the placeholder 'mm/dd/yyyy', and a dropdown menu for selecting a country. Below the form is a blue 'Sign up' button. At the bottom of the form, there is a link that says 'Already have an account? Login here!'.

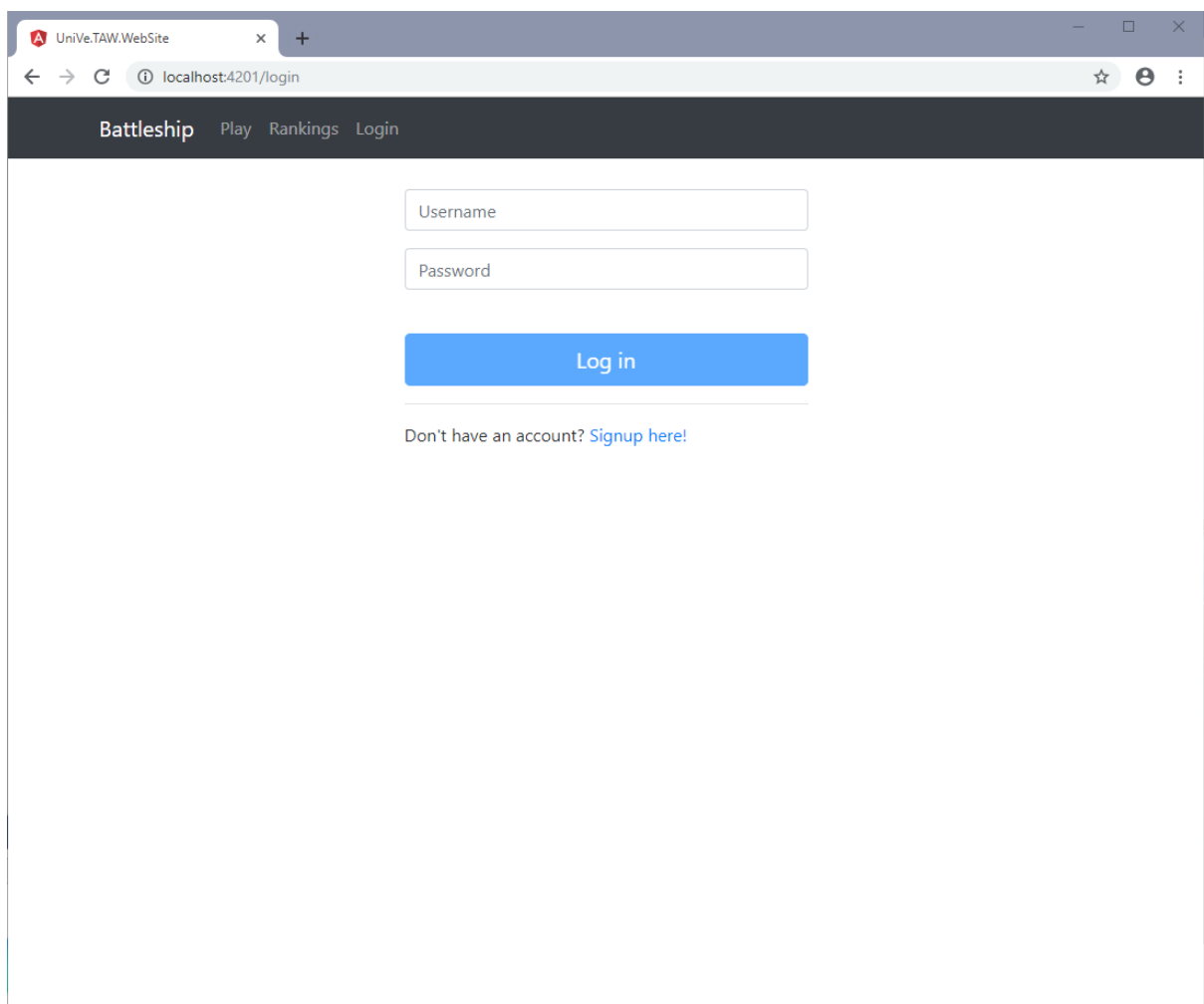
Login

La pagina di login è mappata sulla route “/login” e visualizza il component “app/ui/identity/login/login.component.ts”, il quale mette a disposizione il form per l'autenticazione, i cui campi sono:

- Username
- Password

Tramite il servizio “app/services/identity.service.ts” effettua la [procedura di autenticazione](#) per ottenere il JWT necessario alla validazione delle richieste successive.

Pagina di login



The screenshot shows a web browser window with the address bar displaying 'localhost:4201/login'. The page has a dark header with the text 'Battleship' and navigation links 'Play', 'Rankings', and 'Login'. The main content area features a login form with two input fields: 'Username' and 'Password'. Below these fields is a blue 'Log in' button. At the bottom of the form, there is a link that says 'Don't have an account? Signup here!'.

Match Finder

Questa pagina fornisce lo strumento per la ricerca e creazione delle partite.

È mappata sulla route “/match-finder”, in cui viene caricato il component “app/ui/game/match-finder/match-finder.component.ts”.

