FOOTBALL MANAGEMENT SYSTEM

Sistema per il raccoglimento di informazioni sportive e vendita biglietti per eventi sportivi

Specifica dei Requisiti

Modena

Monelli Filippo matr. 108856

Dichiaro che questo elaborato è frutto del mio personale lavoro, svolto sostanzialmente in maniera individuale e autonoma.

Indice

3 Introduzione

3.1 Requisti funzionali

3.2 Requisiti non funzionali

4 Appendice

4.1 Modellazione UML

4.1.1 Class Diagram
4.1.2 Activity Diagram
4.1.3 Sequence Diagram

4.2 Design Pattern

4.2.1 Strategy
4.2.2 Command
4.2.3 Composite

1 Introduzione

La presente sezione ha lo scopo di riportare la visione globale dell'intero documento di Specifica dei Requisiti. Per la stesura si è seguito lo standard ANSI/IEEE 830 noto come SRS (*Software Requirements Specifications).*

1.1 Obiettivo

L'obiettivo del documento è di fornire una rappresentazione corretta della soluzione pensata per un' applicazione destinata alla raccolta di informazioni sportive tramite database, alla creazione di eventi sportivi, all'acquisto di biglietti per le competizioni presenti nel sistema. Tale rappresentazione deve essere:

- Corretta (ogni requisito riportato è tra quelli che deve essere soddisfatto dal Software)
- Coerente (deve esserci coerenza tra requisito ed esigenza del software)
- Non ambigua (ogni requisito riportato ha una sola interpretazione)
- Completa (deve riportare tutti i requisiti significativi, nulla deve essere "da determinare", ma al contrario deve essere definito e completo)
- Consistente (non devono esserci conflitti interni tra requisiti)
- Classificata (per importanza dei requisiti riportati)
- Verificabile (ogni requisito deve poter essere verificato)
- Modificabile (deve avere una struttura che consenta la flessibilità alle modifiche dei requisiti in modo semplice e completo in modo da mantenerne la struttura)
- Non ridondante
- Tracciabile (si deve poter identificare l'origine di ogni requisito riportato)

1.2 Campo di applicazione

Il prodotto che ci apprestiamo a sviluppare è una applicazione destinata alla raccolta di informazioni calcistiche riguardati: giocatori, società,stadi e partite. Permetterà la creazione di nuove competizioni, fornendo agli utilizzatori la possibilità di acquistare biglietti per queste e generando automaticamente il/i ticket/s in formato pdf con codice univoco identificativo del biglietto. Infine di esportare le informazioni presenti sul sistema come foglio di calcolo in locale. Ogni operazione che verrà svolta attraverso il prodotto deve essere registrata su tabelle presenti nel database del sistema.
L'applicazione deve supportare 3 tipi di utenti:

- **Client**
- **Manager**
- **Administator**

le cui funzionalità variano in base al tipo di profilo con cui si accede al sistema.
Per accedere al sistema sarà necessario usufruire di "username" e "password", nel caso in cui non si sia in possesso di queste è possibile crearli tramite una operazione di registrazione. Nel caso in cui si voglia creare un profilo "Manager" o "Administrator" sarà necessario possedere la password corretta per accedere alla schermata di registrazione. Nel caso di un nuovo profilo "Client" l'accesso alla schermata di registrazione sarà libero.

1.3 Definizioni,acronimi e abbreviazioni

Client	Utente registrato al sistema: ha la possibilità di aggiungere e/o aggiornare le informazioni presenti all'interno del sistema, esportarle su un foglio di calcolo in locale. Può visualizzare tutti i biglietti acquistati e acquistarli per gli eventi sportivi disponibili.
Manager	Utente registrato con possibilità aggiuntive rispetto all'utente di tipo client. Ha la possibilità di
Admin	Utente amministratore del sistema.

1.4 Fonti

- SIAER- settore paghe

1.5 Struttura del documento

Il documento prosegue con i 2 seguenti capitoli:

- **Descrizione generale**: in questo capitolo si affrontano temi legati alle macro-funzionalità del sistema unitamente a caratteristiche legate ai requisiti da analizzare in futuro.
- **Specifiche funzionali e non funzionali del sistema**: vengono esposti ed analizzati i requisiti funzionali (quali sono più nello specifico le funzionalità del sistema, ciò che il sistema deve fare) e non funzionali del prodotto (aspetti del sistema che non sono direttamente correlati al suo funzionale comportamento, come ad esempio performance, usabilità…).

2 Descrizione generale

2.1 Inquadramento

2.1.1 Interfaccia sistema/utente

L'interfaccia di interazione tra l'applicazione e l'utente deve privilegiare innanzitutto la semplicità e l'intuitività. Ogni azione deve poter essere fatta con il minor numero di click possibili e deve essere guidata: opportuni messaggi devono comparire nei momenti appropriati per guidare l'utente nelle operazioni da svolgere.
Ogni elemento presente nella schermata deve essere consono alla sua funzione, i bottoni presenti dovranno essere esplicativi e dovranno riportare chiaramente il loro nome, dal quale si può dedurre la funzione corrispondente. Non è necessario concordare i nomi utilizzati per funzioni e componenti,
purché siano universalmente comprensibili e non ambigui.
Assolutamente da evitare l'accesso ridondante alle funzioni, ovvero la possibilità di fare

un'azione in più modi: ogni azione, al fine dell'intuitività, deve doversi svolgere in modo unico e più naturale possibile, così da non creare ambiguità o incongruenze.
Per agevolare l'utilizzo dell'applicazione si è deciso di seguire lo "standard de facto" che si vede utilizzando una qualsiasi applicazione, introducendo il menu di navigazione nella parte in alto della schermata principale.

2.1.2 Interfaccia hardware(N/A)

2.1.3 Interfaccia software (N/A)

2.1.4 Interfaccia di comunicazione (N/A)

2.1.5 Vincoli relativi all'occupazione di memoria (N/A)

2.1.6 Operazioni (N/A)

2.1.7 Vincoli per l'installazione

Per poter installare l'applicazione è necessaria installare preliminarmente:

- **Java Virtual Machine** scaricabile dal sito: https://www.java.com/download/
- **MySQL** scaricabile dal sito: https://dev.mysql.com/downloads/mysql/

2.2 Macro funzionalità del sistema

Il sistema si da carico di:

- Gestione delle informazioni inserite nel database
- Gestione delle informazioni degli utenti iscritti al sistema
- Organizzazione di nuovi eventi sportivi
- Vendita biglietti
- Esportazione delle informazioni in locale

2.3 Caratteristiche degli utenti

L'utente è il soggetto che interagisce con il sistema. Questo agisce con il software per ogni operazione che deve essere svolta.
In base al profilo utente che questo possiede le operazioni che possono essere svolte saranno più o meno limitate.

2.4 Vincoli generali

È necessario che il sistema su cui si intende utilizzare il software abbia la Java Virtual Machine configurata e funzionante.
È necessario aver installato MySQL sul proprio dispositivo, è sufficiente MySQL Community.
Solamente per administrator è necessario installare e configurare correttamente MySQL workbench per poter lavorare direttamente tramite DBMS sul database del sistema

2.5 Assunzioni e dipendenze (ipotesi iniziali)

- **Amministratore rilascio password per accesso al sistema:** l'amministratore, che si presume rappresenti l'utente con piena libertà all'interno del sistema, ha il compito di fornire le password per la registrazione di nuovi utenti amministratore e manager.
- **Manager riconosciuti:** per poter creare un nuovo account manager l'utente dovrà dimostrare, attraverso la consegna di documenti, di appartenere ad una società sportiva. Questo permetterà di verificare il corretto inserimento di nuovi matches all'interno dell'applicazione.

2.6 Requisiti da analizzare in futuro

I requisiti da inserire in futuro sono:

- Eliminare la dipendenza da software di terze parti per l'utilizzo dell'applicazione
- Espandere le estensioni possibili per il salvataggio dei fogli di calcolo
- Migrare l'intera applicazione su browser web tramite la creazione di un sito su cui riprodurre tutta l'applicazione
- Aggiungere la parte di gestione completa del database senza applicazione
- Migliorare la creazione automatica dei tickets inserendo un QR-code / codice a barre collegato al codice univoco sul biglietto permettendo così di poter stampare a casa o utilizzare lo smartphone per accedere agli eventi sportivi.

