BIASET
ALWAYS BY OUR SIDE

SDD: System Design Document

Riferimento	
Versione	1.5
Data	05/01/2022
Destinatario	Prof. Carmine Gravino
Presentato da	Danilo Aliberti
Approvato da	

Data	Versione	Descrizione	Autore
30/09/2021	0.1	Stesura iniziale	Danilo Aliberti
09/10/2021	1.0	Aggiornamento	Danilo Aliberti
14/11/2021	1.1	Aggiornamento	Danilo Aliberti
02/12/2021	1.2	Aggiornamento	Danilo Aliberti
05/01/2022	1.5	Revisione e aggiornamento	Danilo Aliberti

Sommario

1. Introduzione

1.1 Obiettivi del sistema

Il fantacalcio è uno dei giochi più amati e giocati dai giovani italiani. Esistono numerose piattaforme che permettono la gestione automatizzata di campionati fantacalcistici, ma, la maggior parte, si basa sempre sulle regole classiche del fantacalcio. Ciò non fa altro che standardizzare e rendere monotona l'esperienza dei fantallenatori, i quali potrebbero, col passare del tempo, perdere interesse verso il gioco. La piattaforma BiaSet offre un approccio differente, rivoluzionando le regole del fantacalcio, rendendolo più dinamico e avvincente. Ad oggi non esiste alcuna piattaforma con tali regole, quindi sarebbe la prima app "sul mercato" a fornire un servizio innovativo e coinvolgente.

1.2 Design goals

Nella tabella seguente sono illustrati gli obiettivi di design e le corrispondenti proprietà (a numeri più bassi corrisponde una priorità più alta). Per ogni goal è riportata anche la sua origine.

Priorità	ID	Descrizione	Categoria	Origine
1	DG_1	**Robustezza**: Il sistema deve sopravvivere agli input errati degli utenti. In un caso del genere il sistema non deve accettare l'input e notificare l'utente invitandolo a correggere l'errore	Reliability	RNF_R_3
2	DG_2	**Sicurezza**: Il sistema deve comunicare tramite protocollo HTTPS in modo da assicurare una maggiore sicurezza	Reliability	RNF_R_1 RNF_R_2
1	DG_3	**Usabilità**: Il sistema deve essere facile da apprendere ed intuitivo da utilizzare senza necessariamente consultare la documentazione. I contenuti dovranno essere fruibili attraverso dispositivi sia desktop che mobile ed accessibili attraverso un numero ridotto di interazioni	Usability	RNF_U_1 RNF_U_2 RNF_U_3
2	DG_4	**Costi di sviluppo**: Bisogna abbattere i costi di sviluppo del sistema	Costo	Top Management

1	DG_5	**Tempi di sviluppo**: Bisogna sviluppare il sistema in al più due mesi	Costo	Top Management
3	DG_6	**Throughput**: Il sistema deve essere in grado di supportare oltre 100 utenti connessi	Performance	RNF_P_4
3	DG_7	**Tempi di risposta**: Il sistema deve elaborare le richieste e produrre output in meno di 2 secondi (al netto di ritardi dovuti alla trasmissione su rete)	Performance	RNF_P_2
2	DG_8	**Leggibilità**: Il codice prodotto dev'essere semplice da comprendere. Ogni metodo e campo non banale dev'essere documentato opportunamente al fine di aumentarne la comprensione	Supportability	RNF_S_1

1.2.1 Trade-offs

Tempo di rilascio vs Funzionalità
Nonostante i tempi di sviluppo ridotti, il team si impegna nel consegnare il sistema completo di tutte le sue funzionalità "core", tenendo conto di un possibile ritardo nella consegna.
Prestazioni vs Costi
Per rientrare nel budget a disposizione, il team cercherà di ottenere le migliori prestazioni ma nelle ore-lavoro garantite dal budget.

1.3 Definizioni, acronimi e abbreviazioni

Nel corso del documento si ricorrerà alla seguente distinzione:

- **CRUD**: Create, Read, Update, Delete;
- **Piattaforma**: applicazione web;
- **CA**: Championship Admin;
- **LA**: League Admin;
- **COTS**: Component Off The Shelf;
- **Modello ER**: modello Entità-Relazione per dati persistenti.

