Università degli Studi di Salerno
Corso di Laurea in Informatica, a.a. 2022-23
Progetto del corso di Ingegneria del Software
prof. A. De Lucia
GitHub: /is-iPlay-22-23/

iPlay

> Object Design Document

Versione 0.9.0

Coordinatore del progetto

Nome	Matricola
Andrea De Lucia **Manuel De Stefano**	**0512100000**

Partecipanti

Nome	Matricola
Vincenzopaolo Esposito	**0512105337**
Francesco Festa	**0512106189**
Samantha Iudici	**0512110197**

Revision History

Data	Versione	Descrizione	Autore
2/12/2022	0.1.0	Creazione documento, creazione capitoli 1, 2 e 3 (Introduzione, Packages, Class Interface Glossary) e scrittura paragrafi 1.1 (Interface Documentation e guidelines).	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
5/12/2022	0.2.0	Creazione paragrafo 1.2 (Definizioni).	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
6/12/2022	0.3.0	Creazione capitoli 4 e 5 (Design Patterns e Priorità di sviluppo),	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
7/12/2022	0.4.0	Aggiunto nel trade-off sicurezza ed efficienza il duplice controllo dei dati; aggiunti livelli di priorità.	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
09/12/2022	0.4.1	Corretto versioning; Corretto trade-off sicurezza ed efficienza.;	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
16/12/2022	0.5.0	Creati i paragrafi “Back-end Packages” e “Front-end Packages”; Aggiunto nei “Back-end Packages” il package “bean”; Aggiunto priorità di sviluppo “utente” e “notifiche.	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
19/12/2022	0.5.1	Corretto trade-off sicurezza ed efficienza e capitolo 5 “priorità di sviluppo”;	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
20/12/2022	0.6.0	Inserito nel class inteface bean la classe “Campo Bean”.	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
21/12/2022	0.7.0	Rimossa la “medium priority” dalla scala delle priorità; Modificata la tabella delle priorità di sviluppo sostituendo le funzionalità con gli identificativi dei requisiti; Corretto trade-off sicurezza ed efficienza e capitolo 5 “priorità di sviluppo”; Modificato tabelle class interface.	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
22/12/2022	0.8.0	Aggiunto paragrafo “1.3 Riferimenti”; Aggiunte definizioni “RAD”, “SDD” e “ODD”; Completati capitoli 2.1 (Back-end Packages) e 2.2 (Front-end Packages).	Vincenzopaolo Esposito Francesco Festa Samantha Iudici
23/12/2022	0.9.0	Completato il capitolo 3 “Class Interface Glossary”.	Vincenzopaolo Esposito Francesco Festa Samantha Iudici

Indice

1. INTRODUZIONE

Fin ora abbiamo definito in maniera dettagliata gli obiettivi del nostro sistema. In questa fase, approfondiremo gli aspetti implementativi, descrivendo con accuratezza tutte le funzionalità chiarite in precedenza, per arrivare alla realizzazione della piattaforma "**iPlay**".
In questo documento descriviamo i trade-off, le linee guida per le interfacce dei sottosistemi, la decomposizione dei sottosistemi in package e classi, le classi di interfacce e design pattern.

Iniziamo considerando i seguenti trade-off:

-Interfaccia vs Usabilità
Non conoscendo l'esatto target di età a cui la piattaforma è rivolta, è stato pensato di favorire l'usabilità ai requisiti di interfaccia, in modo da permettere all'utente di muoversi in maniera intuitiva all'interno del sito. L'utente ha la possibilità di prendere visione di tutte le funzionalità messe a disposizione in base al tipo di ruolo, direttamente dalla sua area riservata. Tutte le ricerche e le visualizzazioni di più elementi sono organizzate in tabelle, inoltre, sono presenti le singole operazioni che si possono effettuare con i singoli elementi della tabella.