3 Specifiche funzionali e non funzionali del sistema

3.1 Requisiti funzionali

3.1.1 Procedura di login

RF01	**Schermata iniziale**	**Procedura di login**
Input	Username e password	
Processo	L'utente accede al sistema con il suo identificativo, attiva la procedura di login tramite l'apposito pulsante. Messaggio di errore se l'operatore ha inserito dati errati oppure dati non presenti all'interno del database del sistema.	
Output	Se l'operatore è registrato, viene visualizzata la schermata di home dell'applicazione.	

3.1.2 Procedura di registrazione tramite l'apposita schermata

RF02	**Schermata iniziale**	**Procedura di registrazione**
Input	Nome, cognome, username, password, data di nascita, immagine del profilo	
Processo	L'utente accede alla schermata di registrazione scegliendo il tipo account che si vuole creare, le procedure di registrazione amministratore e manager potette da password fornita da un	

	amministratore del sistema. L' utente attiva la procedura di registrazione tramite l'apposito pulsante. L'utente deve inserire le informazioni riguardanti: • name • surname • username (univoco all'interno di tutto il sistema) • password • data di nascita • immagine del profilo (si può scegliere tra quelle presenti all'interno del sistema) Messaggio di errore se l'operatore ha inserito un username già presente all'interno del database dell'applicazione.
Output	Al termine della fase di registrazione viene visualizzata la procedura di login.

3.1.3 Visualizzazione schermata aggiornamento dati personali

RF03	**Schermata applicazione**	**Menù "Home"**
Input	Click sul bottone "Update my info"	
Processo	L'utente può aggiornare le proprie informazioni. L'aggiornamento si basa sulla registrazione modificando semplicemente l'operazione che viene svolta sul database.	
Output	Se l'operatore ha concluso l'aggiornamento, corretta visualizzazione delle informazioni all'interno dell'applicazione.	

3.1.4 Procedura di disconnessione dal sistema

RF04	**Schermata applicazione**	**Menù "Home"**
Input	Click sul bottone "Logout "	
Processo	L'utente si disconnette dal sistema.	
Output	Visualizzazione schermata di login.	

3.1.5 Procedura di inserimento nuovo giocatore

RF05	**Menu giocatori**	**Procedura di inserimento**
Input	Click sul bottone "Add"	
Processo	L'utente inserisce le informazioni: • nome • cognome • nazionalità • squadra (deve essere già presente all'interno del sistema) • ruolo	

	Infine deve attribuire un valore da 0 a 10 (compresi) per ognuno dei seguenti campi: • Fisicità • Velocità • Capacità mentale • Attacco • Difesa • Tecnica Si completa il processo cliccando sul bottone“Register”
Output	Nuovo giocatore inserito nella tabella del menu e nel database del sistema.

3.1.6 Procedura di aggiornamento informazioni di un giocatore

RF06	**Menu giocatori**	**Procedura di aggiornamento**
Input	Selezione di una giocatore presente in tabella e click sul bottone “Update”.	
Processo	L’utente può aggiornate i campi sopra elencati nel punto 3.1.5. Per terminare la procedura l’utente deve cliccare su “Update”.	
Output	Visualizzazione degli attributi aggiornati sulla tabella e aggiornamento dei dati nel database.	

3.1.7 Procedura di cancellazione di un giocatore dal sistema (operazione non permessa a utenti “Client”)

RF07	**Menu giocatori**	**Procedura di cancellazione**
Input	Selezione di una giocatore presente in tabella e click sul bottone “Delete”.	
Processo	L’utente seleziona il giocatore che desidera eliminare e lo elimina.	
Output	Eliminazione della linea selezionata dalla tabella e dei dati nel database.	

3.1.8 Procedura di esportazione della tabella

RF08	**Menu giocatori**	**Procedura di esportazione**
Input	Click su tasto “Export”	
Processo	L’utente esporta la tabella come foglio di calcolo potendo scegliere il path di destinazione. Il foglio di calcolo può avere due tipi di estensioni “*.xlsx” e “*.xls”.	
Output	Visualizzazione menu giocatore, tabella esportata nell’indirizzo scelto e nel formato selezionato. Messaggio positivo in caso di riuscita esportazione dei dati	

	altrimenti messaggio di errore.

3.1.9 Procedura di inserimento nuova squadra

RF09	**Menu squadre**	**Procedura di inserimento**
Input	Click sul bottone “Add”	
Processo	L’utente inserisce le informazioni: • nome squadra • nazione • stato (professionista o dilettante) • Valore: valore numerico per definire il valore complessivo della società • stadio (deve essere già inserito nel sistema) • anno di fondazione Infine deve attribuire un valore da 0 a 5(compresi) per ognuno dei seguenti campi: • Gestione • Stabilità • Supporto della tifoseria Si completa il processo cliccando sul bottone“Register a new team”	
Output	Nuova società inserita nella tabella del menu e nel database del sistema.	

3.1.10 Procedura di aggiornamento informazioni di una squadra

RF10	**Menu squadre**	**Procedura di aggiornamento**
Input	Selezione di una società presente in tabella e click sul bottone “Update”.	
Processo	L’utente può aggiornate i campi sopra elencati nel punto 3.1.9. Per terminare la procedura l’utente deve cliccare su “Update”.	
Output	Visualizzazione degli attributi aggiornati sulla tabella e aggiornamento dei dati nel database.	

3.1.11Procedura di cancellazione di un giocatore dal sistema (operazione non permessa a utenti “Client”)

RF11	**Menu squadre**	**Procedura di cancellazione**
Input	Selezione di una società presente in tabella e click sul bottone “Delete”.	
Processo	L’utente seleziona la società che desidera eliminare e lo elimina.	
Output	Eliminazione della linea selezionata dalla tabella e dei dati nel database.	

3.1.12 Procedura di esportazione della tabella come foglio di calcolo

RF12	**Menu squadre**	**Procedura di esportazione**
Input	Click su tasto “Export”	
Processo	L’utente esporta la tabella come foglio di calcolo potendo scegliere il path di destinazione. Il foglio di calcolo può avere due tipi di estensioni “*.xlsx” e “*.xls”.	
Output	Visualizzazione menu squadre, tabella esportata nell’indirizzo scelto e nel formato selezionato. Messaggio positivo in caso di riuscita esportazione dei dati altrimenti messaggio di errore.	

3.1.13 Procedura di inserimento nuovo utente (operazione riservata ad amministratori del sistema)

RF05	**Menu utenti**	**Procedura di inserimento**
Input	Click sul bottone “Add”, selezione del tipo di utente da creare	
Processo	L’Amministratore deve inserire le informazioni presenti nel punto 3.1.2. Per terminare l’inserimento utilizza il tasto “Register”.	
Output	Nuovo utente inserito nella tabella dell’applicazione e nel database del sistema.	

3.1.14 Procedura di aggiornamento informazioni di un utente (operazione riservata ad amministratori del sistema)

RF14	**Menu utenti**	**Procedura di aggiornamento**
Input	Selezione di una giocatore presente in tabella e click sul bottone “Update”.	
Processo	L’utente può aggiornate i campi sopra elencati nel punto 3.1.2. Per terminare la procedura l’utente deve cliccare su “Update”.	
Output	Visualizzazione degli attributi aggiornati sulla tabella e aggiornamento dei dati nel database.	

3.1.15 Procedura di cancellazione di un utente dal sistema (operazione riservata ad amministratori del sistema)

RF15	**Menu utenti**	**Procedura di cancellazione**
Input	Selezione di una giocatore presente in tabella e click sul bottone “Delete”.	
Processo	L’utente seleziona l’utente che desidera eliminare e lo elimina. Nel caso l’utente eliminato fosse quello che ha effettuato l’accesso questo verrà disconnesso automaticamente dopo la	

	cancellazione dei dati dal database.
Output	Eliminazione della linea selezionata dalla tabella e dei dati nel database. Nel caso di cancellazione dell'utente collegato visualizzazione della schermata di login.