1.4 Riferimenti

- **Requisiti funzionali:** sezione 3.1 del RAD;

- **Requisiti non funzionali:** sezione 3.2 del RAD.

1.5 Overview

Nel seguente documento sarà effettuata l'analisi di architetture di sistemi simili, la scomposizione in sottosistemi del sistema proposto, con definizione della strategia di deploy e le condizioni limite. Verranno, infine, definiti i servizi offerti da ciascun sottosistema.

2. Architettura di sistemi simili

Attualmente, la lega è gestita da una piattaforma obsoleta e limitata. Quest'ultima, allo stato attuale, permette la gestione di un **singolo** campionato di fantacalcio, precludendo la partecipazione di numerosi allenatori che avrebbero piacere e voglia di far parte della lega. Molti processi, quali, ad esempio, l'assegnazione dei giocatori ad una squadra e il calcolo automatizzato, sono complicati da gestire e richiedono una conoscenza approfondita della piattaforma da parte del semplice utente o gestore. Il sistema che ospita la piattaforma è a sua volta vecchio e, pertanto, probabilmente soggetto a problemi di sicurezza, in quanto non aggiornato alle ultime versioni. Anche e soprattutto dal punto di vista progettuale risulta superato, siccome non è effettivamente scritto utilizzando il paradigma OOP.
Un altro sistema simile è quello fornito da Fantacalcio.it, il quale offre la possibilità di gestire più campionati simultanei, inserire formazioni/calendario e calcolo automatizzato. È un sistema limitato, in quanto è vincolato alle regole classiche del fantacalcio e non permette variazione. La sua architettura è molto simile a quella che sarà adottata nel sistema proposto. Dopo un'analisi effettuata, risulta che Fantacalcio.it è sviluppato con il framework ASP.NET Core e, di conseguenza, con MVC come pattern architetturale. Si appoggia ad un DBMS, probabilmente SQL Server, per restare in linea con il framework di famiglia Microsoft.

3. Proposed System

3.1 Panoramica

BiaSet è una applicazione distribuita accessibile mediante interfaccia web. Si appoggia ad un database relazionale per la persistenza dei dati. Tale sezione rappresenta il punto focale del documento.