-Prestazioni, Costi e Robustezza
Per non sovraccaricare il sistema e rientrare nei costi di sviluppo, il sistema comunica con un secondo server che utilizza dei componenti off-the-shelf, come ad esempio MailTrap, per gestire

l'invio di e-mail in fase di attivazione del profilo e recupero password. Questo permette di concentrarci di più su altri fattori, come le prestazioni della piattaforma, al fine di offrire un prodotto migliore.

-Sicurezza ed Efficienza
Il sistema garantisce la completa protezione dei dati, implementando meccanismi di crittografia dei dati sensibili (credenziali di accesso), principalmente nella fase di autenticazione utilizzando SHA-256 in modo tale da rendere più complicato il recupero delle informazioni in caso di compromissione del database. Il sistema offre controlli e supervisione sui dati inseriti dagli utenti nei form (login, registrazione, ricerche e ecc.), sia nel front-end che nel back-end, in modo tale da guadagnarsi la fiducia del cliente. Qualsiasi tipo di input proveniente dall'utente viene filtrato in modo tale da evitare attacchi di tipo XSS o SQL Injection. Il sistema effettua back up giornalieri ed elimina i dati superflui per conservare una copia dei dati in caso di manomissioni al sistema. Nonostante tutte queste misure preventive impattino sulle prestazioni del sistema, sono necessarie per rispettare le norme minime del GDPR.

1.1. Interface documentation e guidelines

Per evitare fraintendimenti e omissioni che potrebbero causare fault e delay e/o creare ambiguità, ci avvaliamo dell'utilizzo di convenzioni che aiutano anche a rendere più chiaro e comprensibile il nostro lavoro.
Le annotazioni descritte permettono di avere un'accurata comunicazione, grazie ad una semantica ben definita e adattabile per rappresentare i vari aspetti del sistema.
La documentazione delle linee guida e le convenzioni saranno un punto di riferimento per tutti i membri del team che andranno a sviluppare le varie parti del sistema, e sono considerate il fattore più importante per migliorare la comunicazione tra gli sviluppatori.
Di seguito elenchiamo una lista di regole da seguire:

Class Conventions
I nomi delle classi sono costituiti da un singolo nome, più precisamente un sostantivo, al singolare, che descrive in modo specifico cosa rappresenta quella determinata classe. Le classi devono iniziare con la lettera maiuscola e contengono solo lettere.

Methods Conventions
I nomi dei metodi sono costituiti da un verbo (es getName) che descrive nella maniera più appropriata le azioni che svolge quel metodo. Essi devono seguire lo stile Camel case in modo tale da essere più leggibili.

Parameters Conventions
I nomi dei parametri sono costituiti da un sostantivo che caratterizza quel determinato parametro. Il nome deve iniziare con una lettera minuscola e deve contenere solo lettere. Nel caso in cui si necessita dell'aggiunta di un ulteriore nome, si utilizza lo stile Camel case.
Le stesse convenzioni vengono applicate alle variabili.

Error Conventions
Lo stato degli errori deve essere ritornato come un'eccezione, non come un valore di ritorno.

JSP, HTML, Coding Conventions
Le pagine JSP possono contenere codice Java, Javascript, HTML e devono seguire le buone norme

di indentazione. Ogni azione racchiusa in un gruppo di tag, deve essere come un unico blocco, allo stesso livello di indentazione. Per quanto riguarda le operazioni più interne, bisogna usare i Tab per mostrare la gerarchia del codice tramite l'indentazione.

1.2. Definizioni

Design pattern: sono soluzioni di template che gli sviluppatori definiscono nel tempo per risolvere una gamma di problemi ricorrenti. Un design pattern ha quattro elementi:

-Un **nome** che identifica univocamente un pattern da altri pattern;
-Una **descrizione** di un problema che descrive la situazione dove viene usato il pattern;
-Una **soluzione** presentata come insieme di classi e interfacce;
- Un insieme di **conseguenze** che descrive i trade-off e le alternative che devono essere considerate rispetto ai design goal fissati;

RAD: Requirement Analysis Document
SDD: System Design Document
ODD: Object Design Document

1.3. Riferimenti

Questo documento utilizza riferimenti a concetti introdotti ed esplicitati nel **RAD** e nel **SDD**.

