**Progetto di IUM-TWEB**

Anno 2023/2024

realizzato dagli studenti:

* Bilguun Batsaikhan
* Emanuele Machetti
* Alex Agliuzza

***Introduzione***  
Abbiamo realizzato 9 pagine per il nostro sito: home, competition, chat, registrazione, login, about, club, player e search. Le principali classi CSS utilizzate servono per creare la navbar e formattare le tabelle (usando Bootstrap), rendendo le pagine responsive e ben organizzate, inclusa la pagina di registrazione. Il JavaScript è presente in tutte le pagine, tranne nella pagina statica "about", ed è utilizzato per effettuare chiamate Axios e inserire nel DOM gli elementi ricavati dalle query.

***Homepage***

**Soluzione:** la homepage contiene gli 8 giocatori più popolari. Cliccando su ciascuno di essi, si accede alla pagina del giocatore. Questi giocatori sono ottenuti tramite una query che seleziona quelli con il valore di mercato attuale più alto.

Inoltre, nella homepage vengono visualizzate le 8 competizioni più popolari, inserite staticamente. Cliccando su ciascuna competizione, si accede alla relativa pagina di competizione. Abbiamo selezionato queste competizioni in base alla loro popolarità, utilizzando come fonte questo sito:<https://www.eurosport.it/calcio/iffhs-serie-a-miglior-campionato-al-mondo-dell-anno-solare-2023.-inseguono-premier-league-e-liga_sto9991361/story.shtml>.  
**Problemi tecnici:** L'unico inconveniente riscontrato è stato legato alla grafica, soprattutto nell'utilizzo del CSS per rendere la sezione dei giocatori e delle competizioni responsive.  
**Conformità della soluzione:** La soluzione è conforme poiché abbiamo utilizzato le chiamate Axios mostrate durante le lezioni, insieme alle classi CSS e al JavaScript per inserire dinamicamente gli elementi.  
**Limitazioni:** Per quanto riguarda i giocatori popolari, abbiamo scelto di inserirli dinamicamente per consentire una variazione nel caso in cui vengano aggiunti nuovi giocatori con un alto valore di mercato nel database. Per le competizioni, invece, non abbiamo adottato un approccio dinamico poiché avrebbe richiesto l'implementazione di query per ottenere le diverse competizioni. Tuttavia, considerando che la popolarità delle competizioni non varia significativamente nel tempo, abbiamo optato per questa soluzione pur conoscendo le relative limitazioni.

**Searchbar  
Soluzione:** Per facilitare l'accesso alle pagine dei giocatori e dei club, abbiamo introdotto una barra di ricerca. Questa consente agli utenti di digitare un testo e premere Invio per ottenere risultati correlati alle pagine dei giocatori e dei club, utilizzando una query "like". I risultati vengono visualizzati in una nuova pagina, dove è possibile fare clic sui nomi delle squadre o dei giocatori per accedere alle rispettive pagine.  
**Problemi tecnici:** Non sono stati riscontrati problemi tecnici durante l'implementazione di questa funzionalità.  
**Conformità della soluzione:** Sebbene sia una semplificazione, questa soluzione si è dimostrata efficiente per un uso didattico. Si basa su chiamate leggere con Axios e su una manipolazione semplice con JavaScript.  
**Limitazioni:** In situazioni reali, potrebbe essere preferibile mostrare i risultati della ricerca tramite un menù a tendina direttamente sulla stessa pagina in cui è stata effettuata la ricerca. Tuttavia, abbiamo scelto una soluzione più leggera per ridurre l'impatto sul server, dato che l'esecuzione della query di ricerca ad ogni variazione del testo digitato potrebbe causare un carico elevato sul server. Questa soluzione è stata presa anche considerando che persino in siti di grande portata, questa operazione può risultare lenta, rendendoci consapevoli delle sfide nell'implementare una soluzione efficiente.