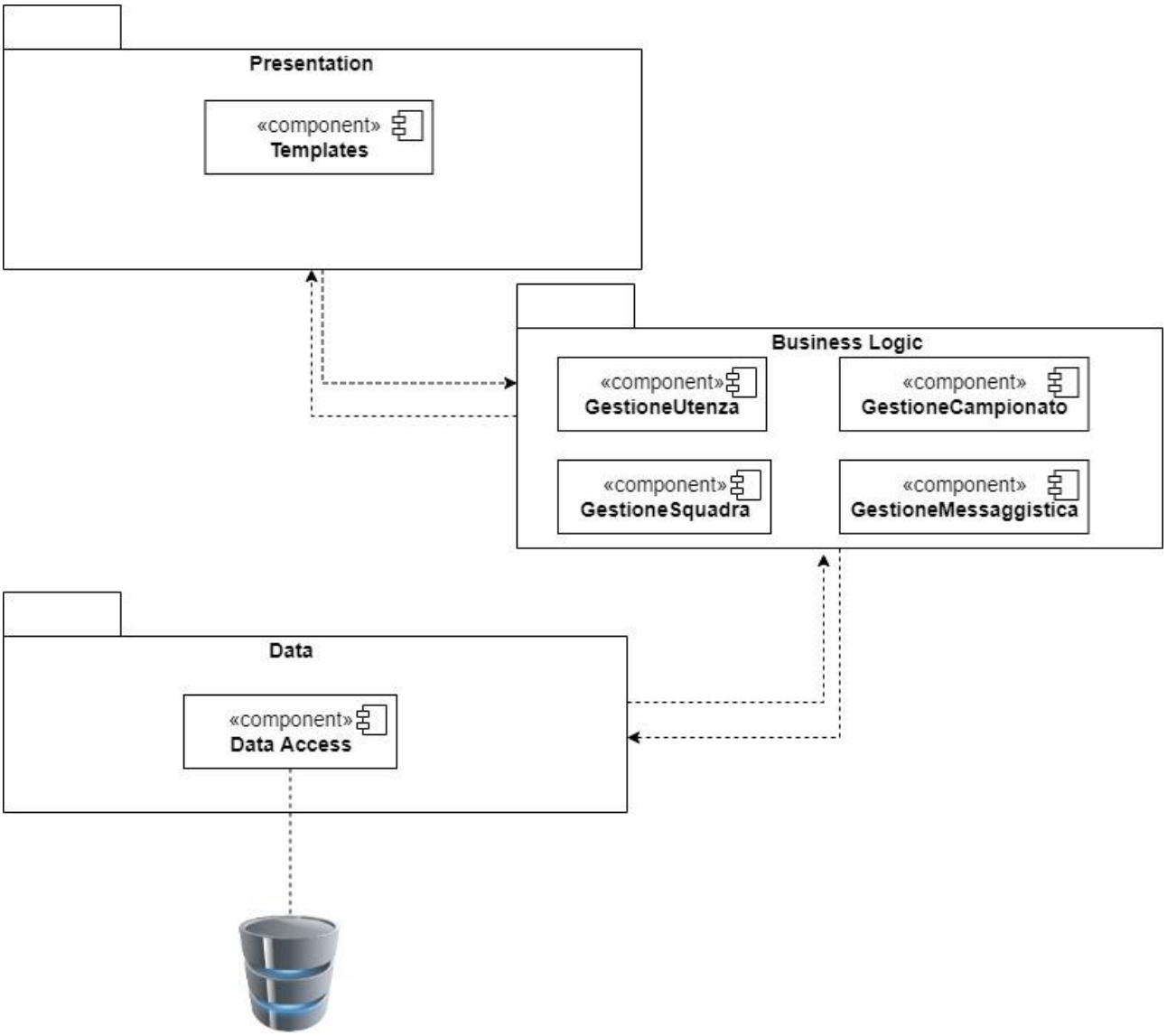
- Nella sezione **3.2** sarà affrontata la scomposizione in sottosistemi, con cenni al modello architetturale utilizzato;
- Nella **3.3**, invece, si procederà al mapping hardware-software;
- La **3.4** riguarda la gestione dei dati persistenti, con un modello ER primordiale;
- La sezione **3.5** tratta dei permessi di accesso alle funzionalità del sistema, attraverso una matrice degli accessi;
- Nella **3.6** si tratterà del flusso globale dell'applicazione, con cenni alla gestione thread-driven;
- Nella sezione **3.7** sarà trattato il comportamento del sistema a fronte di condizioni limite, quali shutdown, fallimento e startup;
- Nella **3.8** saranno listate e descritte le interfacce offerte dai sottosistemi.