2.PACKAGES

L'implementazione del back-end è suddivisa nei seguenti pacchetti, contenuti nella directory /java/.

- Bean
- Collection
- Control
- Database
- Handler
- Model
- Utils

L'implementazione del front-end è suddivisa nei seguenti pacchetti, organizzati in maniera gerarchica.

- View (containers)
  Coincide con la cartella /webapp/
  - img
    Coincide con la cartella /webapp/img/
  - js
    Coincide con la cartella /webapp/js/
  - pages
    Coincide con la cartella /webapp/pages/
    - forms
      Coincide con la cartella /webapp/pages/forms/
    - navBars
      Coincide con la cartella /webapp/pages/navBars/

2.1 Back-end Packages

bean	
Classe	**Descrizione**
CampoBean	Classe che delinea le caratteristiche di un campo sportivo.
EventoBean	Classe che delinea le caratteristiche di un evento sportivo.
FasciaOrariaBean	Classe che delinea le caratteristiche di una fascia oraria.
HaLuogoBean	Classe che memorizza le informazioni di un evento sportivo.
ProprietarioStrutturaSportivaBean	Classe che delinea le caratteristiche di un proprietario di una struttura sportiva.
UtenteSportivoBean	Classe che delinea le caratteristiche di un utente sportivo.

collection	
Classe	**Descrizione**
DizionarioRichiestaPagina	Collezione che permette di organizzare i dati per effettuare il reindirizzamento ad una specifica pagina.
SetCampoFasce	Collezione che permette di organizzare i dati riguardanti un campo sportivo e le relative fasce orarie disponibili e occupate.
SetInfoEvento	Collezione che permette di organizzare i dati relativi ad un evento sportivo.
SetFasciaOrariaOccupata	Classe che estende FasciaOrariaBean che permette di organizzare i dati relativi ad una fascia oraria occupata.

model	
Classe	**Descrizione**
CampoModelDM	Descrive l'interazione col database per CampoBean.
EventoModelDM	Descrive l'interazione col database per EventoBean.
FasciaOrariaModelDM	Descrive l'interazione col database per FasciaOrariaBean.

HaLuogoModelDM	Descrive l'interazione col database per HaLuogoBean.
ProprietarioStrutturaSportivaModelDM	Descrive l'interazione col database per ProprietarioStrutturaSportivaBean.
UtenteSportivoModelDM	Descrive l'interazione col database per UtenteSportivoBean.

control	
Classe	**Descrizione**
CampiSportiviServlet	Servlet che gestisce i campi sportivi.
EventiServlet	Servlet che gestisce gli eventi.
FasceServlet	Servlet che gestisce le fasce orarie.
LoginServlet	Servlet che gestisce l'accesso alla piattaforma.
RicercaServlet	Servlet che gestisce le ricerche.
SignInServlet	Servlet che gestisce la registrazione alla piattaforma.
UserServlet	Servlet che gestisce gli utenti.

database	
Classe	**Descrizione**
DriverManagerConnectionPool	File di configurazione dell'accesso al database tramite i driver JDBC.

handler	
Classe	**Descrizione**
FiltroEventiByLuogoData	Filtro che si occupa di ricercare gli eventi in base al luogo ed una data.
FiltroEventiByLuogoDataTipo	Filtro che si occupa di ricercare gli eventi in base al luogo, una data e un tipo di sport.
FiltroFasceOraByStruttSportData	Filtro che si occupa di ricercare le fasce orarie di una struttura in base allo sport ed una data.
FiltroStrutturaByLuogoDataTipo	Filtro che si occupa di ricercare le strutture sportive in base ad un luogo, una data, e ad un tipo di sport.
RicercaHandlerInterface	Interfaccia da implementare nei filtri.

utils	
Classe	**Descrizione**
DataChecker	Utility che consente il controllo dei dati.
PasswordHasher	Utility che consente di crittografare e decodificare le password.

