Easy Fut5al

System Design Document

Sommario

1. Introduzione 2
1.1 Obiettivi del Sistema 2
1.2 Design Goals 3
1.3 Definizioni, acronimi e abbreviazioni 4
1.4 Riferimenti 4
1.5 Panoramica 4
2. Architettura del sistema corrente 5
3. Architettura del sistema proposto 5
3.1 Panoramica 5
3.2 Decomposizione in sottosistemi 6
3.2.1 Decomposizione in Layer 6
3.2.2 Decomposizione in sottosistemi 8
3.3 Mapping hardware/software 10
4. Servizi dei Sottisistemi 14

1. Introduzione

1.1 Obiettivi del Sistema

Il sistema da sviluppare ha il compito di semplificare l'organizzazione di partite di calcio a 5, fornendo supporto sia all'atleta e sia al gestore del campetto in cui si svolge una data partita. L'atleta deve essere in grado di creare una nuova partita in modo intuitivo e immediato tramite l'interfaccia del sistema, dovrà ricevere un riscontro da 9 partecipanti ed aggiudicarsi il campo per mezzo di una prenotazione automatica che verrà notificata al gestore, il quale non dovrà far altro che visualizzare sul suo calendario l'elenco delle prenotazioni con i relativi orari. Il sistema prevede inoltre un metodo di autentificazione: è necessario un login (e un logout), una registrazione atleta, una registrazione particolare per il gestore (che specificherà anche i dettagli sul campetto, i quali saranno verificati di persona da una figura addetta, tale amministratore). Sono presenti anche altre funzioni secondarie che consistono nella visualizzazione e modifica di dati: ogni atleta può visualizzare il calendario delle partite per ogni campetto, dare una valutazione di esso al termine della partita e visualizzare il proprio profilo. Il gestore ha la possibilità di vedere le prenotazioni solo sul proprio campo. Il sistema così proposto può essere diviso in sottosistemi in base alle funzionalità:

- *Activity Atleta*
  - **Comprende tutte le attività che si interessano della partecipazione e creazione della partita da parte di un atleta, ossia: visualizzazione delle partite pubbliche a cui è possibile unirsi (Sezione home), funzione "CreaPartita", funzione di ricezione inviti.**
- *Activity Gestore*
- *Activity Autenticazione*
  - **Fornisce la possibilità di controllare il flusso di eventi verso Login, Registrazione Atleta, Registrazione Gestore.**
- *Registrazione (per atleta e per gestore)*
- *Login/Logout*
- *Calendary*
  - **Visualizza le partite prenotate in un dato giorno per un campetto.**
- *Profilo*
  - **Visualizza profilo.**
- *Prenotazioni Management*
  - **Si occupa del check del controllo dello stato delle partite.**
- *ConfermaCampetto*
  - **Consente di confermare o negare l'agibilità di un campetto.**

1.2 Design Goals

Sono identificati i seguenti design goals per il sistema:

- *Criteri di affidabilità:*

Il sistema deve essere robusto: non devono essere ammessi input non validi, i quali saranno notificati tramite messaggi di errore. I dati del campetto sono attendibili poiché sono verificati da figure apposite.
Il sistema deve essere accessibile in qualunque momento onde evitare rallentamenti nell'organizzazione delle partite. Easy Fut5al deve supportare un bacino minimo di 100 profili online.
Il sistema è accessibile 24/24 h da un utente a meno che esso non sia in manutenzione. In quel caso il sistema sarà temporaneamente sospeso per non più di 2 ore.

- *Performance:*

Per quanto riguarda la risposta del sistema nell'inviare notifiche e messaggi ad un utente, il tempo massimo previsto è di 2 minuti.
La verifica da parte dell'amministratore del sistema dell'esistenza effettiva di un campetto, quando viene sottomesso da un gestore non deve superare i 5 giorni lavorativi.

- *Criteri di costo:*

Si stima un costo di 650 ore di lavoro per la progettazione e sviluppo del sistema.

- *Supportabilità:*

Il sistema è dedicato a utenti in suolo italiano, poiché si vogliono registrare solo campetti italiani. Non si esclude un'espansione del sistema al di fuori dei confini nazionali. Al momento il sistema necessita di un unico server.

1.3 Definizioni, acronimi e abbreviazioni

- *EF5:* acronimo che fa riferimento al sistema "Easy Fut5al"
- *Greenfield Engineering*: tipologia di sviluppo software che comincia da zero. Non esiste alcun sistema a priori e i requisiti sono ottenuti dall'utente finale e dal cliente. Quindi, nasce a partire dai bisogni richiesti dall'utente.