**Rotte dinamiche e passaggio di parametri GET**Prima di esplorare le prossime pagine, è importante comprendere la soluzione implementata per consentire al sito web di caricare diverse pagine in base all'URL navigato.   
Le rotte dinamiche permettono al sito di adattarsi e fornire contenuti specifici in base all'URL visitato dall'utente. Ad esempio, se accediamo alla pagina di competizione che mostra i match della Coppa Italia, come nell'URL seguente:  
[**http://localhost:3000/competition-table?competition=italy-cup&country=Italy**](http://localhost:3000/competition-table?competition=italy-cup&country=Italy), la parte principale dell'URL è "/competition-table", la quale rappresenta la rotta principale mappata nel sistema. I parametri aggiuntivi, come "competition=italy-cup" e "country=Italy", forniti nella query string dell'URL, specificano i dettagli della competizione che l'utente desidera visualizzare. Questi parametri vengono utilizzati dal sito per determinare quali contenuti mostrare, ad esempio recuperando dati specifici dalla competizione di Coppa Italia dal database e visualizzandoli sulla pagina.  
**Problemi tecnici:** Trovare questa soluzione non è stato immediato. Benché avessimo visto alcuni esempi di rotte durante le lezioni, concepire un approccio del genere ha richiesto un po' di tempo e sforzo. Tuttavia, ha aperto la strada a un'implementazione dinamica e flessibile nella visualizzazione dei dati. La sfida principale è stata più nel concepire la soluzione che nell'effettuare l'implementazione pratica.  
**Conformità della soluzione:** La nostra soluzione si adatta perfettamente al contesto del sito web, e lo stesso approccio è impiegato con successo su piattaforme di grandi dimensioni come [www.fotmob.com](http://www.fotmob.com). Questo modello consente di eseguire interrogazioni dinamiche senza dover utilizzare menu a tendina per selezionare i valori e interrogare nuovamente il database tramite rotte statiche.  
**Limitazioni:** È importante gestire attentamente i parametri dell'URL per evitare valori non validi o manipolazioni da parte degli utenti. Nel caso in cui uno dei parametri assuma un valore incoerente, è consigliabile restituire una pagina di errore appropriata per garantire un'esperienza utente coerente e sicura.

**Competitions  
Soluzione:** La pagina delle competizioni è accessibile dalla homepage tramite la sezione delle competizioni popolari o la navbar, che consente di filtrare per tipo e paese. Per il filtro per tipo di competizione, viene eseguita una query per ottenere i diversi tipi di competizioni disponibili (es: europa\_league\_qualifying). Una volta posizionato il cursore del mouse su un tipo specifico di competizione, viene eseguita un'altra query per ottenere una competizione più specifica (es: europa-league-qualifikation). Per il filtro per paese, viene effettuata una query per ottenere i diversi paesi delle competizioni (es: Turkey, Ukraine, Spain, Italy, ecc.). Successivamente, posizionando il cursore del mouse su un paese specifico, vengono selezionate delle competizioni specifiche per quel paese (es: selezionando Italy comparirà Serie A). Nella pagina della competizione viene mostrata una tabella con tutti i match di quella particolare competizione, con le seguenti colonne: Date, Home Team, Score, Away Team, Stadium. Cliccando sul nome delle squadre è possibile raggiungere la pagina della squadra. È inoltre possibile ordinare i risultati per anno e scorrere avanti e indietro tra i risultati mostrati, visualizzati come pagine. Tutti i dati sono ottenuti con chiamate Axios.  
**Problemi tecnici:** Ci sono stati problemi con la paginazione a causa del pulsante "show more" perché occorreva ricalcolare il numero delle pagine. Inoltre, abbiamo incontrato difficoltà nell'utilizzo di più server per ottenere determinate informazioni (era quasi come fare una join in SQL, ma usavamo l'ID di una tabella come parametro per ottenere informazioni da un'altra tabella). All'inizio è stato complicato costruire le tabelle dinamicamente, anche perché sono state create due schede per mostrare i dati nella stessa tabella.  
**Conformità della soluzione:** La soluzione utilizza Axios e javascript per ottenere le query, per tale motivo risulta conforme.  
**Limitazioni:** Sarebbe forse stato preferibile trovare un modo per ottenere direttamente il tipo di competizione più specifica, dato che il nome varia solo leggermente (es: europa\_league\_qualifying vs europa-league-qualifikation).

**Clubs**

**Soluzione:** La pagina del club è accessibile dalla searchbar o cliccando sul nome della squadra nelle altre pagine. Questa pagina mostra informazioni generali sulla squadra, il valore di mercato e la lista dei giocatori dell’ultima stagione, escludendo i giocatori del passato. Le chiamate axios sono basate sul parametro GET club\_id.  
**Problemi tecnici:** Uno dei problemi tecnici è stato implementare la paginazione in JavaScript.  
**Conformità della soluzione:** La soluzione è conforme.  
**Limitazioni:** Sarebbe stato utile includere una sezione storica per i giocatori che hanno militato nella squadra in passato.