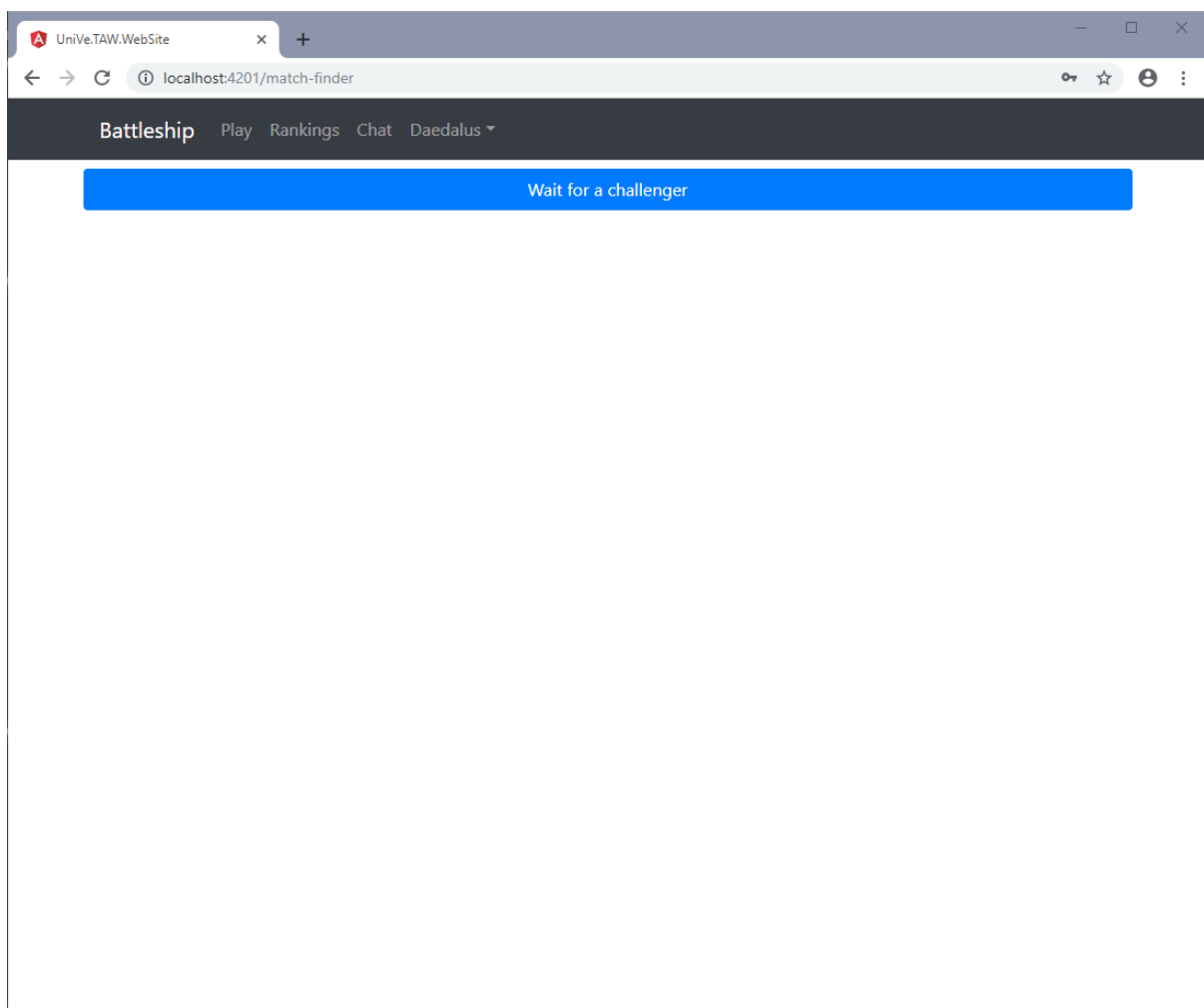
È possibile unirsi alle partite "in attesa" aperte da altri utenti tramite la lista delle partite disponibili, oppure aprirne una propria ed attendere che qualcuno raccolga la sfida.

Nel caso non si voglia più attendere è sempre possibile chiudere la propria partita in attesa e unirsi ad un'altra partita in attesa.

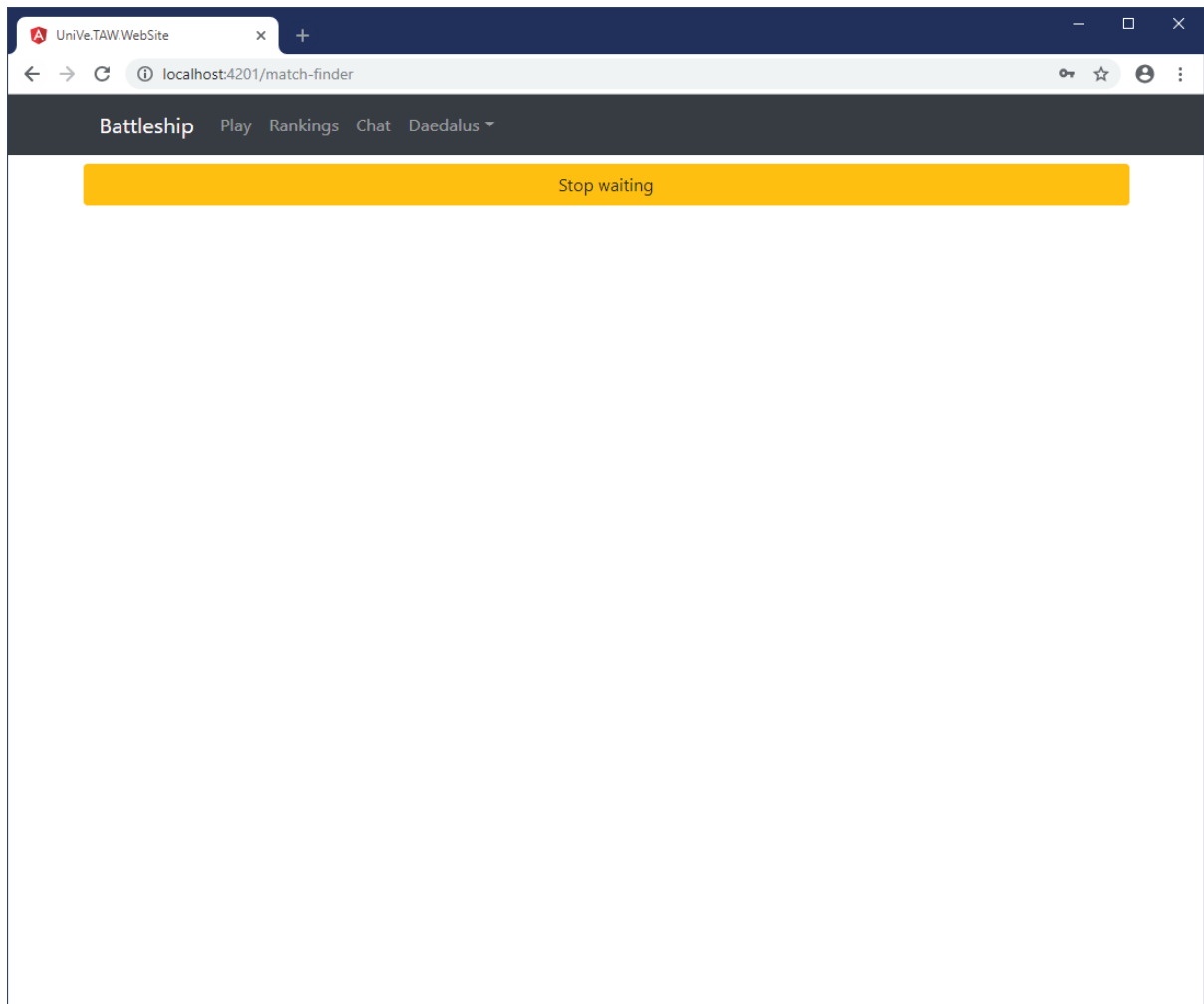
Quando ci si unisce ad un partita in attesa si viene reindirizzati alla pagina Match ad essa relativa.