1.4 Riferimenti

- Bernd Bruegge & Allen H. Dutoit, *Object-Oriented Software Engineering: Using UML, Patterns and Java*, (3rd edition),Prentice-Hall.
- *Visual Paradigm for UML*

1.5 Panoramica

In questo documento al secondo punto verrà presentata l'architettura del sistema e le eventuali decomposizioni in sottosistemi. Si gestirà poi, al terzo punto, il mapping hardware/software, i dati persistenti, il controllo degli accessi e sicurezza, il controllo del flusso globale del sistema e le condizioni limite. Infine al quarto punto vengono elencati i vari servizi per ogni sottosistema.

2. Architettura del sistema corrente

Non esiste un sistema con le caratteristiche proposte da EF5. L'organizzazione di partite di calcetto avviene comunemente per via telefonica e poiché la disponibilità dei partecipanti può variare, capita spesso che le partite vengano annullate o rimandate.

Quindi si cerca di proporre un'alternativa tecnologica, quale la creazione di questo sistema, alle classiche "prenotazioni telefoniche" per due principali motivi: il primo è quello di invogliare giovani utenti a creare più facilmente una partita con un semplice tocco di cellulare ed il secondo è quello di consentire ad un gestore di campetto ad incrementare i suoi profitti.

Alcuni sistemi cercano di risolvere tali problema ma nessuno si è dimostrato efficiente e realmente applicabile.

Lo sviluppo del nostro sistema non si basa su nessun sistema precedente. Per tal motivo, EF5 è da considerarsi nella categoria *Greenfield Engineering*.

3. Architettura del sistema proposto

3.1 Panoramica

Il sistema in questione è composto da più client, che eseguiranno le proprie mansioni su un app per dispositivi Android, e da un server in locale che si occuperà dell'accesso al database. Più nel dettaglio:

- Il *client* istanzia l'interfaccia grafica dell'app, permette l'esecuzione da parte dell'utente (atleta o gestore che sia) di funzioni di visualizzazione, ricerca, management di partite, notifica etc. … inoltre tiene conto degli eventi che genera quest'ultimo.

- Il *server* si occuperà della gestione dei dati persistenti e tutta la logica relativa ad essi. Tali dati saranno salvati su un database all'interno di un'unica macchina. Alcune funzionalità speciali, sono eseguite sul lato Server come lo scheduling delle partite che si aggiudicano il campetto, il cambiamento di stato di una partita e la verifica d'agibilità del campetto.

  La scelta dell'architettura si dirige verso il "Fat-client", poiché la maggior parte della logica del sistema è insita nel client. Questa euristica è ideale affinché si rispettino i criteri di affidabilità e performance.

L'hardware degli attuali smartphone con Android OS riescono a computare la maggior parte del carico del lavoro che deve svolgere il sistema in modo efficiente.