3.1.16 Procedura di esportazione della tabella

RF16	**Menu utenti**	**Procedura di esportazione**
Input	Click su tasto "Export"	
Processo	L'utente esporta la tabella come foglio di calcolo potendo scegliere il path di destinazione. Il foglio di calcolo può avere due tipi di estensioni "*.xlsx" e "*.xls".	
Output	Visualizzazione menu utente, tabella esportata nell'indirizzo scelto e nel formato selezionato. Messaggio positivo in caso di riuscita esportazione dei dati altrimenti messaggio di errore.	

3.1.17 Procedura di inserimento nuova partita (operazione non permessa a utenti "Client")

RF17	**Menu partite**	**Procedura di inserimento**
Input	Click sul bottone "Add"	
Processo	L'utente inserisce le informazioni: • Squadra in casa • Squadra in trasferta • Stadio (deve essere già presente nel sistema) • data dell'evento Si completa il processo cliccando sul bottone"Add match"	
Output	Nuova partita inserita nella tabella dell'applicazione e nel database del sistema.	

3.1.18 Procedura di aggiornamento informazioni di una nuova partita (operazione non permessa a utenti "Client")

RF18	**Menu partite**	**Procedura di aggiornamento**
Input	Selezione di una partita presente in tabella e click sul bottone "Update".	
Processo	L'utente può aggiornate i campi sopra elencati nel punto 3.1.17. Per terminare la procedura l'utente deve cliccare su "Update".	
Output	Visualizzazione degli attributi aggiornati sulla tabella e aggiornamento dei dati nel database.	

3.1.19 Procedura di cancellazione di una partita dal sistema (operazione non permessa a utenti "Client")

RF19	**Menu partite**	**Procedura di cancellazione**
Input	Selezione di una partite presente in tabella e click sul bottone "Delete".	
Processo	L'utente seleziona la partita che desidera eliminare e lo elimina.	
Output	Eliminazione della linea selezionata dalla tabella e dei dati nel database.	

3.1.20 Procedura di esportazione della tabella

RF20	**Menu partite**	**Procedura di esportazione**
Input	Click su tasto "Export"	
Processo	L'utente esporta la tabella come foglio di calcolo potendo scegliere il path di destinazione. Il foglio di calcolo può avere due tipi di estensioni "*.xlsx" e "*.xls".	
Output	Visualizzazione schermata di login.	

3.1.21 Procedura di visualizzazione delle partite in base allo stadio selezionato

RF21	**Menu partite**	**Procedura di visualizzazione**
Input	Selezione di una voce presente nel menù a tendina composto da tutti gli stadi presenti all'interno del database e l'opzione per visualizzare tutte le partite.	
Processo	L'utente fa una scelta nel menu a tendina.	
Output	Viene aggiornata la tabella delle partite in base alla scelta dell'utente	

3.1.22 Procedura di visualizzazione delle partite future cui è possibile acquistare biglietti

RF22	**Menu partite**	**Procedura di visualizzazione**
Input	Click su tasto "Matches not yet started"	
Processo	L'utente clicca sul bottone.	
Output	Viene aggiornata la tabella delle partite e viene abilitato il tasto "Buy tickets"	

3.1.23 Procedura di acquisto biglietti della partita selezionata

RF23	**Menu partite**	**Procedura di acquisto**
Input	Selezione di una partita della tabella e click su tasto "Buy tickets"	
Processo	L' utente selezione la partita di cui vuole comprare i biglietti, dichiara quanti biglietti vuole acquistare (un numero compreso	

	tra 1 e 4). Ora l'utente può scegliere il path in cui salvare il file "*.pdf" (nominandolo a piacere) contente i suoi biglietti per l'evento precedentemente selezionato. Messaggio di errore nel caso di un errore nell'operazione.
Output	Visualizzazione schermata di

3.1.24 Visualizzazione tabella dei biglietti venduti dal sistema (operazione riservata ad utenti amministratori)

RF24	**Menu biglietti**	**Procedura di inserimento**
Input	Click sul bottone del menu "Tickets"	
Processo	L' amministratore con un click sul pulsante visualizza la tabella contente tutti i biglietti venduti e le relative informazioni	
Output	Visualizzazione della tabella	

3.1.25 Visualizzazione tabella dei biglietti acquistati tramite l'applicazione

RF25	**Menu biglietti**	**Procedura di inserimento**
Input	Click sul bottone del menu "Tickets"	
Processo	L' utente (cliente e manager) con un click sul pulsante visualizza la tabella contente tutti i biglietti da lui acquistati con e relative informazioni.	
Output	Visualizzazione della tabella	

3.1.26 Procedura di esportazione della tabella biglietti

RF26	**Menu biglietti**	**Menù "Tickets"**
Input	Click su tasto "Export"	
Processo	L'utente esporta la tabella come foglio di calcolo potendo scegliere il path di destinazione. Il foglio di calcolo può avere due tipi di estensioni "*.xlsx" e "*.xls".	
Output	Visualizzazione schermata con tabella biglietti	

3.1.27 Procedura di inserimento nuovo stadio (operazione non permessa a utenti "Client")

RF27	**Menu stadi**	**Procedura di inserimento**
Input	Click sul bottone "Add"	
Processo	L'utente inserisce le informazioni: • nome stadio • posti a sedere • città	

	• anno di fondazione Si completa la procedura cliccando sul pulsante “Insert a new stadium”
Output	Nuovo stadio inserito nella tabella dell’applicazione e nel database del sistema.

3.1.28 Procedura di aggiornamento informazioni di uno stadio

RF28	**Menu stadi**	**Procedura di aggiornamento**
Input	Selezione di uno stadio presente in tabella e click sul bottone “Update”.	
Processo	L’utente può aggiornate i campi sopra elencati nel punto 3.1.27. Per terminare la procedura l’utente deve cliccare su “Update”	
Output	Visualizzazione degli attributi aggiornati sulla tabella e aggiornamento dei dati nel database.	

3.1.29 Procedura di cancellazione di uno stadio dal sistema (operazione non permessa a utenti “Client”)

RF29	**Menu stadi**	**Procedura di cancellazione**
Input	Selezione di uno stadio presente in tabella e click sul bottone “Delete”.	
Processo	L’utente seleziona lo stadio che desidera eliminare e lo elimina.	
Output	Eliminazione della linea selezionata dalla tabella e dei dati nel database.	

3.1.30 Procedura di esportazione della tabella

RF30	**Menu stadi**	**Menù “Stadiums”**
Input	Click su tasto “Export”	
Processo	L’utente esporta la tabella come foglio di calcolo potendo scegliere il path di destinazione. Il foglio di calcolo può avere due tipi di estensioni “*.xlsx” e “*.xls”.	
Output	Visualizzazione schermata di login.	

3.2 Requisiti non funzionali

In questa sezione si vuole rappresentare ed analizzare le caratteristiche non funzionali emerse dalla fase di analisi dei bisogni dell’utente raccolte durante i vari incontri.

RN01	Requisiti prestazionali	Tempi di risposta
Descrizione	Il software deve consentire all'utente di ottenere risposte in tempo reale durante le operazioni che interagiscono con il database inoltre è indispensabile disporre di una connessione ad internet per poter effettuare un collegamento al database attraverso l'applicazione scaricata ed installata in locale. È necessario disporre di un computer con buone prestazioni.	

RN02	Database	Banca dati
Descrizione	Il software deve mettere a disposizione un sistema in grado di garantire la persistenza delle informazioni poi deve garantire la consistenza dei dati. È inoltre opportuno garantire la confidenzialità dei dati attraverso il DBMS. Per quanto concerne gli utenti, ci saranno tre profili: amministratore (permessi di inserimento, visualizzazione, modifica e cancellazione), manager (permessi di inserimento, visualizzazione, modifica) e utente esterno (permessi di inserimento limitato, visualizzazione, modifica). Le altre tabelle che deve essere generate sono: • countries → tabella contente tutte le nazioni del mondo • football matches → tabella contente le partite dell'applicazione • player → tabella contente i giocatori • team → tabella contente le squadre presenti nel sistema • ticket → tabella contente i biglietti venduti attraverso il sistema Si effettua la scelta di adottare il DBMS MySQL community.	