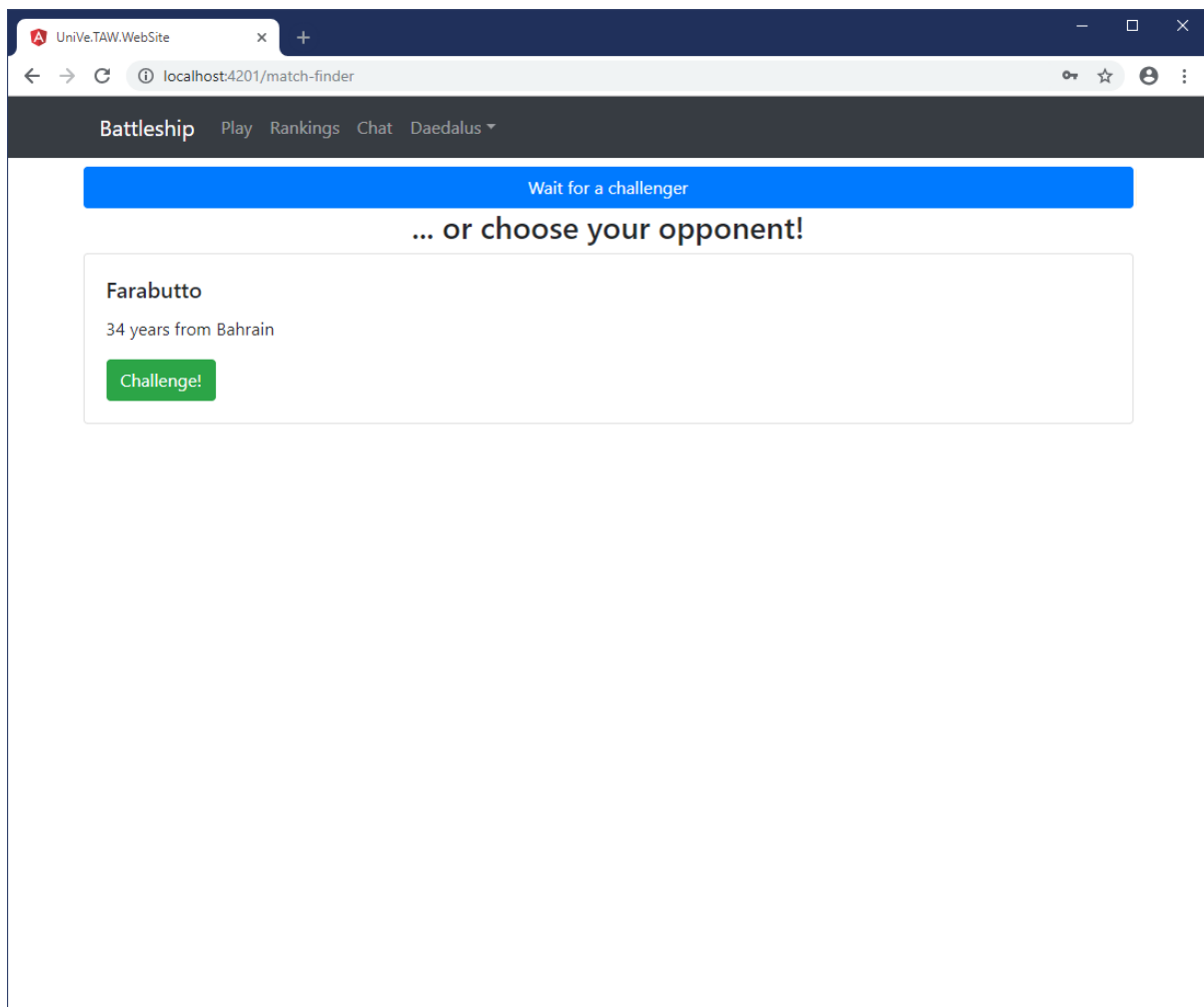
Il reindirizzamento avviene anche nel caso qualcuno si unisca alla partita in attesa creata dall'utente corrente mentre egli si trova nella pagina Match Finder.