3.2 Decomposizione in sottosistemi

3.2.1 Decomposizione in Layer

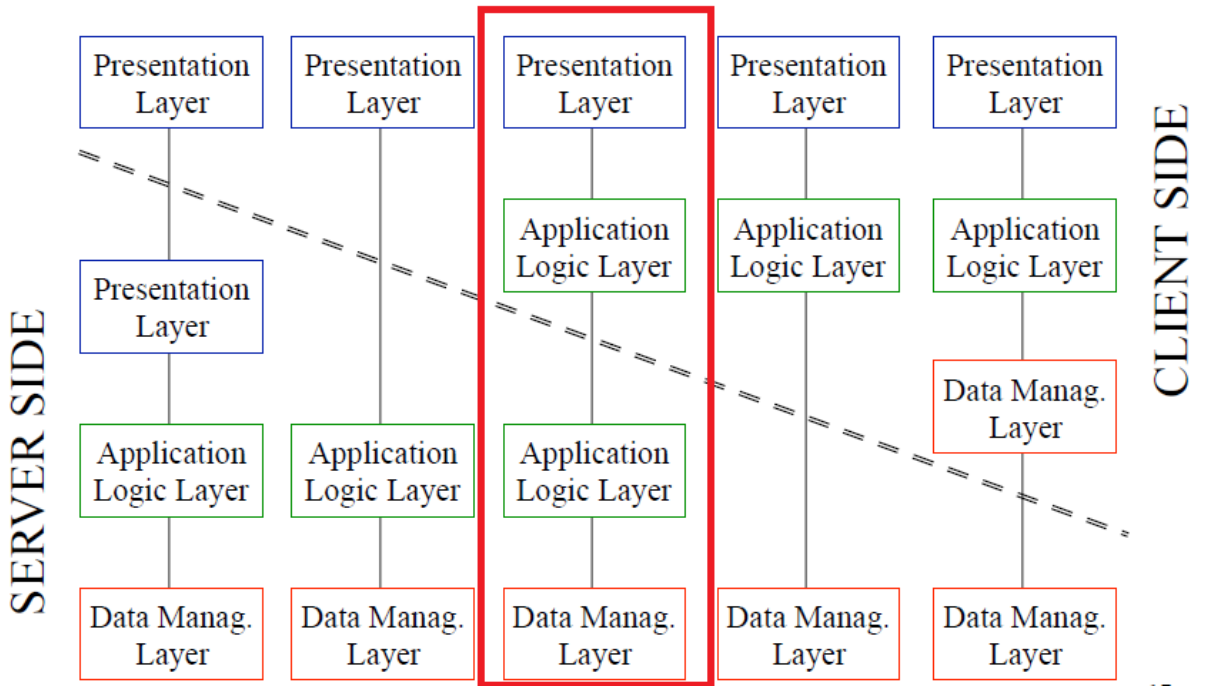
La decomposizione del sistema viene espressa in 3 livelli diversi, tipica dell'architettura *Multi-tier*. In realtà lo strato di Application può essere scisso in ulteriori due layer, l'Application-Logic-client e l'Application-Logic-Server, evidenziando così la natura del sistema distribuito che delega la logica di business a due macchine diverse. Ogni layer assume una gestione diversa per ogni aspetto e funzionalità:

- *Interface:* visualizza l'interfaccia grafica e traccia e gestisce tutti gli eventi generati dall'utente.

- *Application (Application Logic Client+ Application Logic Server):* è lo strato che si occupa della logica dell'intero sistema.

- *Storage:* gestisce lo scambio dei dati persistenti del database tra i vari sottosistemi.

Grazie all'architettura *Multi-tier*, la suddivisione dei compiti è più razionale e ogni funzionalità può essere eseguito come un processo a parte. Per tal motivo riduce lo spreco delle risorse.


SERVER SIDE
CLIENT SIDE
Presentation Layer
Application Logic Layer
Data Manag. Layer