3.2 Scomposizione in sottosistemi

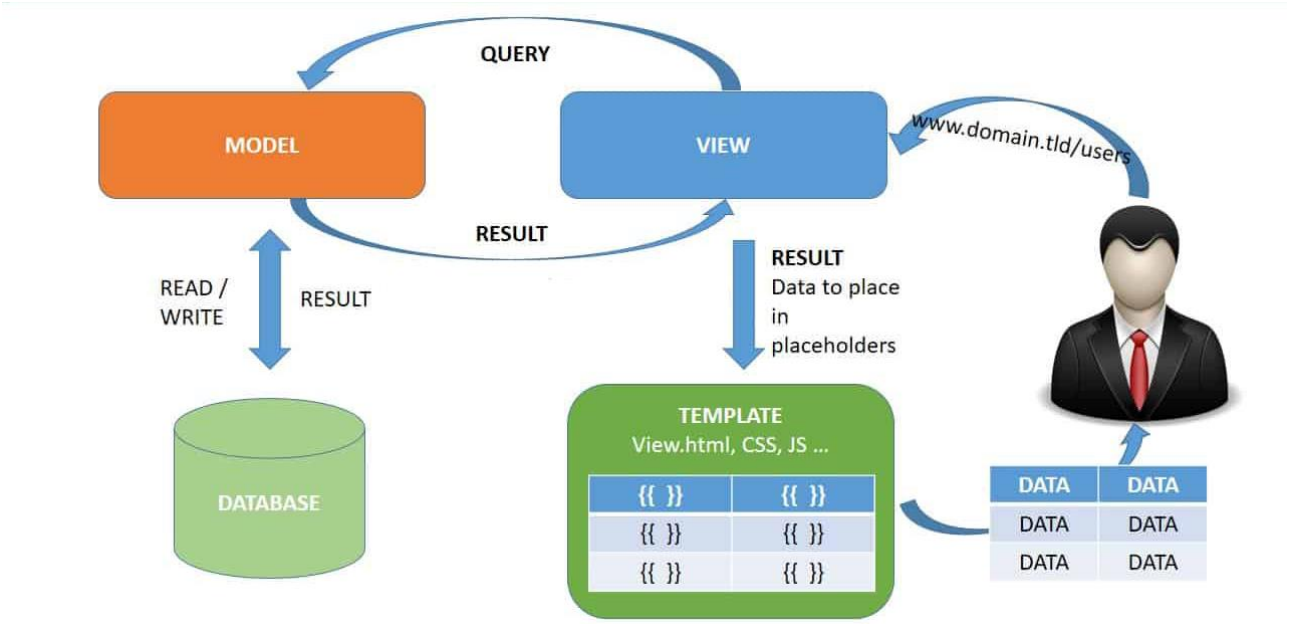
Si è deciso di far basare il sistema su un'architettura MVT (Model View Template), organizzata in modo da minimizzare l'accoppiamento e favorire un'alta coesione, rendendo altrettanto semplici le modifiche da effettuare. Tale architettura va a suddividere il sistema in 3 livelli:

- **Presentation:** (identificata con il Template) la quale contiene la parte del Frontend, ovvero l'interfaccia con la quale interagiranno gli utenti;
- **Business Logic:** (identificata con la View) cioè il "cervello" della nostra applicazione, la parte logica che rende la piattaforma intelligente e viva;

- **Data:** (identificata con il Model) insieme al Data Access, il quale ha accesso al database per scrivere/cancellare/modificare dati.



La differenza sostanziale con il classico MVC è che, in quest'ultimo, dobbiamo scrivere tutta la parte di codice relativa al Control. In un MVT, invece, la parte del Controller è interamente gestita dal framework. Ad esempio: immaginiamo di voler visualizzare una lista di Libri disponibili in una libreria, salvati in una tabella chiamata *libri*. In un sistema ad architettura MVC dovremmo scrivere tutta la parte di codice per il fetch dei dati dal database, il presentation layer (pagine HTML per la presentazione dei dati), mappare il tutto con una URL e inviarli all'utente. Nei framework basati su MVT, come quello utilizzato nel nostro sistema, non bisogna scrivere alcuna riga di codice collegata al recupero dei dati dal database né mapparli attraverso una URL. Tutte queste attività sono interamente gestite dal framework. L'unica cosa da fare è quella di indicare al framework quali dati presentare. Esso creerà automaticamente una View basata su quegli specifici dati e la mostrerà all'utente. Di seguito è rappresentato il funzionamento di un'architettura MVT.