RN03	Attributi del sistema	Sicurezza
Descrizione	La protezione del software da accessi accidentali o malintenzionati, da modifiche, distruzioni o divulgazioni dei dati presenti, occorre utilizzare tecniche di crittografia. Vista la sensibilità dei dati presenti all'interno del database è necessario effettuare aggiornamenti settimanali al server su cui verrà creato il database. Verrà svolto giornalmente un backup dei dati presenti nel database su dischi seguendo la tecnica di RAID 1 (ridondanza completa dei dati).	

RN04	**Altri requisiti**	**Scalabilità**
Descrizione	Con probabilità il sistema in futuro verrà utilizzato da un numero sempre maggiore di utenti quindi è necessario che l'upgrade sia gestibile con costi e tempi ridotti, sia a livello hardware sia software.	

4 Appendice

4.1 Modellazione UML

Breve introduzione

L'UML (Unified Modeling Language) è una famiglia di notazioni grafiche che aiuta nella descrizione e progettazione di sistemi software (in particolare per i sistemi creati utilizzando lo stile orientato agli oggetti (OO)). Molti utilizzano l'UML sia per il suo utilizzo concettuale, poiché rappresenta i concetti principali del software, ma anche perché permette la modellazione del software stesso. Nonostante sia abbastanza preciso, l'UML è un linguaggio descrittivo. Non si può guardare un diagramma UML e capire esattamente quale sia la corrispondente realizzazione nel codice, tuttavia ci si può fare una idea di come funzioni. L'UML ha regole descrittive, da un diagramma UML non si può mai dedurre nulla per assenza.

4.1.1 Class Diagram

Il primo diagramma che decidiamo di modellare è il Class Diagram che ha come obiettivo principale la rappresentazione delle classi che intervengono per la realizzazione del software. Le classi sono descritte tramite attributi e metodi approfondendo così la loro futura implementazione. La classe principale per l'applicazione è la classe HomeWindow che ha lo scopo di creare la schermata principale dell'applicazione su cui verrà, a seguito di un evento (click su un bottone), posizionato un pannello che viene rappresentato dalle varie classi "Panel" estendenti a loro volta la classe JPanel della libreria "java swing". Per semplificare la lettura e comprensione del diagramma si è deciso dei non introdurre i metodi per i pannelli però è importante sottolineare che per ogni classe "Panel" sarà necessario implementare 4 bottoni ("Add", "Delete", "Update" e "Export") e i rispettivi metodi che variano in base al pannello in cui sono situati.
Essa infatti interagisce con la maggior parte delle classi aumentando l'accoppiamento e rendendo il codice più difficilmente mantenibile (se si dovessero modificare le classi che coinvolge, inevitabilmente si avrebbero ripercussioni in cascata al suo interno). Tale classe mantiene un riferimento a tutti gli oggetti principali del progetto, ed è in essa che avvengono le chiamate sequenziali ai metodi delle classi durante la sessione di gioco.
La classe DBManager è necessaria per gestire la connessione con il database e comunicare con lui attraverso i metodi "executeQuery(..)" per eseguire queries che non apportano modifiche alle tabelle del database e "executeUpdate(..)" per eseguire queries apportano modifiche alle tabelle del database.
La classe Data è necessaria per la creazione del database locale, contiene tutti i metodi per la creazione delle tabelle e inoltre fa inserimenti utile nella fase di test.
La classe DataManager è la classe più importante dopo la classe "HomeWindow" infatti contiene tutti i metodi che vengono utilizzati per tutte le operazioni del sistema con il database. Sfrutta la classe Queries in cui sono state inserite le queries sotto forma di stringhe che rappresentano il testo, secondo la sintassi SQL, per effettuare le operazioni su database.

Tutte le classi "*Panel" rappresentano le informazioni sotto forma di tabella (Jtable) fatta eccezione per la classe InfoPanel che fornisce all'utente la possibilità modificare (se lo desidera) o semplicemente vedere le proprie informazioni.

4.1.2 Activity Diagram

Introduciamo ora la rappresentazione delle attività principali che vengono svolte durante l'acquisto di biglietti per eventi sportivi attraverso l'Activity Diagram.
Per il diagramma si è deciso di rappresentale la sola fase di acquisto e salvataggio dei biglietti in locale ipotizzando che la tutte le fasi preliminari necessarie siano state svolte correttamente.
L' Activity Diagram fa uso delle partizioni (User, HomeWindow, BuyTicketDialog) per rendere più chiara la modellazione dei soggetti delle azioni compiute.

4.1.3 Sequence Diagram

Il Sequence Diagram fornisce una rappresentazione in linea temporale delle chiamate ai metodi durante la sessione.
In seguito sono rappresentati due diagrammi raffiguranti:

1. ***Operazione di inserimento nuovo giocatore***

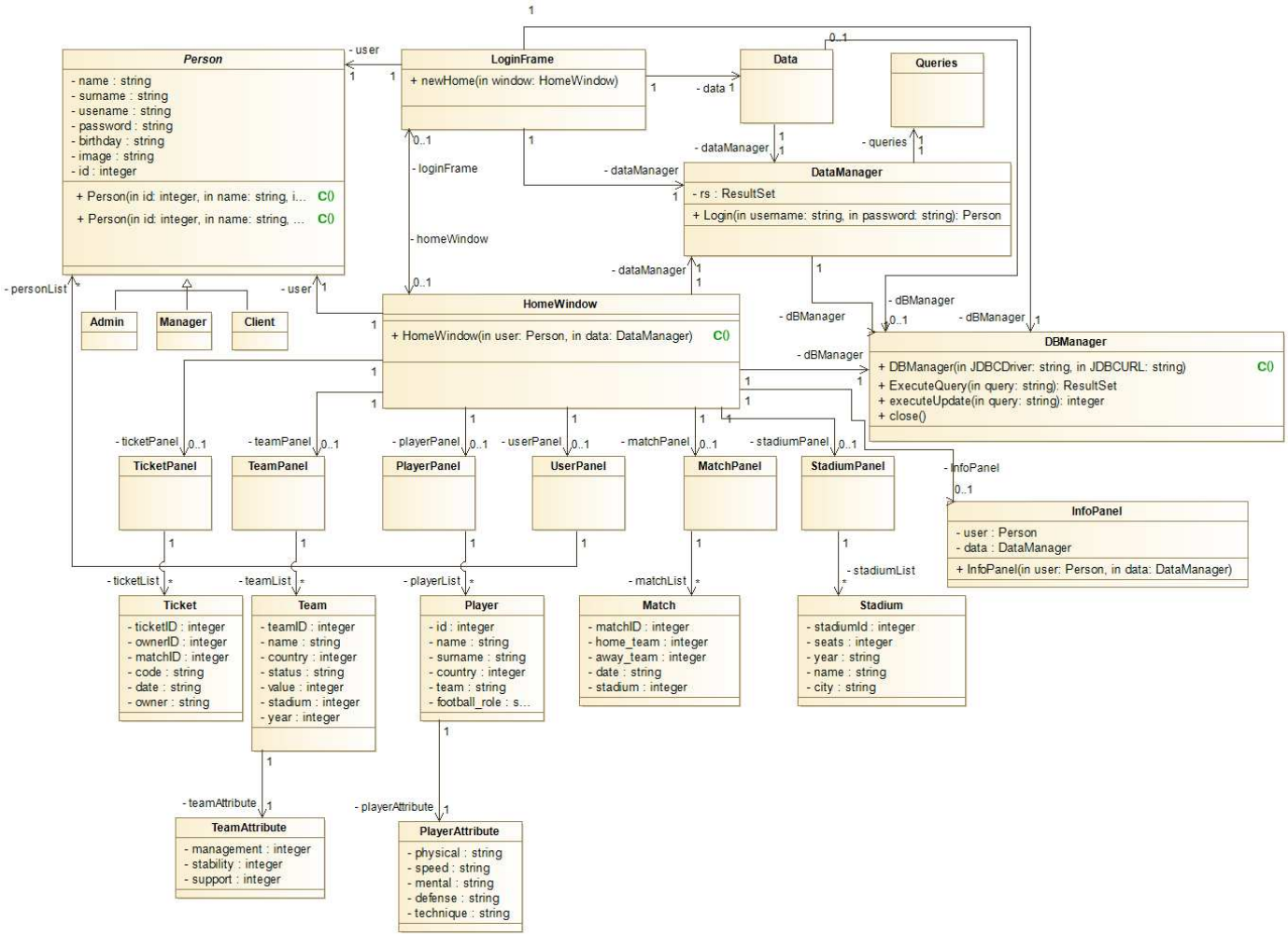
In questo diagramma si rappresentano tutti i metodi utilizzati partendo dalla fase di login fino all'inserimento nel database attraverso la classe "Data" del nuovo giocatore.
Si consideri che le classi "User", "LoginDialog", "Data", "DBManager" e "Queries" siano già state create.