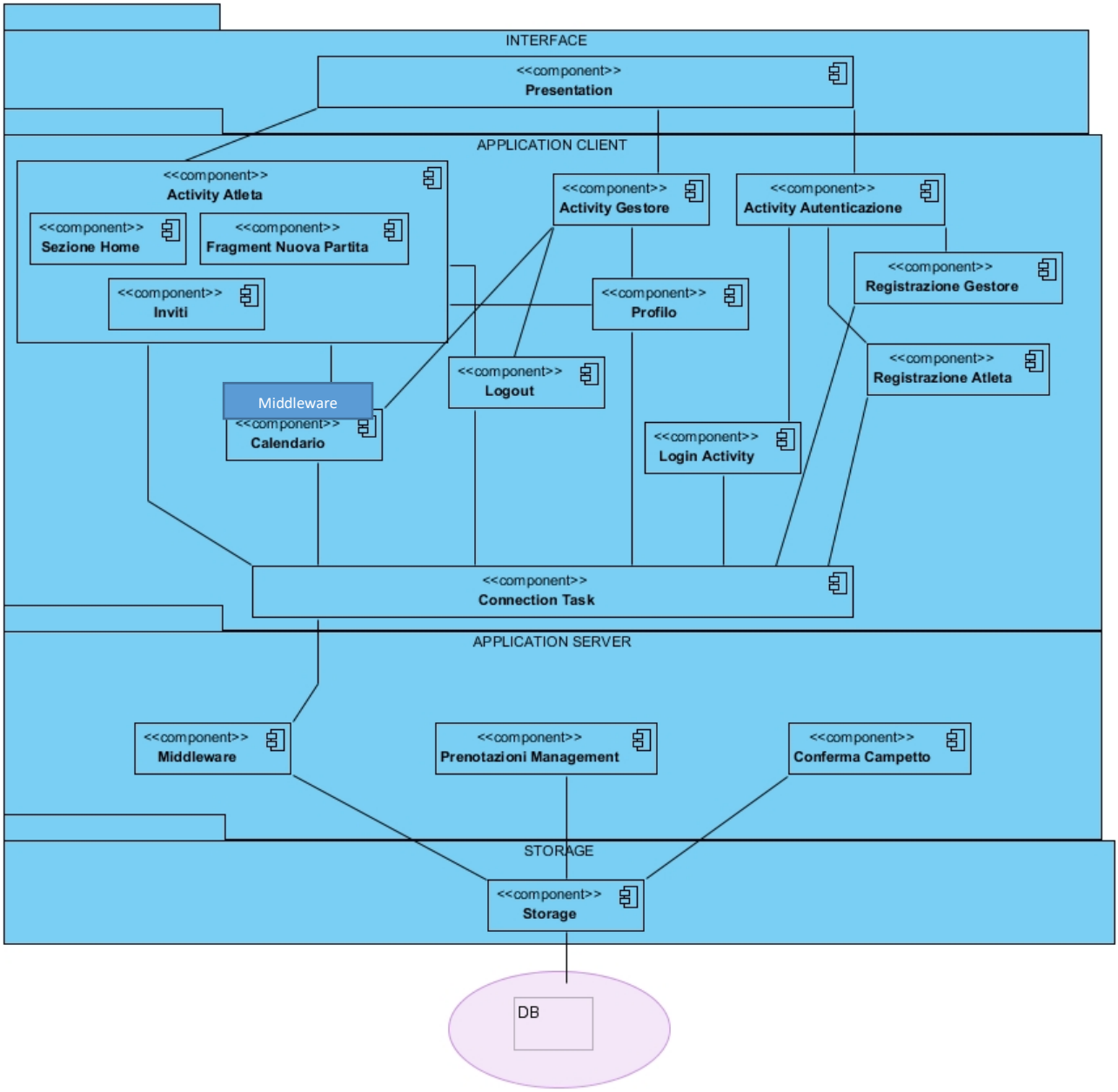
3.2.2 Decomposizione in sottosistemi

Il sistema è stato partizionato in diversi componenti a fronte del rispetto del principio di *coesione* e di *accoppiamento*. Si è fatto in modo che i sottosistemi siano *loosely coupled*, ossia che le modifiche apportate ad un sottosistema abbiano poco impatto con un altro sistema. Siffatto abbiamo frapposto un componente *Storage* per evitare un arduo cambiamento dei sottosistemi in caso di cambio DB. E' stata necessaria l'introduzione anche di altri component come:

Connection Task: risiede nel layer Application-Logic-Client mette a fattor comune tutte le funzionalità di connessione al Server, fornendo dei servizi che comunicano con il layer Application-Logic-Server sottostante.

Middleware: risiede nello strato Application-Server e riceve dall'Application-Logic-Client le richieste e le reinterpreta in base alle specifiche software del layer Storage sottostante.

La parte *server* è installata su una sola macchina e ingloba lo strato di *Storage* e Application-Logic-Server, quindi anche la verifica della veridicità del campetto (un'equipe specializzata farà la dovuta indagine e il risultato verrà annunciato all'amministratore dell'unica macchina, il quale inserirà l'esito) e il management delle prenotazioni. Quest'ultime diventano effettive quando 10 membri avranno confermato la partecipazione alla partita, in più bisogna gestire lo scheduling della coda delle partite in caso di prenotazione di un campo nello stesso orario per diverse partite.


INTERFACE
<<component>>
Presentation
APPLICATION CLIENT
<<component>>
Activity Atleta
<<component>>
Sezione Home
<<component>>
Fragment Nuova Partita
<<component>>
Inviti
<<component>>
Activity Gestore
<<component>>
Activity Autenticazione
<<component>>
Registrazione Gestore
<<component>>
Profilo
<<component>>
Registrazione Atleta
Middleware
<<component>>
Logout
<<component>>
Calendario
<<component>>
Login Activity
<<component>>
Connection Task
APPLICATION SERVER
<<component>>
Middleware
<<component>>
Prenotazioni Management
<<component>>
Conferma Campetto
STORAGE
<<component>>
Storage
DB

Sottosistemi per Livelli:

Livello Interface:

Presentation

Livello Application:

- *Client side:*

  Login/Logout
  Registrazione Atleta/ Registrazione Gestore
  Activity Atleta (comprende Sezione Home, FragmentNuovaPartita, Inviti)
  Activity Gestore
  Activity Autenticazione
  Profilo
  Calendario
  ConnectionTask

- *Server side:*

  Middleware
  Prenotazioni Management
  Conferma Campetto

Livello Storage:

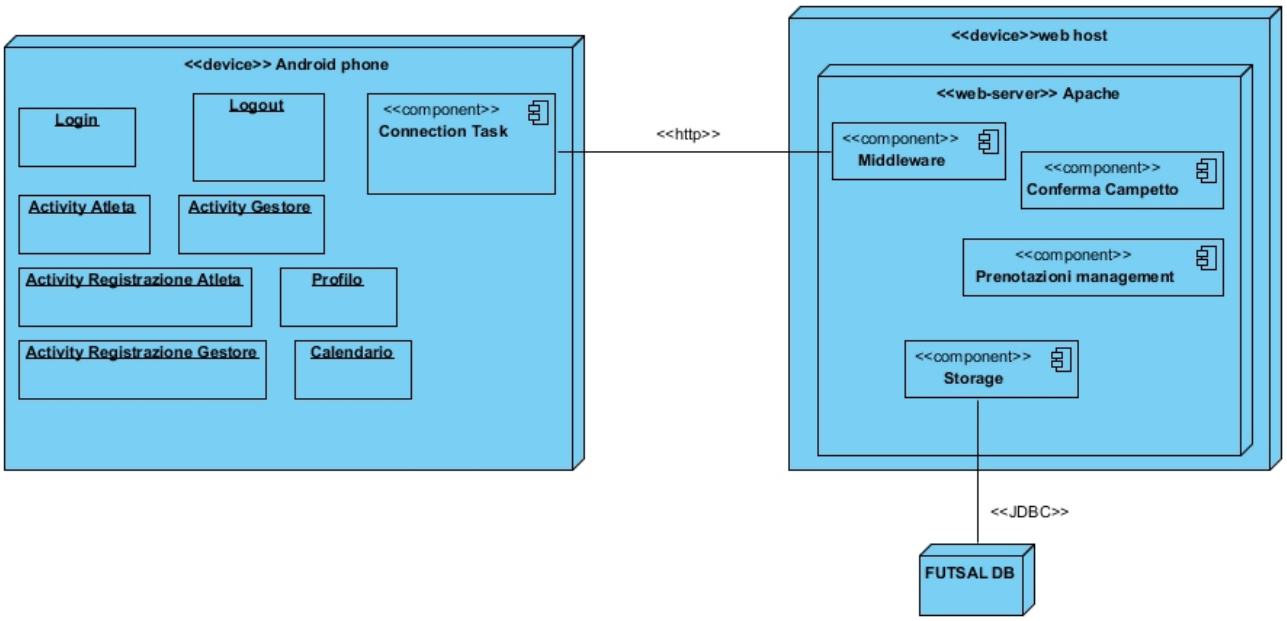
Storage

3.3 Mapping hardware/software

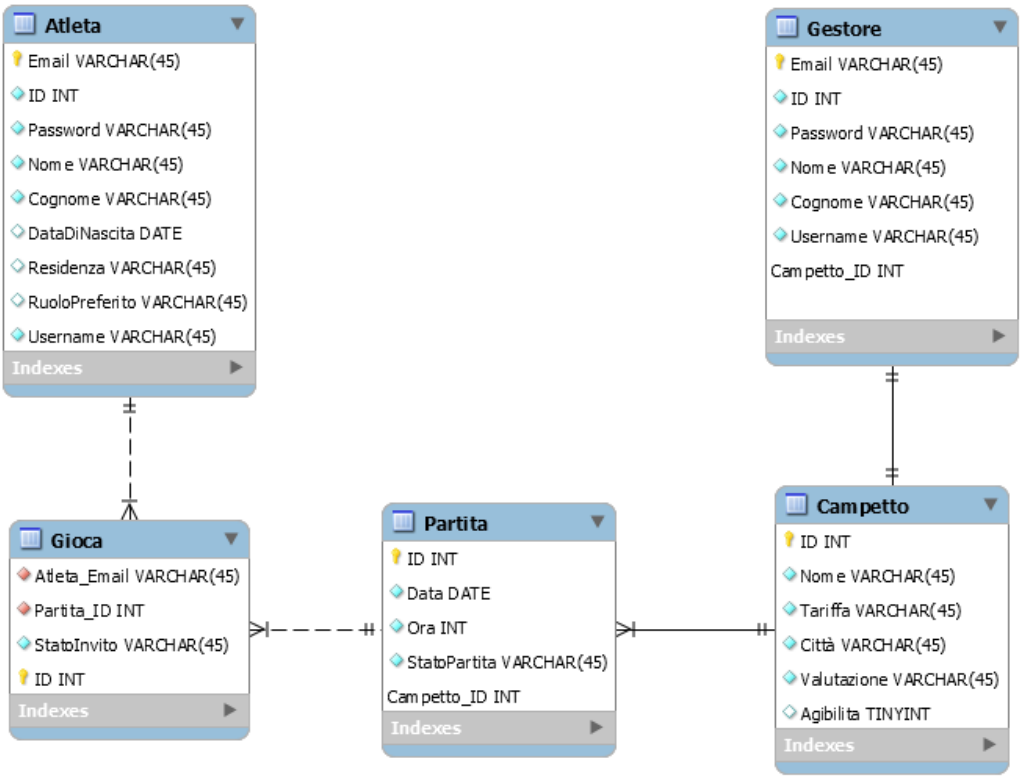
Il sistema, come detto prima avrà una parte server composta da una sola macchina ed utilizzerà il *DBMS MySQL* installato su di esso. Si utilizza un DBMS poiché vi saranno accessi concorrenti (in tal caso esso effettuerà controlli di consistenza e concorrenza) da gestire e anche perché in tal modo è possibile accedere ai dati da varie piattaforme.

I vari client corrispondono ai vari dispositivi Android di ogni cliente, i quali accederanno e utilizzeranno i servizi dedicati per mezzo dell'app scaricata da Play Store.

La logica *Fat-client* permette di distribuire le computazioni "che possono essere eseguite localmente" su ogni dispositivo Android.



3.4 Gestione dati persistenti



3.5 Controllo degli accessi e sicurezza

E' stato necessario stabilire delle politiche di accesso alle informazioni. Ogni cliente, gestore o atleta che sia, dispone di una username e una password.

Operazioni consentite dall'utente:

Attore\Sottosistema	***Activity Autenticazione***	***Activity Atleta***	***Activity Gestore***	***Calendario***	***Profilo***	***Prenotazioni Management***	***Conferma Campetto***
Atleta	*Login* *Registrazione atleta*	*Menu Atleta* *Sezione Home* *Nuova Partita* *Unisciti partita* *Inviti*		*Visualizza partite disponibili per un campetto specificato*	*visualizza profilo.*		