Pagina del match finder



Match finder - In attesa





Match

La pagina "Match" visualizza i controlli per la gestione di tutte le fasi di una partita. È mappata sulla route `/match/:matchId` e visualizza il component `app/ui/game/match/match.component.ts`, che a sua volta utilizza i seguenti components:

- `app/ui/game/match/match/fleet-configurator.component.ts`
- `app/ui/game/match/match/own-turn-controller.component.ts`
- `app/ui/game/match/match/enemy-turn-controller.component.ts`
- `app/ui/game/match/match/match-chat.component.ts`

Fleet Configurator

Questo component controlla la prima fase della partita: la configurazione del proprio schieramento.

Mette a disposizione i tasti "Randomize" e "Play", i quali permettono rispettivamente di generare un nuovo schieramento randomizzato e di confermarlo inviandolo al server.

Una volta che entrambi hanno terminato la fase di schieramento, il component viene chiuso e il controllo torna al parent component (“match.component.ts”) che visualizza i campi dei giocatori e la chat.

Own Field Controller & Enemy Field Controller

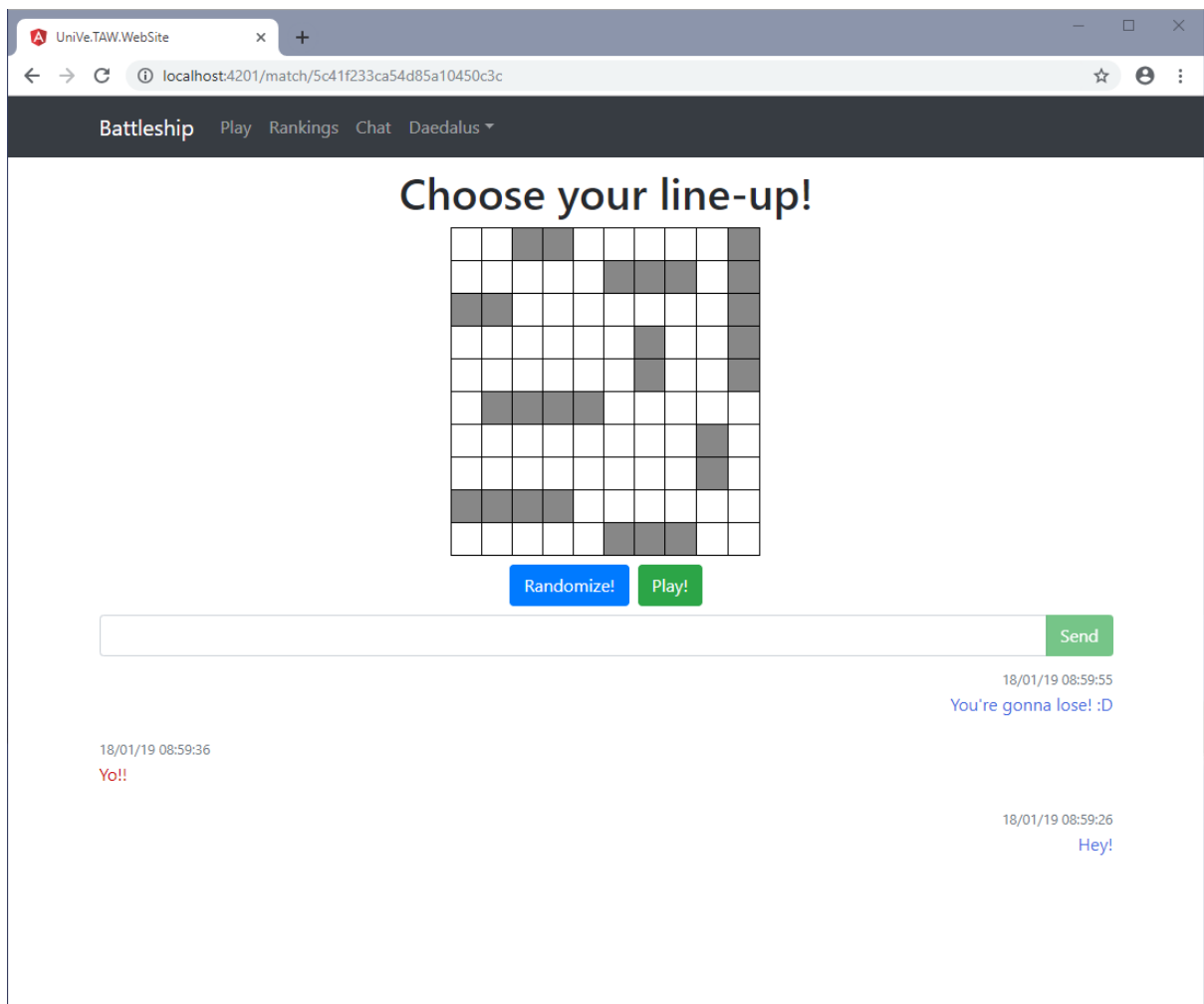
Questi due componenti visualizzano le griglie dei campi battaglia dei due giocatori.

- “app/ui/game/match/match/own-field-controller.component.ts” rappresenta il campo del giocatore loggato, visualizza le proprie navi e le coordinate colpite dal nemico
- “app/ui/game/match/match/enemy-field-controller.component.ts” rappresenta visualizza le navi nemiche, i punti già colpiti e, durante il proprio turno, permette di sferrare un attacco tramite click sulla coordinata che si vuole colpire.

Il nome dell'avversario inoltre è un link alla pagina del profilo del giocatore.

Match chat

La pagina del match visualizza anche la chat fra i 2 utenti in partita. La chat elenca tutti i messaggi scambiati in passato e permette di comunicare durante il match.



Pagina del match - In attesa che l'avversario schieri la sua flotta

UniVe.TAW.WebSite

localhost:4201/match/5c41f233ca54d85a10450c3c

BattleshipPlayRankingsChatDaedalus

Waiting for Farabutto to line-up ...

Send

18/01/19 08:59:55
You're gonna lose! :D

18/01/19 08:59:36
Yo!!