2.2 Front-end Packages

/webapp/	
JSP	**Descrizione**
index	View container che include altre views.
style	File contenenti le regole di stile della piattaforma.

/webapp/js/	
JS	**Descrizione**
checkSignForm	Script JS che controlla i dati dei form in fase di inserimento.
utils	Script JS che inoltra richieste alle servlet.

/webapp/pages/	
JSP	**Descrizione**
creaCampo	View che consente al Proprietario Struttura Sportiva di inserire i dati inerenti alla creazione di un campo sportivo.
creaEvento	View che consente all' Utente Sportivo di inserire i dati inerenti alla creazione di un evento sportivo.
creaFascia	View che consente al Proprietario Struttura Sportiva di inserire i dati inerenti alla creazione di una fascia oraria per un proprio campo sportivo.
creazioneEvento	View che consente all' Utente Sportivo di inserire i dati inerenti alla ricerca di un Proprietario Struttura Sportiva.
login	View che consente ad un utente non autenticato di inserire i dati per effettuare

	l'accesso alla piattaforma.
modificaCampo	View che consente al Proprietario Struttura Sportiva di inserire i dati inerenti alla modifica di un proprio campo sportivo.
modificaEvento	View che consente all'Utente Sportivo di inserire i dati inerenti alla modifica di un proprio evento sportivo.
modificaFasciaOrariaCampo	View che consente al Proprietario Struttura Sportiva di inserire i dati inerenti alla modifica di una propria fascia oraria per un proprio campo sportivo.
modificaProfilo	View container che include le views dei forms inerenti alla modifica del profilo utente in uso.
modificaStatoEvento	View che consente al Proprietario Struttura Sportiva di inserire i dati inerenti alla modifica di evento sportivo che si vuole ospitare.
mostraEventiProprietarioStrutturaSportiva	View che consente al Proprietario Struttura Sportiva di visualizzare gli eventi sportivi che si vuole ospitare nella propria struttura.
mostraEventiUtenteSportivo	View che consente all'Utente Sportivo di visualizzare gli eventi sportivi creati.
paginaProprietarioStrutturaSportiva	View che mostra le informazioni relative al Proprietario Struttura Sportiva e i relativi campi con le relative fasce orarie (siponibili e occupate).
profiloManager	View container che include le views del profilo utente in uso.
profiloProprietarioStrutturaSportiva	View che mostra le informazioni del profilo relative al Proprietario Struttura Sportiva.
profiloUtenteSportivo	View che mostra le informazioni del profilo relative all'Utente Sportivo.
risultatiRicercaEventi	View che mostra i risultati della ricerca eventi.
risultatiRicercaProprietariStruttureSportive	View che mostra i risultati della ricerca Proprietari Strutture Sportive.
signin	View container che include i form relativi alla registrazione.
visualizzaCampi	View che mostra i campi sportivi di un Proprietario Struttura Sportiva.

visualizzaFasceOrarieCampo	View che mostra le fasce orarie di un campo sportivo di un Proprietario Struttura Sportiva.
welcome	View della landing page di iPlay che include anche i form di ricerca.

/webapp/pages/forms/	
JSP	**Descrizione**
modificaProprietarioStrutturaSportiva	View che consente al Proprietario Struttura Sportiva di inserire i dati relativi alla modifica delle informazioni inerenti al proprio profilo.
modificaUtenteSportivo	View che consente all'Utente Sportivo di inserire i dati relativi alla modifica delle informazioni inerenti al proprio profilo.
registrazioneProprietarioStrutturaSportiva	View che consente al Proprietario Struttura Sportiva di inserire i dati relativi alla registrazione sulla piattaforma.
registrazioneUtenteSportivo	View che consente all'Utente Sportivo di inserire i dati relativi alla registrazione sulla piattaforma.