Gestore	*Login* *Registrazione gestore*		*Menu gestore*	*Visualizza partite disponibili per il proprio campetto.*	*visualizza profilo.*		
Amministratore							*Dichiara agibilità campetto.*

3.6 Controllo flusso globale

Per come è stato concepito il sistema, il tipo di controllo di flusso è tramite threads. Questo perché alcune operazioni devono essere eseguite in modo concorrente per garantire la massima efficienza.

3.7 Condizioni limite

Vanno elaborati nuovi casi d'uso che specificano operazioni invocate dall'amministratore del sistema.

Casi d'uso per amministratore.

Caso d'uso	*Descrizione*
StartUp Fut5al	*L'amministratore è tenuto a configurare un web server ed un servizio di Database MySQL, inizialmente vuoto, per Easy Fut5al. L'amministratore dà l'avvio al sistema.*
ManageData	*L'amministratore accede con le sue credenziali al database MySQL e modifica, cancella, aggiunge un oggetto persistente nel database.*
ReStart	*Possono esserci guasti dovuti al sovraccarico del DB. Per la salvaguardia del sistema, l'amministratore deve periodicamente salvare un file di copia dei dati nel DB prima di effettuare un normale StartUp.*
ShutDown	*L'amministratore ferma il Server, salva le ultime modifiche avvenute nel sistema. Tutti gli altri utenti non possono più usufruire di Easy Fut5al.*
ConfermaCampetto	*//già trattato*

Vi possono essere alcune cause di fallimento:

- *Sovraccarico del DB:* è previsto un salvataggio dei dati ed un successivo riavvio del sistema.
- *Alimentazione interrotta:* nessuna misura prevista.
- *Errori del software:* si consiglia il riavvio del sistema.
- *Malfunzionamenti Hardware:* nessuna misura prevista.

4. Servizi dei Sottisistemi