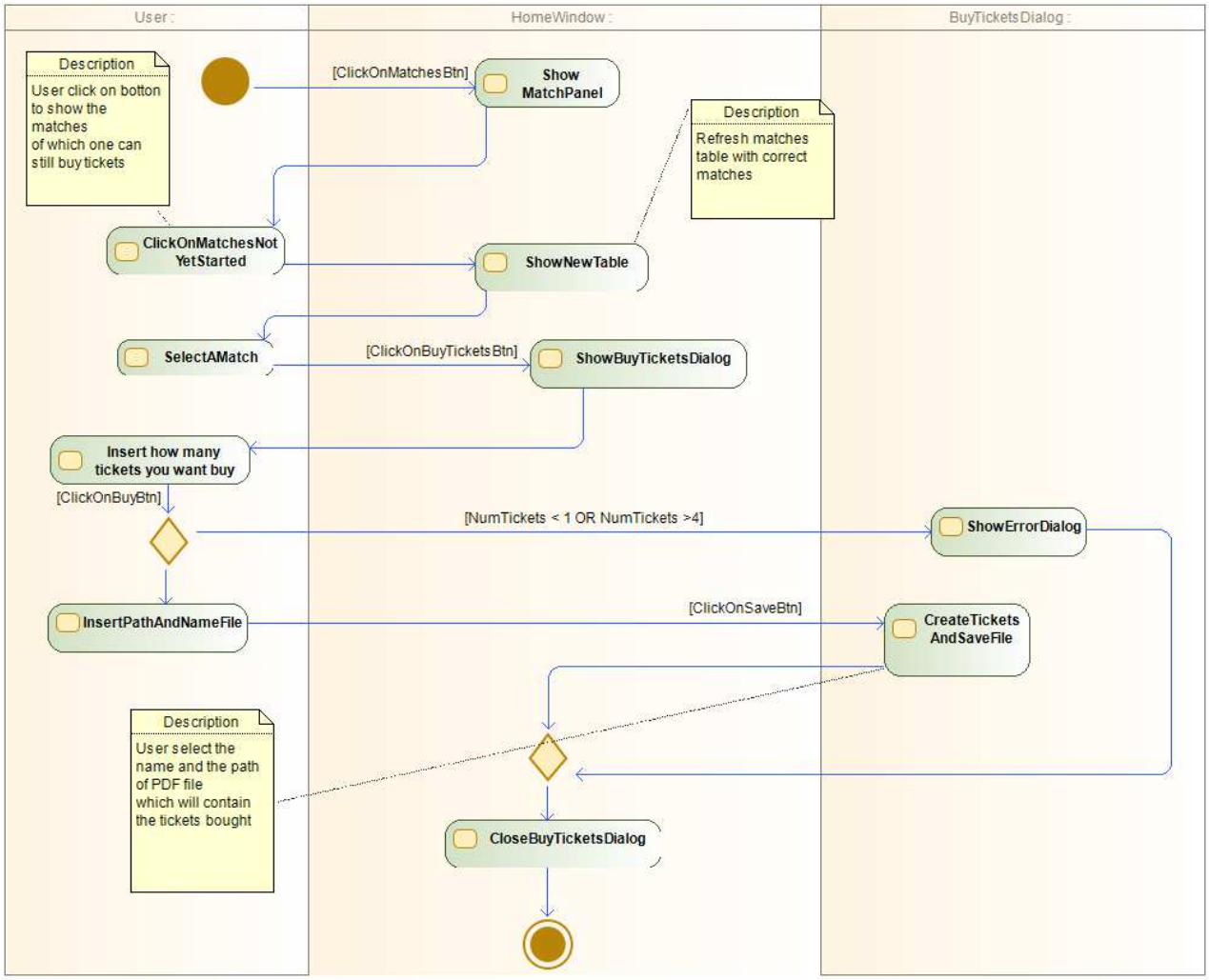
2. ***Operazione di acquisto biglietti per evento sportivo***

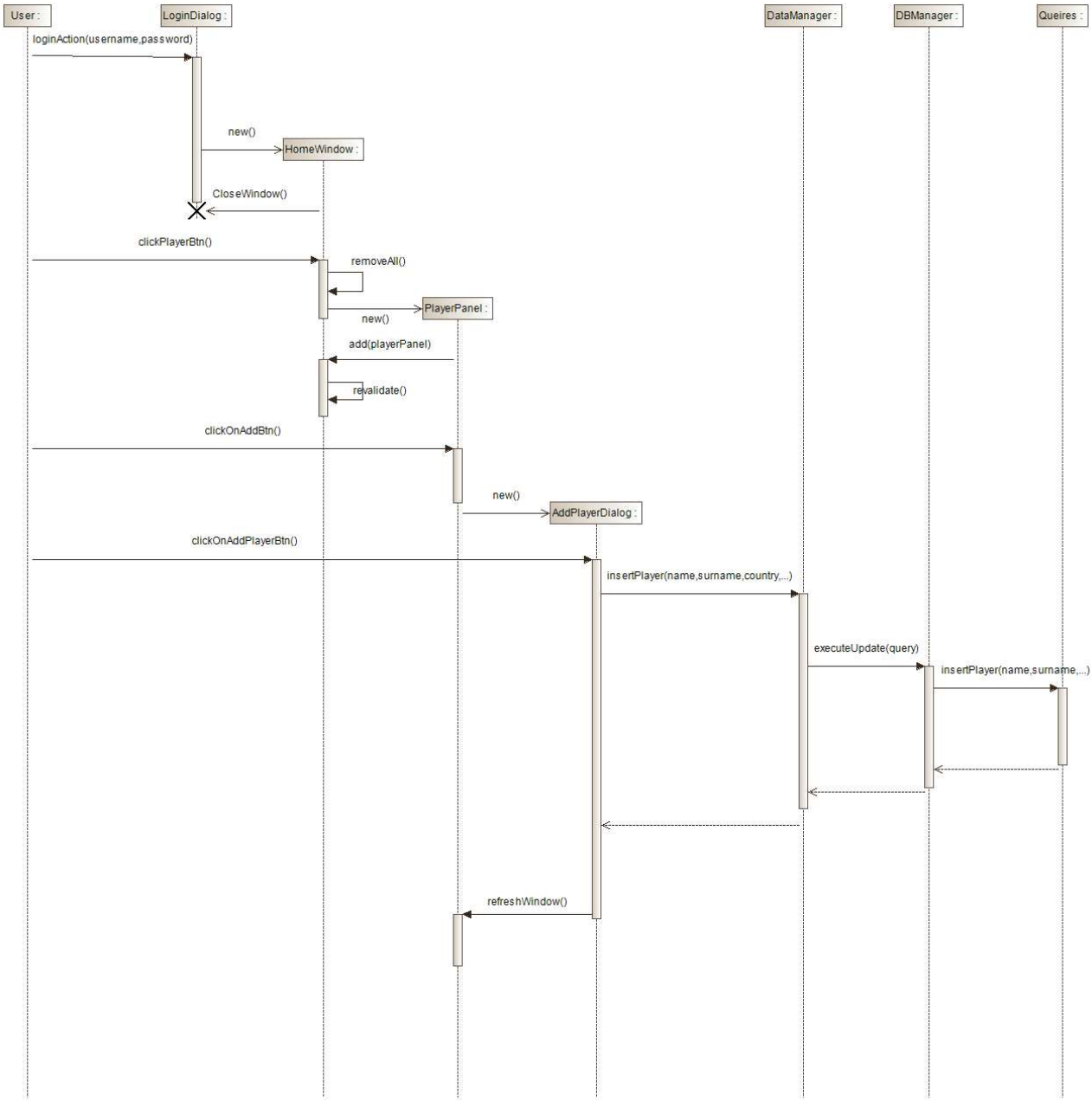
In questo diagramma si rappresentano tutti i metodi utilizzati partendo dalla fase di login fino alla creazione dei biglietti in formato PDF acquistati dall'utente.
Si consideri che le classi "User", "LoginDialog", "Data", "DBManager" e "Queries" siano già state create.