18/01/19 08:59:26
Hey!

UniVe.TAW.WebSite

localhost:4201/match/5c41f233ca54d85a10450c3c

BattleshipPlayRankingsChatDaedalus

Daedalus

Farabutto

Send

18/01/19 08:59:36

Yo!!

18/01/19 08:59:55

You're gonna lose! :D

18/01/19 08:59:26

Hey!

UniVe.TAW.WebSite

localhost:4201/match/5c41f233ca54d85a10450c3c

BattleshipPlayRankingsChatDaedalus

YOU LOST! [Play again!](#)

Daedalus

Farabutto

Send

18/01/19 08:59:36

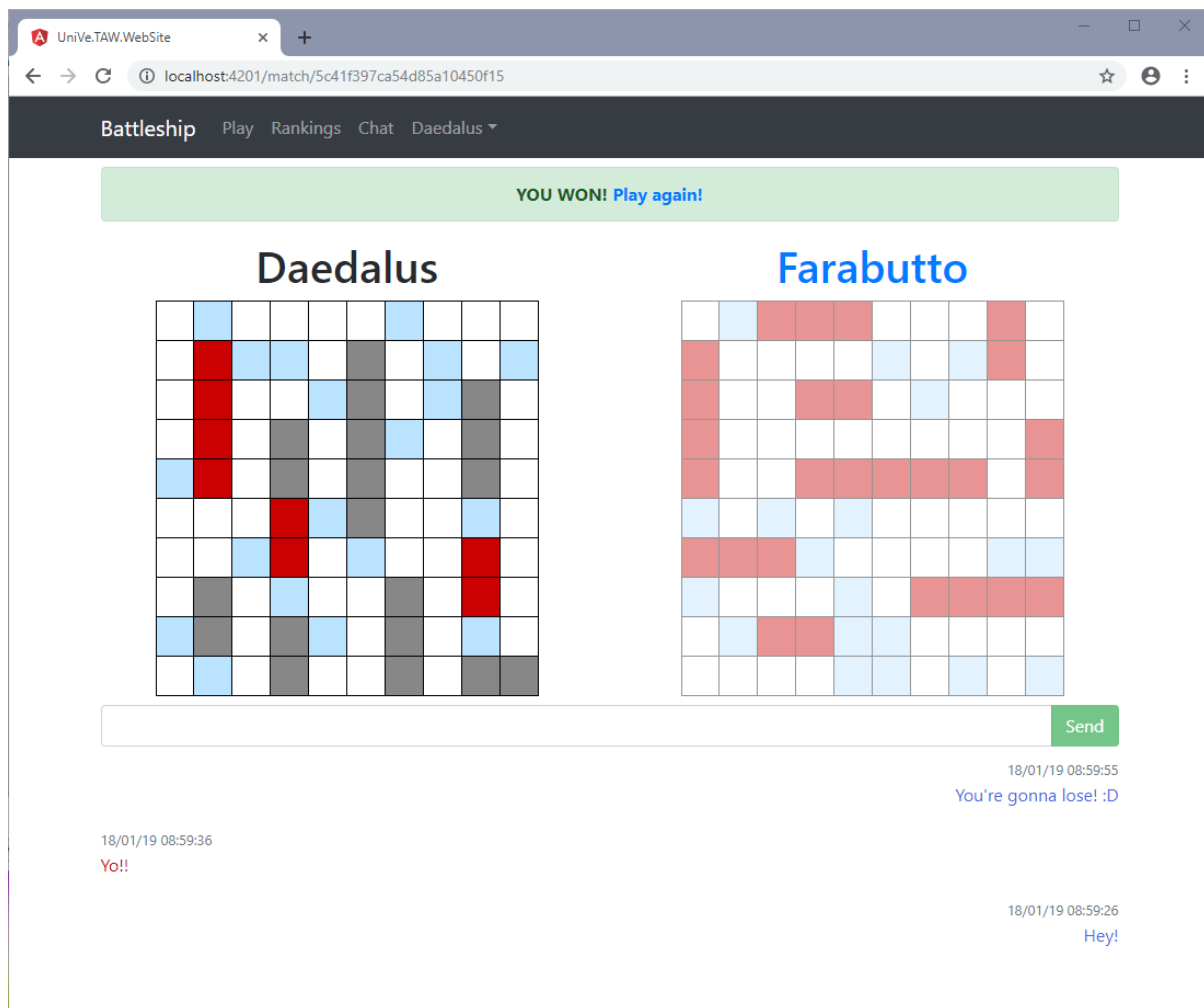
Yo!!

18/01/19 08:59:55

You're gonna lose! :D

18/01/19 08:59:26

Hey!