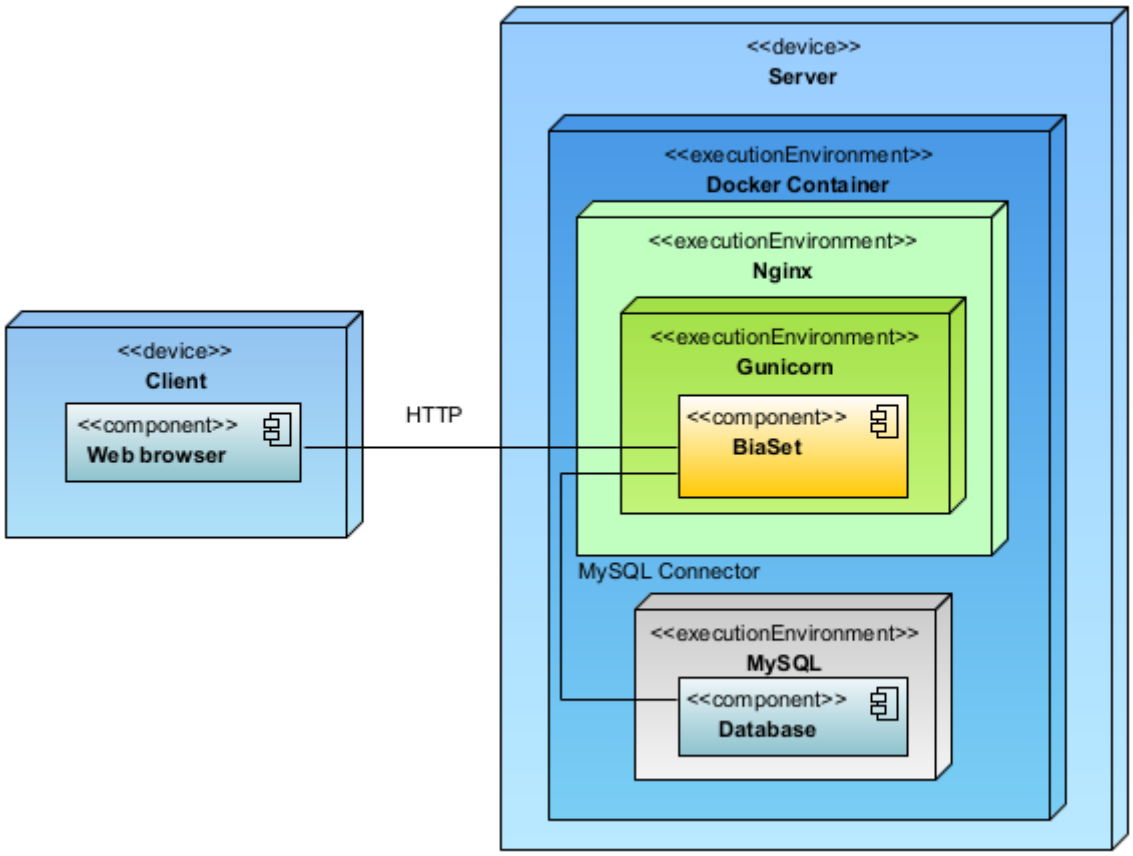


3.3 Hardware/software mapping

BiaSet è un'applicazione distribuita installabile su qualsiasi macchina in grado di eseguire **Docker**. Quest'ultimo automatizza il processo di deployment di applicazioni all'interno di container software fornendo un'astrazione aggiuntiva grazie alla virtualizzazione a livello di sistema operativo di Linux. All'interno del container di Docker, troveremo un ambiente Linux con i seguenti pacchetti preinstallati:

- Python 3.8 ed eventuali sue dipendenze;
- MySQL aggiornato all'ultima versione ed eventuali sue dipendenze;
- Web server Gunicorn.