/webapp/pages/navBars/	
JSP	**Descrizione**
proprietarioStrutturaSportivaNavbar	View che consente la navigazione della piattaforma al Proprietario Struttura Sportiva.
utenteNavbar	View che consente la navigazione della piattaforma all'utente non autenticato.
utenteSportivoNavbar	View che consente la navigazione della piattaforma all'Utente Sportivo.

3. CLASS INTERFACE GLOSSARY

3.1. bean

CampoBean
- idCampo : String - numeroPosti : int - tipo : String - nomeCampo : String - prezzoBigliettoSpettatori : float - prezzoBigliettoAffittuari: float - agibile : boolean - partitalva **:** String
+ CampoBean(String, int, String, String, float, float, boolean, String) + isAgibile() : boolean + setAgibile(boolean) : void + getIdCampo() : String + setIdCampo(String) : void + getNumeroPosti() : int + setNumeroPosti(int) : void + getTipo() : String + setTipo(String) : void + getNomeCampo() : String + setNomeCampo(String) : void + getPrezzoBigliettoSpettatori() : float + setPrezzoBigliettoSpettatori(float) : void + getPrezzoBigliettoAffittuari() : float + setPrezzoBigliettoAffittuari(float) : void + getPartitalva() : String + setPartitalva(String) : void + toString() : String + equals(Object) : boolean

Nome classe	*CampoBean*		
Descrizione	Classe che descrive un campo sportivo.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ CampoBean(String, int, String, String, float, float, boolean, String)			
+ isAgibile() : boolean			
+ setAgibile(boolean) : void			
+ getIdCampo() : String			
+ setIdCampo(String) : void			
+ getNumeroPosti() : int			
+ setNumeroPosti(int) : void			
+ getTipo() : String			
+ setTipo(String) : void			
+ getNomeCampo() : String			
+ setNomeCampo(String) : void			
+ getPrezzoBigliettoSpettatori() : float			
+ setPrezzoBigliettoSpettatori(float) : void			

+ getPrezzoBigliettoAffittuari() : float			
+ setPrezzoBigliettoAffittuari(float) : void			
+ getPartitaIva() : String			
+ setPartitaIva(String) : void			
+ toString() : String			
+ equals(Object) : boolean			

EventoBean
- idEvento : String - pubblico : boolean - nomeEvento : String - stato : String - postiRiservati : int - prezzoEvento : float - partitaIva : String - codiceFiscale : String - idCampo : String
+ EventoBean(String, boolean, String, String, int, float, String, String, String) + getIdEvento() : String + setIdEvento(String) : void + isPubblico() : boolean + setPubblico(boolean) : void + getNomeEvento() : String + setNomeEvento(String) : void + getStato() : String + setStato(String) : void + getPostiRiservati() : int + setPostiRiservati(int) : void + getPrezzoEvento() : float + setPrezzoEvento(float) : void + getPartitaIva() : String + setPartitaIva(String) : void + getCodiceFiscale() : String + setCodiceFiscale(String) : void + getIdCampo() : String + setIdCampo(String) : void + toString() : String + equals(Object) : boolean

Nome classe	*EventoBean*		
Descrizione	Classe che descrive un evento sportivo.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ EventoBean(String, int, String, String, float, float, boolean, String)			
+ getIdEvento() : String			
+ setIdEvento(String) : void			
+ isPubblico() : boolean			

+ setPubblico(boolean) : void			
+ getNomeEvento() : String			
+ setNomeEvento(String) : void			
+ getStato() : String			
+ setStato(String) : void			
+ getPostiRiservati() : int			
+ setPostiRiservati(int) : void			
+ getPrezzoEvento() : float			
+ setPrezzoEvento(float) : void			
+ getPartitaIva(): String			
+ setPartitaIva(String) : void			
+ getCodiceFiscale()			
+ setCodiceFiscale(String) : void			
+ getIdCampo() : void			
+ setIdCampo(String) : void			
+ toString() : String			
+ equals(Object) : boolean			