**Chat  
Soluzione:** Abbiamo implementato una chat, come mostrato durante le lezioni, accessibile solo dopo la registrazione o il login. Questo perché utilizziamo una variabile "nickname" memorizzata nel localStorage.  
**Problemi tecnici:** Qualche piccolo problema con il CSS.  
**Conformità della soluzione:** La chat presenta un codice molto simile all'esempio visto in classe, con la differenza che la stanza della chat a cui accedere viene ottenuta tramite una chiamata Axios. È stata creata una stanza di chat per ciascun nome di club. Inoltre, abbiamo utilizzato il localStorage per mettere in pratica un altro esempio visto in laboratorio. **Limitazioni:** In un contesto reale, il login non può essere gestito semplicemente salvando una variabile nel localStorage. È più opportuno utilizzare un cookie di sessione per garantire una maggiore sicurezza.  
  
**Giocatori  
Soluzione:** La pagina dei giocatori è accessibile dalla searchbar, dalla pagina del club e dalla sezione "popular players" in homepage (solo per gli 8 più popolari). La pagina mostra dettagli sul giocatore selezionato, come altezza, valore di mercato, totale di gol/assist e totale di cartellini gialli/rossi nelle partite disputate. Sulla destra, c’è una paginazione con le partite giocate dal giocatore e i risultati ottenuti/gol segnati.  
**Problemi tecnici:** Le chiamate server-side sono state complesse e abbiamo dovuto fare compromessi, mostrando solo le ultime 300 partite del giocatore. Inoltre, nei match disputati, il giocatore potrebbe aver giocato per una squadra diversa da quella attuale (es: Erling Haaland nel maggio 2022 giocava per il Borussia Dortmund).  
**Conformità della soluzione:** La soluzione è in linea con quanto visto a lezione, perché fa uso di js e delle chiamate Axios.  
**Limitazioni:** Non è stato possibile ottenere più di 300 partite a causa di un problema di dimensione massima (Payload Too Large). Una soluzione ottimale sarebbe stata effettuare singole chiamate Axios per ciascuna pagina dei match, ottenendo solo 10 risultati alla volta, ma per motivi di tempistiche non è stato possibile implementare questo metodo.

***I Servers***  
**Soluzione:** Nella nostra soluzione abbiamo implementato un Main e un Secondary server in Express e un terzo server in Spring Boot. Ogni chiamata proveniente dal client viene inviata tramite axios al Main server che si occupa esclusivamente di dirottare tale richiesta al server corrispondente (secondary o Spring Boot), pertanto il server primario non esegue nessun compito di logica interna, neppure per la chat.  
**Problemi tecnici:** Molto spesso abbiamo avuto qualche problemino con le chiamate server-side, alcune volte perchè restituivano errore 505, altre perchè per un qualche dettaglio non funzionavano, per questo abbiamo usato Postman per simulare le chiamate POST per riuscire a capire i problemi.  
**Conformità della soluzione:** La soluzione rispetta la convenzione di utilizzare il server primario per richieste leggere, inoltre abbiamo implementato i servers come da assegnamento. La soluzione è conforme a quanto richiesto.  
**Limitazioni:** Forse potevano essere implementate chiamate dirette dal js del client, ma provando a fare queste chiamate con Axios abbiamo ottenuto un problema di CORS (Cross-Origin Resource Sharing). Tuttavia, indirizzare le richieste attraverso il main server migliora la sicurezza perché consente di nascondere gli endpoint sensibili e proteggere le informazioni critiche. Inoltre, questa soluzione garantisce che solo le richieste autorizzate possano accedere alle risorse desiderate se i servers venissero configurati solo per accettare richieste provenienti dal Main server.