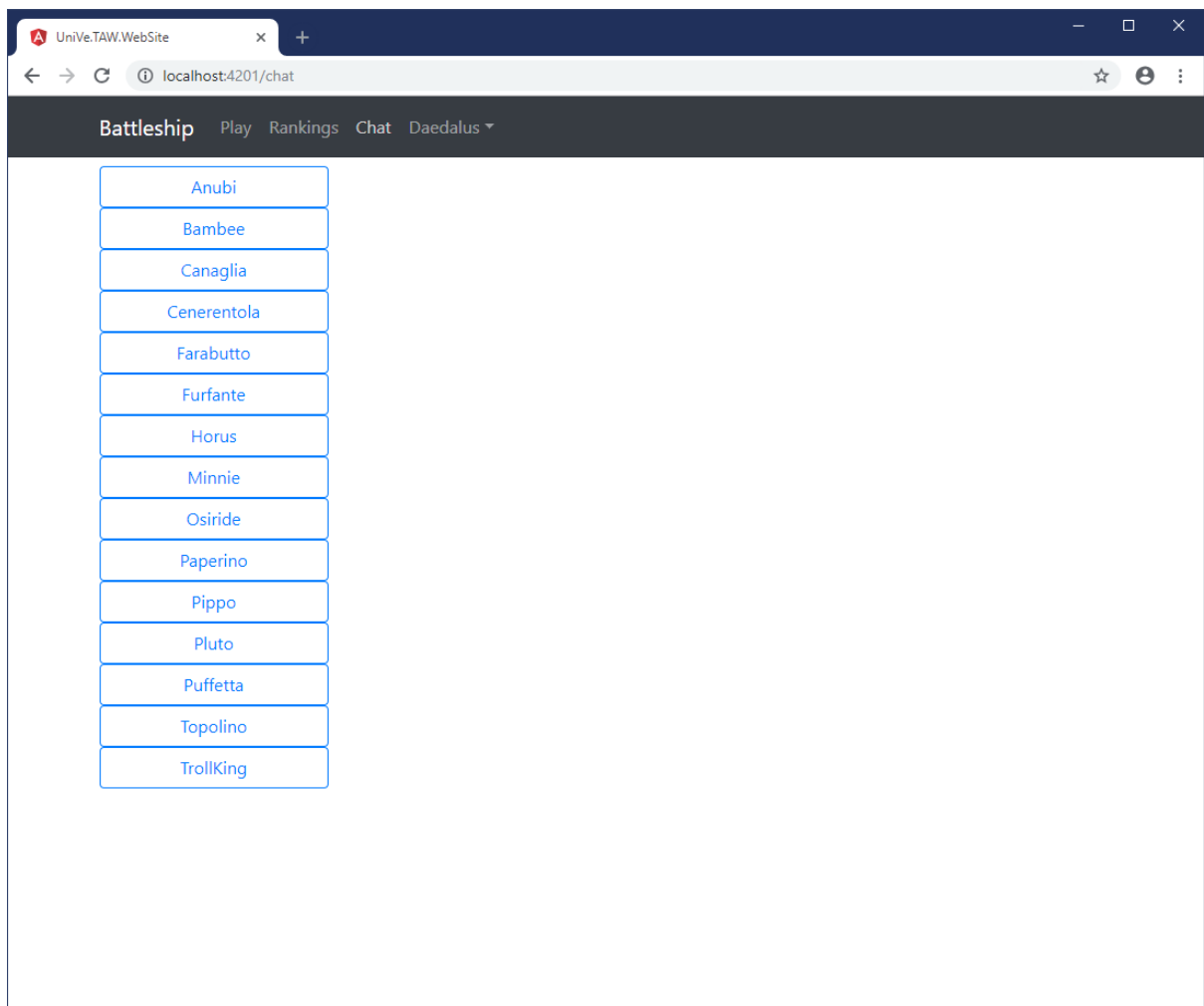


Chat

La pagina "Chat" rappresenta lo strumento per scambiare messaggi privati con qualsiasi utente e visualizzare il loro storico dei messaggi, anche quando non si è in partita. È mappata sulla route `"/chat"` e visualizza il component `"app/ui/game/match/match-chat/match-chat.component.ts"`, il quale mostra:

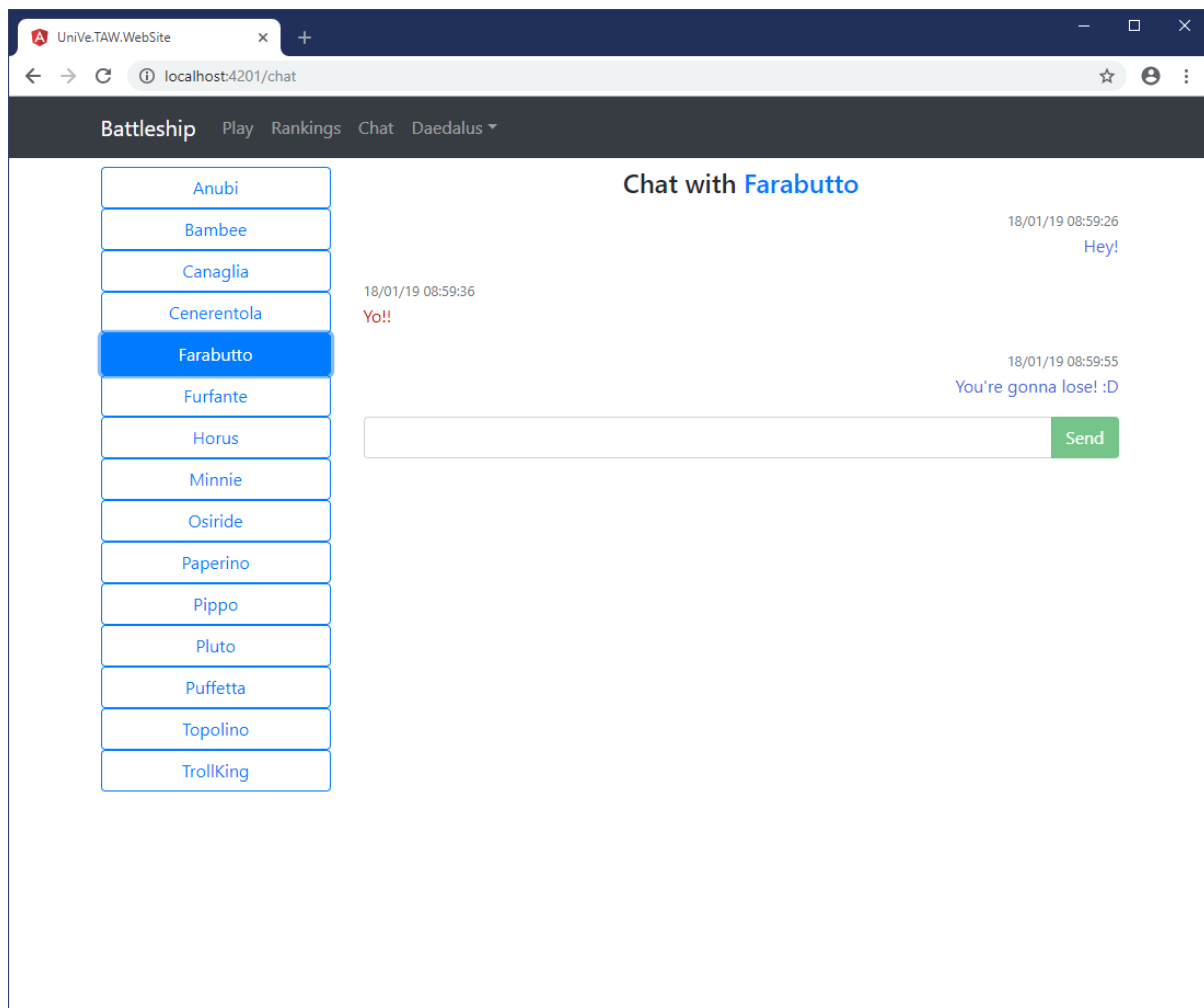
- la lista di tutti gli utenti con cui è possibile scambiare messaggi privati
- l'elenco di tutti i messaggi scambiati con uno specifico utente