Nome classe	*FasciaOrariaBean*		
Descrizione	Classe che descrive una fascia oraria.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ FasciaOrariaBean(String, String, String, String)			
+ getIdFasciaOraria() : String			
+ setIdFasciaOraria(String) : void			
+ getOrarioInizio() : String			
+ setOrarioInizio(String) : void			
+ getOrarioFine() : String			
+ setOrarioFine(String) : void			
+ getIdCampo() : String			
+ setIdCampo(String) : void			
+ toString() : String			
+ equals(Object) : boolean			

Nome classe	*HaLuogoBean*		
Descrizione	Classe che descrive la data e il luogo di un evento sportivo.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ HaLuogoBean(String, String, String, boolean)			
+ getIdEvento() : String			
+ setIdEvento(String) : void			
+ getIdFasciaOraria() : String			
+ setIdFasciaOraria (String) : void			
+ getDataEvento() : String			
+ setDataEvento (String) : void			

+ isDisponibile() : boolean			
+ setDisponibilita(boolean) : void			
+ toString() : String			
+ equals(Object) : boolean			

Nome classe	*ProprietarioStrutturaSportivaBean*		
Descrizione	Classe che descrive un Proprietario Struttura Sportiva.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ ProprietarioStrutturaSportivaBean(String, String, String, String, String, String, String, String, String, String, boolean)			
+ ProprietarioStrutturaSportivaBean(String, String, String, String, String, String, String, String, String, String, String[], boolean)			
+ getPartitalva() : String			
+ setPartitalva(String) : void			
+ getEmail() : String			
+ setEmail (String) : void			
+ getPassword() : String			
+ setPassword (String) : void			
+ getViaNumeroCivico() : String			
+ setViaNumeroCivico (String) : void			
+ getProvincia() : String			
+ setProvincia (String) : void			
+ getCitta() : String			
+ setCitta (String) : void			
+ getCap() : String			
+ setCap (String) : void			
+ getRecapitoTelefonico() : String			
+ setRecapitoTelefonico (String) : void			
+ getNomeStruttura() : String			
+ setNomeStruttura (String) : void			
+ getDescrizione() : String			
+ setDescrizione (String) : void			
+ getFestivi() : String[ ]			
+ getFestiviForUpdate() : String			
+ setFestivi(String) : void			
+ isAttivato(): boolean			
+ setAttivato(boolean) : void			
+ getListaGiorniFestivi() : String [ ]			
+ toString() : String			
+ equals(Object) : boolean			

Nome classe	*UtenteSportivoBean*		
Descrizione	Classe che descrive un Utente Sportivo.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ UtenteSportivoBean(String, String, String, String, String, String, String, String, String, String, boolean)			
+ getCodiceFiscale() : String			
+ setCodiceFiscale(String) : void			
+ getEmail() : String			
+ setEmail (String) : void			
+ getPassword() : String			
+ setPassword (String) : void			
+ getViaNumeroCivico() : String			
+ setViaNumeroCivico (String) : void			
+ getProvincia() : String			
+ setProvincia (String) : void			
+ getCitta() : String			
+ setCitta (String) : void			
+ getCap() : String			
+ setCap (String) : void			
+ getRecapitoTelefonico() : String			
+ setRecapitoTelefonico (String) : void			
+ getNome() : String			
+ setNome (String) : void			
+ getCognome() : String			
+ setCognome(String) : void			
+ isAttivato() : boolean			
+ setAttivato(boolean) : void			
+ toString() : String			
+ equals(Object) : boolean			

3.2. collection

Nome classe	*DizionarioRichiestaPagina*		
Descrizione	Classe che contiene i dati relativi ad una richiesta ed alla sua destinazione.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ DizionarioRichiestaPagina(HttpServletRequest, String)			
+ getRequest() : HttpServletRequest			
+ setRequest(HttpServletRequest) : void			
+ getDestination() : String			
+ setDestination(String) : void			