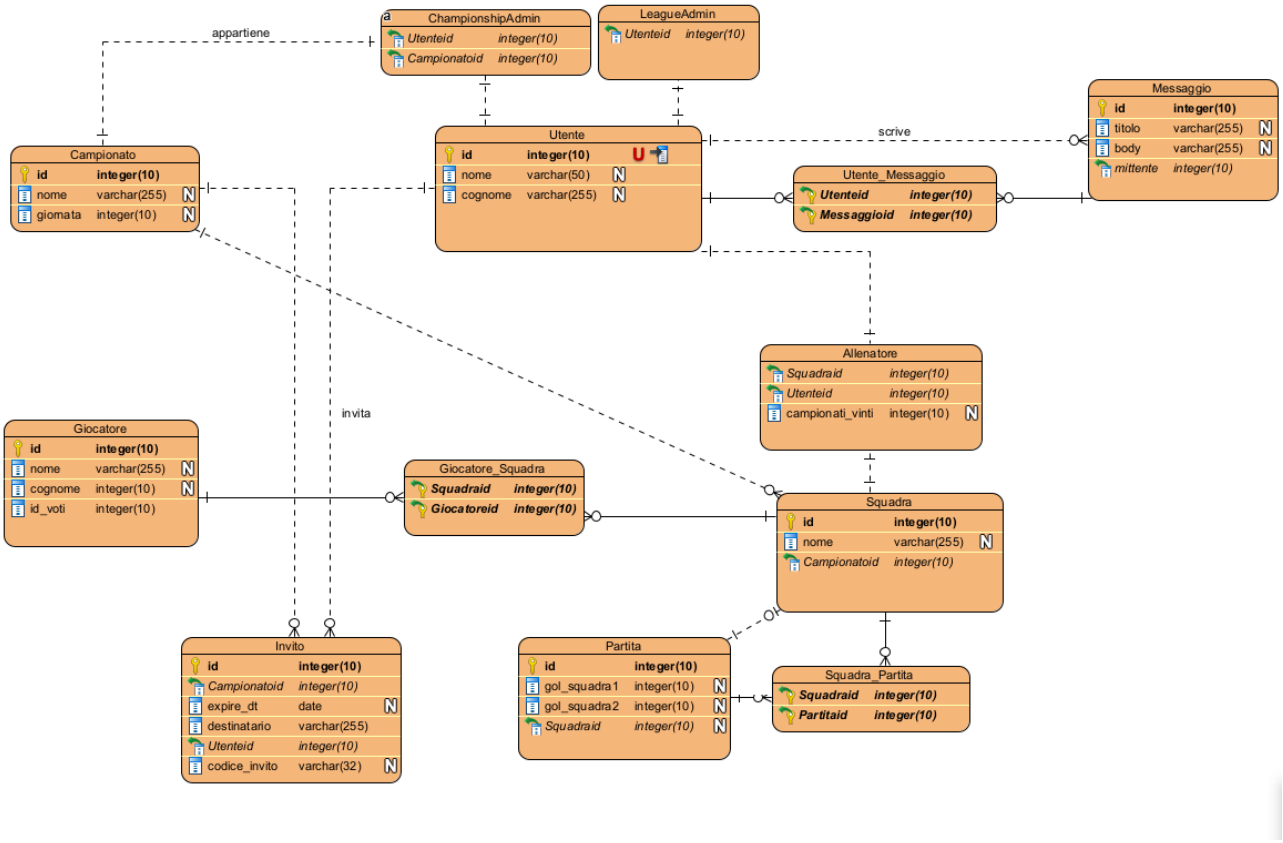
Il container andrà ad isolare totalmente la nostra applicazione, in modo da rendere più sicura la macchina che la ospita, in caso di violazione. Una volta servita sul web, l'applicazione sarà accessibile da qualsiasi browser.



Il sistema sarà accessibile attraverso browser web (lato Client) installato sul dispositivo degli utenti. L'app, lato Server, è situata all'interno di un container della Docker Machine, totalmente isolata dal resto della macchina. All'interno della Docker Machine, troveremo tutti i servizi necessari per il funzionamento e le dipendenze richieste dall'app stessa, come, ad esempio, il database MySQL per la persistenza dei dati.

3.4 Gestione dei dati persistenti

Per la gestione dei dati persistenti, BiaSet si affida ad un DBMS, gestito tramite MySQL. La struttura interna del database segue il seguente schema:



3.5 Controllo degli accessi e sicurezza

Il controllo degli accessi è garantito e gestito tramite un COTS. Gli utenti potranno iscriversi alla piattaforma, previo invito, attraverso i social network. Quindi, non saranno salvate password personali degli utenti, ma bensì token di accesso, i quali dovranno essere rinnovati di volta in volta per confermare l'identità dell'utente. Gli utenti loggati potranno creare o modificare gli oggetti che modellano entità di domini. Si ricorrerà all'utilizzo della sessione del server per tenere traccia dell'utente loggato.