Pagina delle chat - Elenco degli utenti con cui è possibile parlare



The screenshot shows a web browser window with the address bar displaying 'localhost:4201/chat'. The page has a dark blue header with the title 'Battleship' and navigation links for 'Play', 'Rankings', 'Chat', and 'Daedalus'. The main content area features a vertical list of 15 user names, each enclosed in a light blue rectangular button with a thin blue border. The user names are: Anubi, Bambee, Canaglia, Cenerentola, Farabutto, Furfante, Horus, Minnie, Osiride, Paperino, Pippo, Pluto, Puffetta, Topolino, and TrollKing.

Anubi
Bambee
Canaglia
Cenerentola
Farabutto
Furfante
Horus
Minnie
Osiride
Paperino
Pippo
Pluto
Puffetta
Topolino
TrollKing



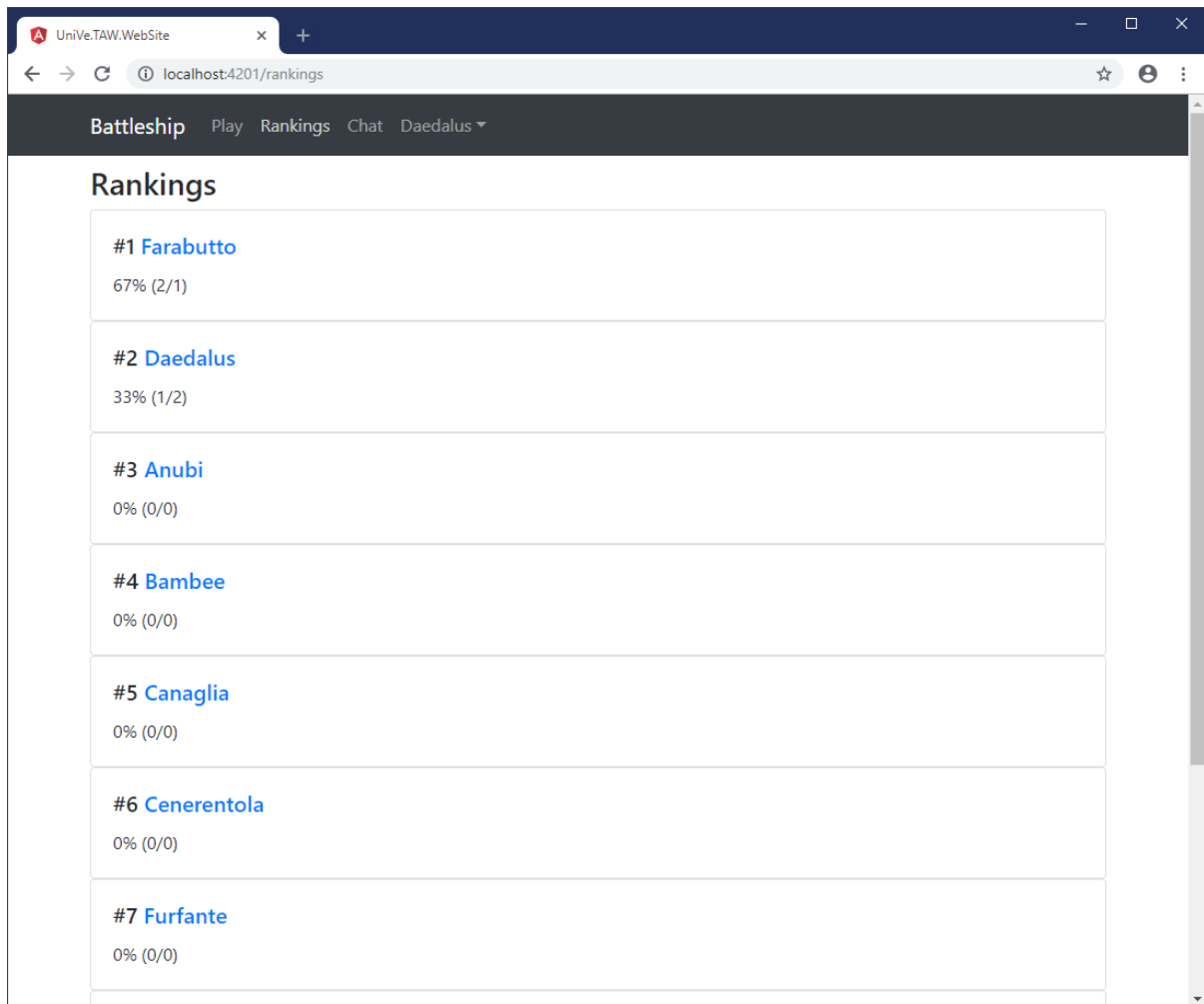
Rankings

La pagina dei rankings permette di visualizzare la classifica dei 10 migliori giocatori, valutati in base alla percentuale di vittorie/sconfitte e al numero partite giocate.