Nome classe	*SetCampoFasce*		
Descrizione	Classe che contiene i dati relativi alle fasce orarie, disponibili e occupate, di un campo sportivo.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ SetCampoFasce(CampoBean campo)			
+ getCampo() : CampoBean			
+ setCampo(CampoBean) : void			
+ getFasceOrarieDisponibili() : Collection<FasciaOrariaBean>			
+ addFasciaOrariaDisponibile(FasciaOrariaBean) : void			
+ getFasceOrarieOccupate() : Collection<SetFasciaOrariaOccupata>			
+ addFasciaOrariaOccupata(SetFasciaOrariaOccupata) : void			
+ toString() : String			

Nome classe	*SetFasciaOrariaOccupata*		
Descrizione	Classe che contiene i dati relativi alle fasce orarie occupate.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ SetFasciaOrariaOccupata(String, String, String, String, EventoBean, HaLuogoBean)			
+ getEvento() : EventoBean			
+ setEvento (EventoBean) : void			
+ getHaLuogo() : HaLuogoBean			
+ setHaLuogo(HaLuogoBean) : void			

Nome classe	*SetInfoEvento*		
Descrizione	Collezione che contiene i dati relativi ad un evento sportivo.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ SetInfoEvento(EventoBean, HaLuogoBean, FasciaOrariaBean, ProprietarioStrutturaSportivaBean, CampoBean)			
+ getProprietarioStrutturaSportiva() : ProprietarioStrutturaSportivaBean			
+ setProprietarioStrutturaSportivaBean(ProprietarioStrutturaSportivaBean) : void			
+ getCampo () : CampoBean			
+ setCampo(CampoBean) : void			

+ getEvento () : EventoBean			
+ setEvento(EventoBean) : void			
+ getHaLuogo() : HaLuogoBean			
+ setHaLuogo(HaLuogoBean) : void			
+ getFascia() : FasciaOrariaBean			
+ setFascia(FasciaOrariaBean) : void			
+ toString() : String			

3.2. model

Nome classe	*CampoModelDM*		
Descrizione	Classe che permette l'interazione con la tabella "campi" del database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ doSave(CampoBean) : void			
+ doUpdate(CampoBean) : void			
+ doDelete(CampoBean) : boolean			
+ doRetrieveByIdCampo (String) : CampoBean			
+ doRetrieveAll() : Collection<CampoBean>			
+ doRetrieveByPartitaIvaTipo(String, String) : Collection<CampoBean>			
+ doRetrieveCampiByPartitaIvaEsterna(String) : Collection<CampoBean>			
+ doRetrieveUniqueForeignKeyByTipoCampo(String) : Collection<String>			

Nome classe	*EventoModelDM*		
Descrizione	Classe che permette l'interazione con la tabella "eventi" del database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ doSave(EventoBean) : void			
+ doUpdate(EventoBean) : void			
+ doDelete(EventoBean) : boolean			
+ doRetrieveByIdEvento(String) : EventoBean			
+ doRetrieveByLuogoDataTipo(String, String) : Collection<EventoBean>			
+ doRetrieveByLuogoData(String, String, String, String) : Collection<EventoBean>			
+ doRetrieveAll() : Collection<EventoBean>			
+ doRetrieveAllByCodiceFiscale(String) : Collection<EventoBean>			
+ doRetrieveAllByPiva (String) : Collection<EventoBean>			

Nome classe	*FasciaOrariaModelDM*		
Descrizione	Classe che permette l'interazione con la tabella "fasce_orarie" del database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ doSave(FascaOrariaBean) : void			
+ doUpdateSave(FascaOrariaBean) : void			
+ doDelete(FascaOrariaBean) : boolean			
+ doRetriveAll() : Collection<FasciaOrariaBean>			
+ doRetriveFasceInRange() : Collection<FasciaOrariaBean>			
+ doRetriveByIdCampo(String) : ArrayList<FasciaOrariaBean>			
+ doRetrieveByIDCampoDataFasciaOraria(String, String, String) : SetFasciaOrariaOccupata			
+ doRetrieveByIdFascia(String) : FasciaOrariaBean			