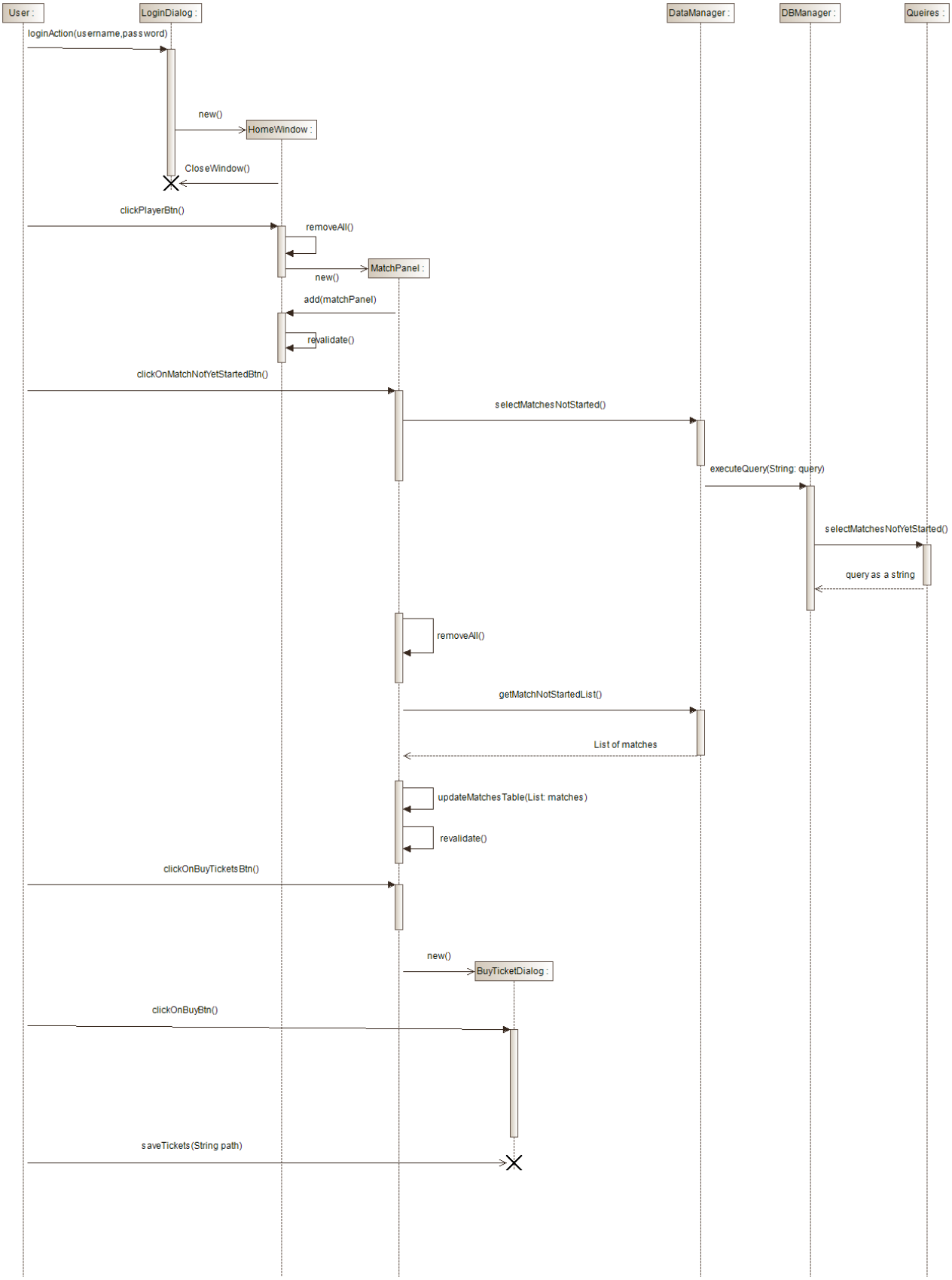
UML DIAGRAMS

SRS UML Design pattern




User :
HomeWindow :
BuyTicketsDialog :
Description
User click on botton to show the matches of which one can still buy tickets
[ClickOnMatchesBtn]
Show MatchPanel
Description
Refresh matches table with correct matches
ClickOnMatchesNot YetStarted
ShowNewTable
SelectAMatch
[ClickOnBuyTicketsBtn]
ShowBuyTicketsDialog
Insert how many tickets you want buy
[ClickOnBuyBtn]
[NumTickets < 1 OR NumTickets >4]
ShowErrorDialog
InsertPathAndNameFile
[ClickOnSaveBtn]
CreateTickets AndSaveFile
Description
User select the name and the path of PDF file which will contain the tickets bought
CloseBuyTicketsDialog


User :
LoginDialog :
DataManager :
DBManager :
Queires :
loginAction(username,password)
new()
HomeWindow :
CloseWindow()
clickPlayerBtn()
removeAll()
new()
PlayerPanel :
add(playerPanel)
revalidate()
clickOnAddBtn()
new()
AddPlayerDialog :
clickOnAddPlayerBtn()
insertPlayer(name,surname,country,...)
executeUpdate(query)
insertPlayer(name,surname,...)
refreshWindow()


User :
LoginDialog :
DataManager :
DBManager :
Queires :
loginAction(username,password)
new()
HomeWindow :
CloseWindow()
clickPlayerBtn()
removeAll()
new()
MatchPanel :
add(matchPanel)
revalidate()
clickOnMatchNotYetStartedBtn()
selectMatchesNotStarted()
executeQuery(String: query)
selectMatchesNotYetStarted()
query as a string
removeAll()
getMatchNotStartedList()
List of matches
updateMatchesTable(List: matches)
revalidate()
clickOnBuyTicketsBtn()
new()
BuyTicketDialog :
clickOnBuyBtn()
saveTickets(String path)

4.2 Design Pattern

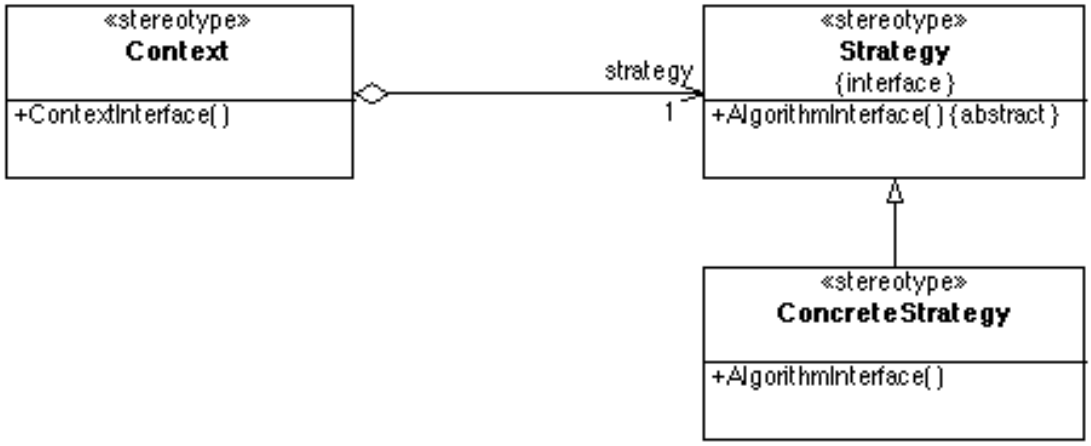
4.2.1 Strategy

Il primo Design Pattern che si è scelto di utilizzare per modellare il parte del progetto software è lo Strategy.