**Divisione dei datasetSoluzione:** Abbiamo suddiviso i database in due categorie, come da assegnamento.  
I dati statici, conservati in PostGres, comprendono i CSV di competitions e players, poiché il tipo di competizioni e i valori di mercato dei giocatori tendono a cambiare raramente. I dati dinamici sono invece salvati in MongoDB e includono informazioni come apparences, club\_games, game\_events, clubs, game\_lineups e player\_valutations. In particolare, le apparizioni sono aggiornate frequentemente. Le partite e gli eventi di gioco vengono aggiunti per ogni partita giocata. Abbiamo deciso di mantenere dinamico anche il dataset delle player\_valutations, poiché sebbene vengano aggiornate solo ad ogni stagione, coinvolgono numerosi giocatori e richiedono un aggiornamento sostanziale dei dati.  
**Problemi tecnici:** L’importazione del csv in PostGres non è stata facilissima, in quanto abbiamo utilizzato PSQL Tool che non eravamo abituati ad utilizzare in precedenza, poiché anche nel corso di basi di dati abbiamo sempre creato le tabelle da zero.  
**Conformità della soluzione:** L'unica vera discrepanza potrebbe essere il salvataggio dei dati relativi ai clubs come dinamici, siccome vengono aggiornati solo ad ogni stagione e nel database viene mantenuta una sola tupla aggiornata per ogni club. Nonostante ciò, non abbiamo avuto modo di correggere questa anomalia nell'ultima fase del progetto perchè avremmo dovuto cambiare molte chiamate Axios, tuttavia riteniamo che la nostra divisione per i restanti dataset rispecchi le esigenze del progetto.  
**Limitazioni:** Non abbiamo trovato alcuna limitazione.  
  
**Analisi dei dati (jupyter)**L'analisi dei dati rappresenta una componente fondamentale del progetto, poiché senza una conoscenza approfondita dei dati è impossibile rappresentarli in modo efficace. Questa fase si articola in diverse fasi principali. Inizialmente, è stato eseguito uno studio preliminare per comprendere la struttura e le caratteristiche generali dei dati. Successivamente, si è proceduto con la pulizia dei dati, rimuovendo le anomalie e correggendo gli errori per garantire l'accuratezza dei dati stessi. Si è quindi passati all'analisi dettagliata dei valori presenti in ogni colonna e allo studio delle correlazioni e delle interazioni tra le diverse colonne. Infine, sono stati realizzati dei grafici per visualizzare le informazioni in modo chiaro e comprensibile.

**Conclusioni**Alla fine del progetto, ci siamo resi conto che è stato un lavoro molto lungo che ha richiesto mesi per essere completato. Tuttavia, questo ci ha permesso di comprendere dettagliatamente il lavoro necessario per la creazione di un sito web e il motivo per cui molti sviluppatori continuano a utilizzare framework o CMS come WordPress per realizzarli.  
Nel complesso, le lezioni sono state estremamente utili e il lavoro di gruppo si è rivelato indispensabile per il supporto reciproco. Senza questa collaborazione, il progetto sarebbe stato molto più oneroso da portare a termine. Abbiamo acquisito preziose competenze tecniche, come ad esempio l’utilizzo di parametri nell’URL (un concetto che avevamo visto solo teoricamente nel corso di Reti), e appreso l'importanza della cooperazione.  
Riassumendo, è stato difficile aggregare una grande quantità di dati e presentarli in modo utile per l'utente.

**Divisione del lavoro**

***Alex*** ha gestito interamente la parte relativa a login, registrazione e logout, il back-end per la searchbar e la pagina del player, oltre ad altre chiamate server-side minori. Ha lavorato su molto CSS tra le diverse pagine. Per l’analisi dei dati, ha fornito un’analisi molto dettagliata e completa di club\_games, clubs e game\_events. Ha supportato Emanuele e Bilguun per la risoluzione di problemi generali.  
***Bilguun*** si è occupato del CSS della homepage, della navbar, del doppio drop-down menu e del footer. In particolare, ha lavorato sulla pagina delle competizioni (una delle più importanti a causa della grande quantità di dati) con il back-end e il front-end, ha inoltre utilizzato Flask per mostrare un grafico. Ha gestito il JavaScript locale per visualizzare i dati nella search page. Ha analizzato i dati relativi a competitions, games e appearances.   
***Emanuele*** ha curato la visualizzazione dei popular players nella homepage, il back-end della chat e la sua totale implementazione. Ha ideato il meccanismo delle rotte dinamiche grazie alla sua precedente esperienza con il framework Laravel di PHP. Inoltre, ha sviluppato il back-end e il front-end della pagina del club e il solo front-end per la pagina del giocatore. Per quanto riguarda l'analisi dei dati, ha fornito un’analisi di game\_lineups, players, player\_valuation e parzialmente di appearances.  
  
 **Informazioni extra**: E’ necessario eseguire “npm install” nella console del Main Server per poter scaricare tutte le dipendenze necessarie. Nel server SpringBoot cliccare a destra sul simbolo dell’elefantino (gradle), successivamente cliccare su “Reload All Gradle Projects”.