Le operazioni che gli utenti dell'applicazione web possono effettuare sugli oggetti sono riportate nella tabella che segue:

	Gestione Utenza	Gestione Campionato	Gestione Squadra	Gestione Messaggistica
LA	Login Logout Consultazione dati Rimozione utente Modifica utente	Inserimento campionato Modifica campionato Associazione/rimozione CA Rimozione voti	Inserimento/ modifica/ rimozione squadra Associazione/ rimozione ad/da allenatore Aggiunta/ rimozione giocatore Consultazione squadre	Invio messaggi globali Invio messaggi campionato Invio messaggi singoli
CA*	Login Logout Consultazione dati utenti Creazione inviti	Generazione calendario Modifica campionato Calcolo giornata Caricamento voti	Inserimento/ modifica/ rimozione squadra Associazione ad allenatore Aggiunta giocatore Consultazione squadre	Invio messaggi campionato Invio messaggi singoli Consultazione
Allenatore*	Login Logout Consultazione dati propri	Inserimento formazione Modifica formazione Consultazione	Licenziamento giocatore Consultazione squadre	Invio messaggi singoli Consultazione

*Per questo tipo di utente, tutte le operazioni interessano esclusivamente il campionato (e i suoi Allenatori) di cui egli fa parte.

3.6 Controllo flusso globale

Il controllo di flusso globale adottato è di tipo thread-driven, in quanto Gunicorn è in grado di gestire in maniera concorrente l'interazione tra la webapp e più clients. Più specificamente, ogni richiesta da parte di un utente genera un thread dedicato, attraverso il quale essa sarà gestita.

3.7 Condizioni limite

3.7.1 Startup sistema

Nome Caso Uso	Startup Sistema	
Attori Partecipanti	Amministratore	
Flusso di Eventi	**Amministratore**	**BiaSet**
	1. L'amministratore scrive in console "systemctl start biaset-app"	
		2. Il sistema inizia l'esecuzione del servizio creato precedentemente, avviando tutte le componenti e l'applicazione stessa.
Pre-Condizioni	L'amministratore è connesso tramite una shell al server che ospita il web server. È presente un servizio registrato col nome "biaset-app" che contenga tutte le istruzioni per l'avvio dell'applicazione e dei servizi accessori.	
Post-Condizioni	L'applicazione è avviata e in attesa di connessioni.	

3.7.2 Shutdown Sistema

Nome Caso d'uso	Shutdown Sistema	
Attori Partecipanti	Amministratore	
Flusso di Eventi	**Amministratore**	**BiaSet**
	1. L'amministratore scrive in console "systemctl stop biaset-app"	
		2. Il sistema esegue lo shutdown
Pre-condizioni	L'amministratore è connesso tramite una shell al server che ospita il web server. Il web server è stato precedentemente avviato.	
Post-condizioni	Il sito non è più raggiungibile.	

3.7.3 Fallimento

BiaSet può incorrere in diversi casi di fallimento, riguardanti sia l'hardware che il software:

- **Fallimenti Hardware**: Crash del disco su cui i dati persistenti sono salvati; il sistema non prevede alcuna strategia di backup e ripristino dei dati;
- **Fallimenti nell'ambiente di esecuzione**: Interruzione della fornitura elettrica al server; il sistema non prevede alcuna strategia che ne garantisca l'operabilità in questo tipo di condizione;

- **Fallimenti Software**: Impossibilità di stabilire una connessione col database; una schermata notifica l'errore all'utente.

3.8 Subsystem services

Sottosistema	Descrizione
Gestione Utenza	Consente ad un Utente non registrato di effettuare la registrazione, previo invito, e a un Utente registrato di effettuare il login. Permette a quest'ultimo di accedere alle funzionalità della piattaforma BiaSet, permettendogli di gestire la propria squadra.