È mappata sulla route “/rankings” ed è rappresentata dal component “app/ui/identity/rankings/rankings.component.ts”.

Per ogni utente in classifica visualizza:

- Username
- Numero della graduatoria
- Percentuale delle partite vinte rispetto al totale delle partite giocate
- Numero di partite vinte
- Numero di partite perse



Rankings
#1 Farabutto 67% (2/1)
#2 Daedalus 33% (1/2)
#3 Anubi 0% (0/0)
#4 Bambee 0% (0/0)
#5 Canaglia 0% (0/0)
#6 Cenerentola 0% (0/0)
#7 Furfante 0% (0/0)

Profilo

La pagina profilo permette di visualizzare tutte le informazioni relative ad un utente, comprendendo oltre all'anagrafica dell'account anche lo storico delle partite giocate e i comandi di moderazione (visualizzabili solo dagli utenti con poteri elevati).

È mappata sulla route `"/user/:userId"` ed è rappresentata dal component `"app/ui/identity/users/profile/profile.component.ts"`, il quale visualizza al suo interno anche il componente relativo al Match History.

Match History

È rappresentato dal component `"app/ui/identity/users/match-history/match-history.component.ts"` e visualizza l'elenco delle ultime 20 partite giocate, visualizzando per ognuna di queste data e ora, avversario ed esito della partita.

UniVe.TAW.WebSite

localhost:4201/users/5c4186c6ca54d85a1045097b

BattleshipPlayRankingsChatDaedalus

Daedalus

Admin

59 years from UnitedKingdom

Stats

1W / 2L (33%)

Match history (last 20 matches)

WON VS Farabutto

18/01/19 16:43:19

LOST VS Farabutto

18/01/19 16:40:28

LOST VS Farabutto

18/01/19 16:35:07

The screenshot shows a web browser window with the URL `localhost:4201/users/5c422c1bd2684440686137f0`. The page title is 'Battleship' and the navigation bar includes 'Play', 'Rankings', 'Chat', and 'Daedalus'. The main content area displays the profile of a user named 'Farabutto' with the role 'Player'. The profile indicates the user is '12 years from SaintKitts_Nevis'. Below this, there are controls to 'Assign role of ...' (set to 'Player') and 'Assign', 'Ban for ...' (with a 'BAN' button), and 'I want to' (with a 'DELETE Farabutto' button). The 'Stats' section shows '15W / 22L (41%)'. The 'Match history (last 20 matches)' section lists four matches:

Result	Opponent	Time
WON	Bambee	18/01/19 20:42:32
LOST	Bambee	18/01/19 20:42:31
WON	Topolino	18/01/19 20:42:30
WON	Pluto	18/01/19 20:42:30

Installazione

- Visual Studio Code
 - [Link per il download](#)
- Node.js
 - [Link per il download](#)
- MongoDB - Community Server
 - [Link per il download](#)
 - Assicurarsi che la path della **cartella** che contiene mongod.exe (es. "C:\Program Files\MongoDB\Server\4.0\bin\mongod.exe") sia presente nella variabile di ambiente **Path** (utente o sistema)
- JDK
 - [Link per il download](#)
- Android Studio
 - [Link per il download](#)
 - Assicurarsi che le seguenti funzionalità siano installate e i relativi percorsi inseriti nella variabile di ambiente **Path**:

- Android SDK Emulator
 - Android SDK Platform-Tools
 - Android SDK Tools
 - Android SDK Build Tools
 - Almeno una versione di Android
- CLI
 - Angular: `npm i -g @angular-cli`
 - Ionic: `npm i -g ionic`
 - Cordova: `npm i -g cordova`
 - gulp: `npm i -g gulp-cli`
- Packages
 - Eseguire nella shell di VSCode di ogni progetto: `npm i`

Avvio

Avvio web service

- Aprire la cartella UniVe.TAW.WebService
- Eseguire: `mongod.cmd`
- Aprire la cartella UniVe.TAW.WebService in Visual Studio Code
- Per generare dati fittizi di utenti e partite, decommentare le righe [\[88, 114\]](#) del file ["src/application/ApiService.ts"](#)
- Eseguire:
 - Primo avvio: `Export dependencies & launch WebService`
 - Riavvio: `Launch WebService`

Avvio web app

Importante: Prima di avviare il client, assicurarsi di che il task "Export dependencies" del progetto WebService sia stato eseguito almeno una volta, tramite il pannello dei task o tramite il debug launcher "Export dependencies & launch WebService"

- Aprire la cartella UniVe.TAW.WebSite in Visual Studio Code
- Eseguire il comando: `npm start`
- Premere: F5

Avvio mobile app

Importante: Prima di avviare il client, assicurarsi di che il task "Export dependencies" del progetto WebService sia stato eseguito almeno una volta, tramite il pannello dei task o tramite il debug launcher "Export dependencies & launch WebService"

- Aprire la cartella UniVe.TAW.Mobile in Visual Studio Code
- Aprire il file [src\app\services\ServiceConstants.ts](#)
- Web browser
 - Impostare l'IP del servizio web a: 127.0.0.1
 - Eseguire il **task**: ionic cordova run browser
- Emulatore Android
 - Impostare l'IP del servizio web a: 10.0.2.2
 - Eseguire il **task**: ionic cordova run android

Avvio applicazione windows

Importante: Prima di avviare il client, assicurarsi di che il task "Export dependencies" del progetto WebService sia stato eseguito almeno una volta, tramite il pannello dei task o tramite il debug launcher "Export dependencies & launch WebService"

- Aprire la cartella UniVe.TAW.Windows in Visual Studio Code
- Eseguire il comando: npm start