Definizione

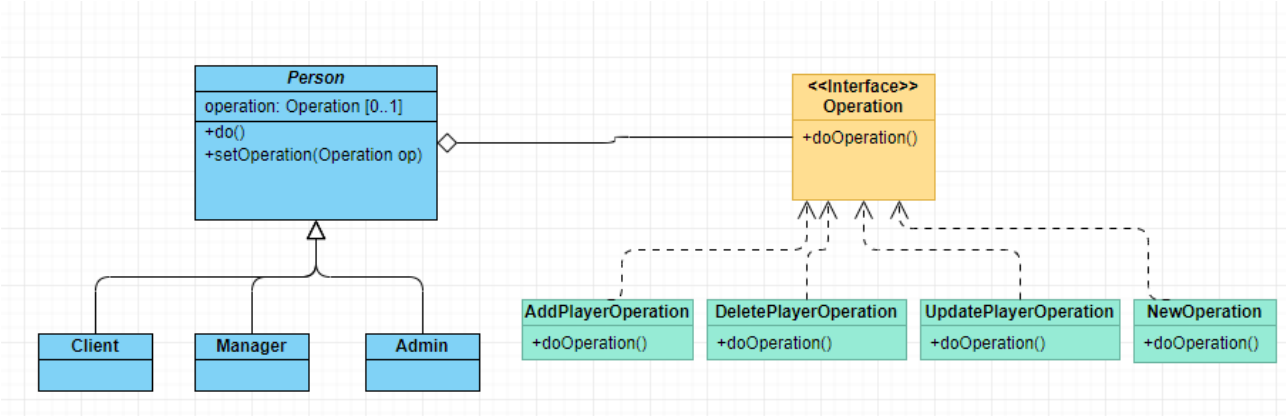
Pattern inizialmente definito dalla Gang of Four, consente di isolare un algoritmo al di fuori di una classe, per far sì che quest'ultima possa variare dinamicamente il suo comportamento, rendendo così gli algoritmi intercambiabili a runtime. Prendendo in esame una famiglia di algoritmi (es. algoritmi di ordinamento, di crittografia, di validazione, etc) grazie allo Strategy Pattern è possibile utilizzare una qualsiasi implementazione (che viene genericamente chiamata Strategy o Strategia), scegliendo fra quelle disponibili, che si rende più opportuna in un determinato contesto, in quanto tutte le implementazioni facenti parte della stessa famiglia hanno interfaccia comune.

Di seguito vediamo una rappresentazione generica del Design Pattern Strategy:



All'interno del nostro sistema abbiamo 3 diverse tipologie di utenti (client, manager, admin), per ognuna di queste il sistema associa automaticamente delle restrizioni sulle operazione che sono possibili effettuare durante una sessione utente. Attraverso l'utilizzo del Pattern Strategy abbiamo quindi l' opportunità di associare ad un utilizzatore una operazione che non è presente nella sua tipologia di utente.

Di seguito vediamo nello specifico del nostro sistema software una possibile rappresentazione:



Nella rappresentazione precedente si è deciso di introdurre come esempio solamente 4 operazioni per rendere più chiaro e leggibile il diagramma ma si può ampliare inserendo tutte le operazioni possibili già implementate nel sistema.
Attraverso la classe "NewOperation" si è voluto evidenziare che questo Design Pattern fornisce la possibilità di inserire nuove funzionalità al sistema inserendo nuove classi che implementeranno l'interfaccia "Operation" senza dover porre alcuna modifica al codice già implementato.
Lo stesso vale se si ha l'intenzione di inserire nuove tipologie di utenti, ad esempio utenti "host" che non necessitano di registrazione, sarà possibile attraverso lo strategy eventualmente abilitare questi utenti a svolgere una operazione all'interno del sistema.
Purtroppo, oltre al pregio di poter cambiare un modo dinamico l'operazione che un utente è in grado di svolgere, lo Strategy ha il limite di poter abilitare al massimo una operazione.

4.2.2 Command

Il secondo Design Pattern che si è scelto di utilizzare è il Command.
Questa scelta è stata presa visto la necessità del sistema di lavorare in continuazione con un database, facendo quindi tante operazioni di diversa tipologia. Il pattern Command quindi ci permette di racchiudere tutte le operazioni della stessa tipologia all'interno di un unica grande classe.

Definizione

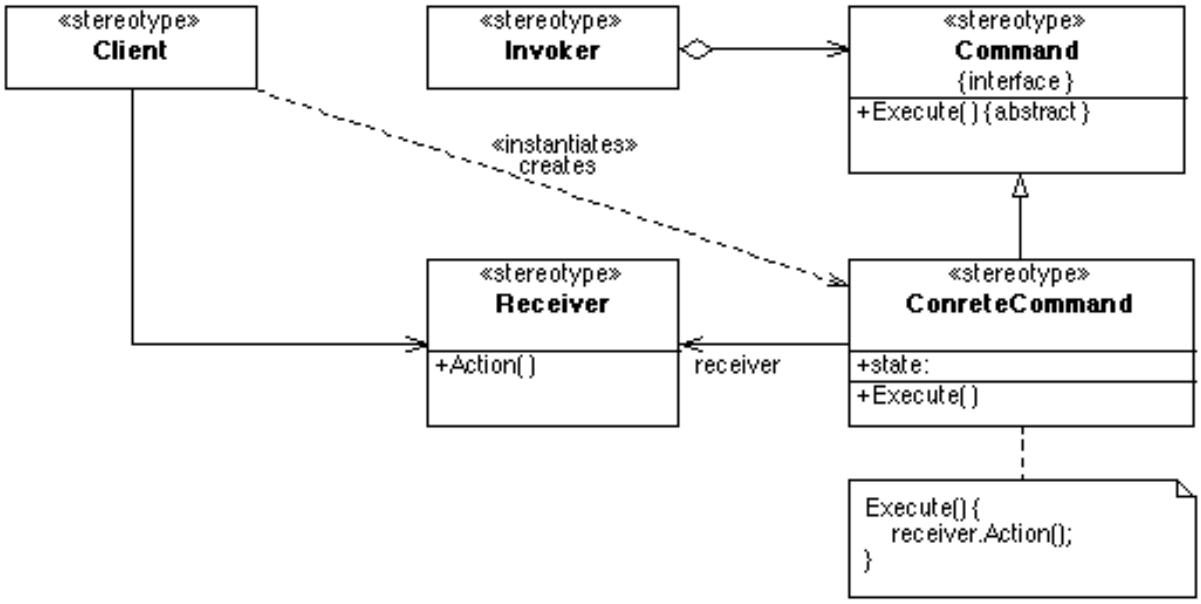
Si tratta di un Pattern comportamentale basato su oggetti e viene utilizzato quando si ha la necessità di disaccoppiare l'invocazione di un comando dai suoi dettagli implementativi, separando colui che invoca il comando da colui che esegue l'operazione. Tale operazione viene realizzata attraverso questa catena: Client->Invocatore->Ricevitore

1. *Client non è tenuto a conoscere i dettagli del comando ma il suo compito è solo quello di chiamare il metodo dell' Invocatore che si occuperà di intermediare l'operazione.*
2. *L'Invocatore ha l'obiettivo di incapsulare, nascondere i dettagli della chiamata come nome del metodo e parametri.*
3. *Il Ricevitore utilizza i parametri ricevuti per eseguire l'operazione*