Servizio	Descrizione
login	Consente ad un Utente di accedere al sistema tramite un social network a sua scelta.
logout	Consente ad un Utente di rimuovere l'accesso al sistema.
socialRegistration	Consente ad un Utente non registrato di registrarsi alla piattaforma utilizzando un social network.
listUserData	Consente ad un Utente registrato di visualizzare i suoi dati personali e quelli degli altri, se un LA o CA.
removeUser	Consente ad un LA di eliminare un Utente.
modifyUser	Consente ad un CA di modificare un Utente.
inviteUser	Consente ad un CA di invitare un Allenatore al proprio campionato.

Sottosistema	Descrizione
Gestione Campionato	Consente ad un Utente registrato di interagire con il sistema al fine di inserire/modificare/consultare dati relativi a formazioni di match, calendario degli scontri; permette la creazione degli inviti e consente di caricare i voti e calcolare la giornata fantacalcistica.

Servizio	Descrizione
insertFormation	Consente ad un Allenatore o un CA di inserire la formazione per la giornata fantacalcistica in corso.
modifyFormation	Consente ad un Allenatore o un CA di modificare la formazione per la giornata fantacalcistica in corso.
generateMatches	Consente ad un LA/CA di generare il calendario degli scontri di un Campionato specifico.
listMatches	Permette di consultare la lista degli scontri di un campionato.
loadMarks	Consente il caricamento automatico dei voti dei giocatori per la giornata fantacalcistica corrente.
removeChampionship	Consente la rimozione di un campionato.
modifyChampionship	Consente la modifica di un campionato.
linkChampionshipAdmin	Consente di associare un CA ad un campionato.
unlinkChampionshipAdmin	Consente di rimuovere un CA da un campionato.

Sottosistema	Descrizione
Gestione Squadra	Consente ad un Utente registrato di interagire con il sistema al fine di inserire/modificare/consultare dati relativi alle squadre, come l'associazione di un Giocatore o l'associazione di un Allenatore ad una Squadra.

Servizio	Descrizione
insertGiocatore	Consente ad un LA/CA di inserire un Giocatore in una Squadra.
removeGiocatore	Consente ad un LA/CA di rimuovere un Giocatore da una Squadra.
insertSquadra	Consente ad un LA/CA di inserire una nuova squadra in uno specifico campionato.
removeSquadra	Consente ad un LA/CA di rimuovere una nuova squadra da uno specifico campionato.
linkAllenatore	Consente di associare un allenatore ad un campionato.
unlinkAllenatore	Consente di rimuovere un allenatore da un campionato.
fireGiocatore	Consente di licenziare un giocatore (analogo a removeGiocatore)
listSquadra	Permette la consultazione di una specifica squadra.

Sottosistema	Descrizione
Gestione Messaggistica	Consente agli utenti di scambiarsi messaggi privati stile e-mail.

Servizio	Descrizione
sendBroadcastPm	Consente ad un CA di inviare un messaggio a tutti gli utenti della piattaforma.
sendPm	Consente ad ogni Utente di inviare un messaggio privato ad un altro Utente.
sendChampionshipPm	Consente ad un LA/CA di inviare un messaggio a tutti gli utenti di uno specifico campionato
viewMessage	Consente ad un Utente di visualizzare un messaggio.

4. Glossario

Python: linguaggio di programmazione ad alto livello orientato agli oggetti.

Applicazione web: programma accessibile tramite browser web ed in grado di elaborare richieste e risposte.

Throughput: misura della capacità del sistema di condurre task contemporanei.

Server: macchina connessa alla rete dotata di un ambiente di esecuzione.

Application Server: sistema software per la gestione delle richieste/risposte provenienti dai client.

DBMS: sistema software per la gestione dei dati persistenti su database.

MySQL: relational database management system composto da un client a riga di comando e un server.

MVT: Model-View-Template, pattern architetturale per lo sviluppo di sistemi software.

CSS3: terza versione di CSS; è un linguaggio di programmazione web utilizzato per descrivere l'aspetto e la formattazione di un sito web al browser lato client.

HTTP: protocollo a livello applicativo usato come principale sistema per la trasmissione d'informazioni sul web.

Django: framework open source per lo sviluppo di applicazioni su piattaforma Python.

Docker: software per la virtualizzazione a livello di sistema operativo (containerizzazione).