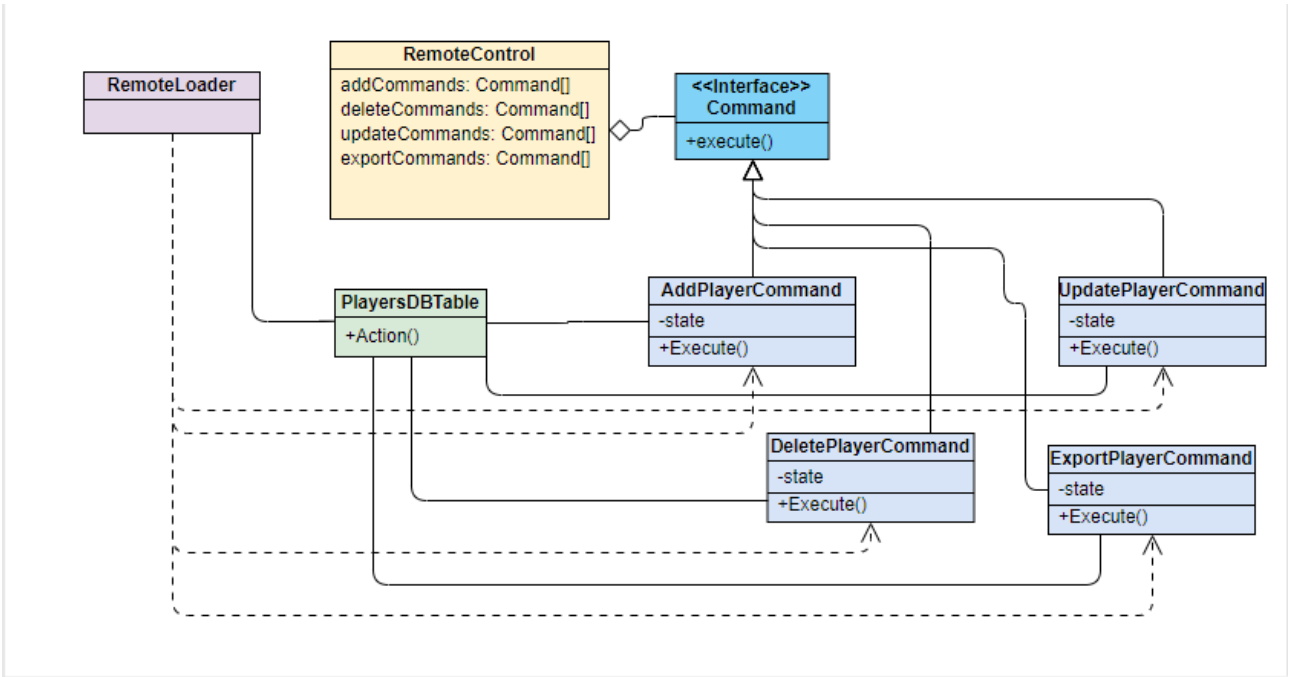
Ma tra l'Invocatore ed il Ricevitore viene posto il Command ossia il comando da eseguire. Il

Command è una semplice interfaccia che viene implementata da una o più classi concrete che invocano il Receiver.

Di seguito vediamo una rappresentazione generica del Design Pattern Command:

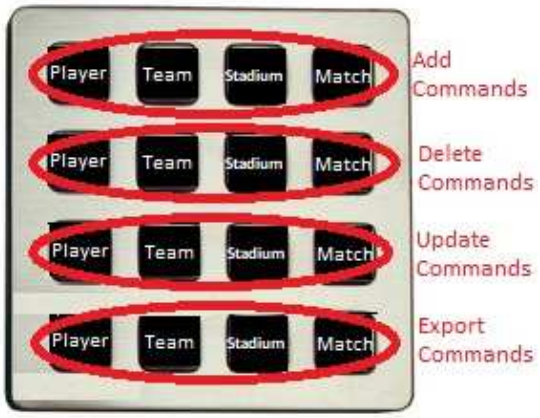


Di seguito vediamo la rappresentazione del Design Pattern Command applicata al nostro sistema:



Per semplificare la visualizzazione si è pensato di rappresentare il class diagram solamente per i comandi relativi alla schermata player.

La classe RemoteLoader sarà la classe che andrà a creare all'interno del sistema il RemoteControl che come possiamo vedere nella figura a fianco (utilizzata per rendere più semplice ed intuitivo quello che andremo ad implementare all'interno del software) rappresenta una pulsantiera contenete tutte le possibili operazioni che un utente durante una sessione può svolgere.
Ogni operazione è rappresentata da una classe che implementa l'interfaccia del Command e quindi implementerà i metodi dichiarati nell'interfaccia.
Ogni tipo di comando sarà associato ad una classe (nel nostro caso PlayerDBTable) che si occuperà di svolgere l'operazione sulla propria tabella del database.



4.2.3 Composite

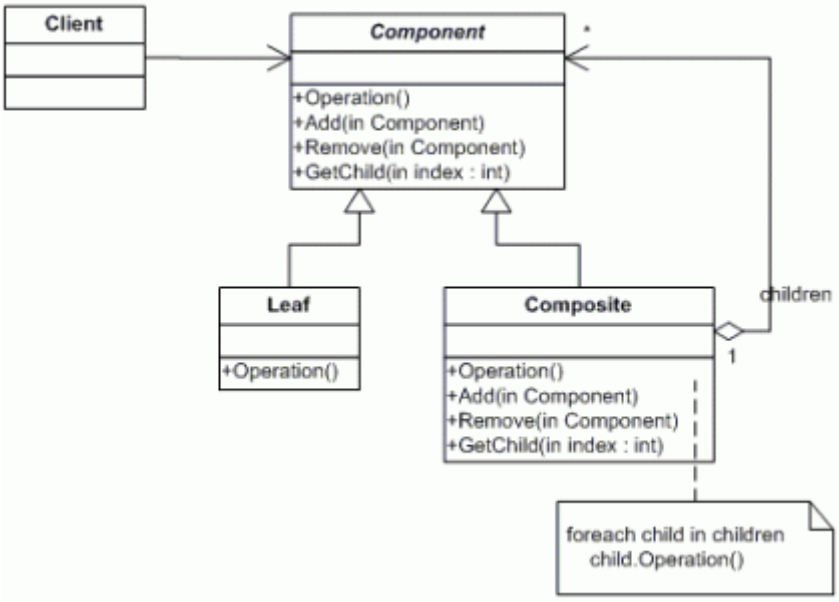
Il terzo Design Pattern che si è deciso di utilizzare è il Composite.

Definizione

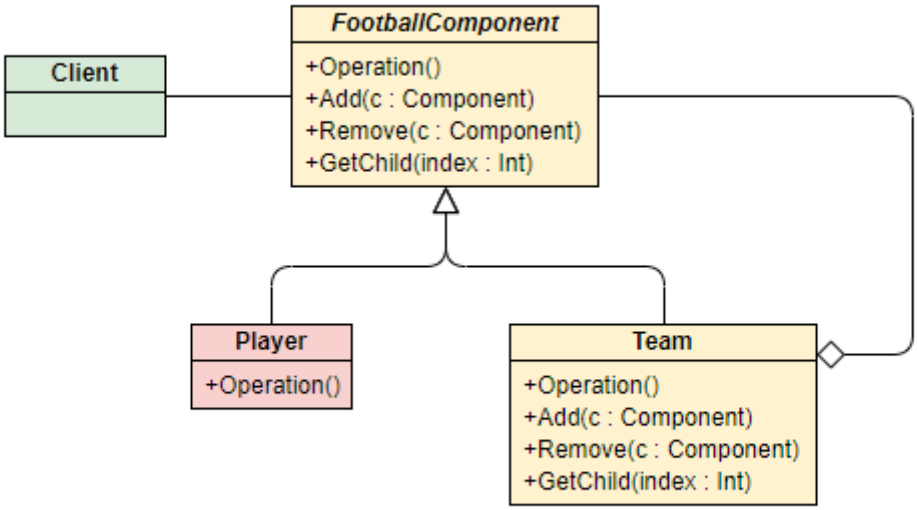
Si tratta di un pattern strutturale basato su oggetti che viene utilizzato quando si ha la necessità di realizzare una gerarchia di oggetti in cui l'oggetto contenitore può detenere oggetti elementari e/o oggetti contenitori. L'obiettivo è di permettere al Client che deve navigare la gerarchia, di comportarsi sempre nello stesso modo sia verso gli oggetti elementari e sia verso gli oggetti contenitori.
Questo pattern è composto dai seguenti partecipanti:

1. *Client: colui che effettua l'invocazione all'operazione di interesse*
2. *Component: definisce l'interfaccia degli oggetti della composizione.*
3. *Leaf: rappresenta l'oggetto foglia della composizione. Non ha figli. Definisce il comportamento "primitivo" dell'oggetto della composizione*
4. *Composite: definisce il comportamento degli oggetti usati come contenitori ed detiene il riferimento ai componenti "figli".*

Di seguito vediamo una rappresentazione generica del Design Pattern Composite:



Di seguito vediamo nello specifico del nostro sistema software una possibile rappresentazione:



Per poter utilizzare questo tipo di Design Pattern è necessario considerare che ogni team può essere composto da giocatori che rappresentano le foglie ma può essere composto da altri team che rappresentano le altre squadre (es. squadre giovanili) presenti all'interno di un'unica società; così facendo abbiamo una estensione delle potenzialità del software.