Nome classe	*HaLuogoModelDM*		
Descrizione	Classe che permette l'interazione con la tabella "ha_luogo" del database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ doSave(HaLuogoBean) : void			
+ doUpdateSave(HaLuogoBean) : void			
+ doDelete(HaLuogoBean) : boolean			
+ doRetrieveByIdEvento(String) : HaLuogoBean			
+ doRetriveByIdFasciaOrariaData(String, String) : int			
+ doRetriveByIdFasciaOrariaIdEvento(String, String) : HaLuogoBean			

Nome classe	*ProprietarioStrutturaSportivaModelDM*		
Descrizione	Classe che permette l'interazione con la tabella "proprietari_strutture_sportive" del database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ doSave(ProprietarioStrutturaSportivaBean) : void			
+ doUpdate(ProprietarioStrutturaSportivaBean) : void			
+ doDelete(String) : boolean			
+ doRetriveByTipoSportPartitaIva(String, String) : Collection<ProprietarioStrutturaSportivaBean>			
+ doRetrieveProprietarioStrtturaSportivaByPartitaIva(Str			

ing) : ProprietarioStrutturaSportivaBean			
+ doRetrievePartitaIvaByRecapitoTelefonico(String) : String			
+ doRetrieveByCampi (String) : Collection<ProprietarioStrutturaSportivaBean>			
+ doRetrieveByLuogo (String, String) : Collection<ProprietarioStrutturaSportivaBean>			
+ doRetrieveAll() : Collection<ProprietarioStrutturaSportivaBean>			
+ doRetrievePartitaIvaAccountAttivatoByEmail (String) : String			

Nome classe	*UtenteSportivoModelDM*		
Descrizione	Classe che permette l'interazione con la tabella "utenti_sportivi" del database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ doSave(UtenteSportivoBean) : void			
+ doUpdate(UtenteSportivoBean) : void			
+ doDelete(String) : void			
+ doRetrieveCodiceFiscaleAccountAttivatoByEmail(String) : String			
+ doRetrieveCodiceFiscaleByRecapitoTelefonico(String): String			
+ doRetrieveUtenteSportivoByCodiceFiscale(String) : UtenteSportivoBean			

3.3. control

Nome classe	*CampiSportiviServlet*		
Descrizione	Servlet che gestisce i campi sportivi di un Proprietario Struttura Sportiva.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void action: ○ **crea** ○ **modifica** ○ **elimina** ○ **mostra** ○ **mostraSingolo**			

Nome classe	***EventiServlet***		
Descrizione	Servlet che gestisce gli eventi sportivi.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void **action:** ○ **crea** ○ **modifica** ○ **elimina** ○ **cambiaStato** ○ **mostraEventiUs** ○ **mostraEventiPss** ○ **mostraSingolo** ○ **mostraSingoloCreazione**			

Nome classe	***FasceServlet***		
Descrizione	Servlet che gestisce le fasce orarie di un campo sportivo di un Proprietario Struttura Sportiva.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void **action:** ○ **crea** ○ **modifica** ○ **elimina** ○ **mostra** ○ **mostraSingolo**			

Nome classe	***LoginServlet***		
Descrizione	Servlet che gestisce l'accesso alla piattaforma.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void **action:** ○ **login** ○ **logout**			

Nome classe	***RicercaServlet***
Descrizione	Classe che gestisce le ricerche che possono

	essere effettuate sulla piattaforma.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void			

Nome classe	*SignInServlet*		
Descrizione	Servlet che gestisce la registrazione di nuovi utenti alla piattaforma.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void			
+ validateUtenteSportivo(HttpServletRequest, HttpServletResponse) : void			
+ validateProprietarioStrutturaSportiva(HttpServletRequest, HttpServletResponse)			

Nome classe	*UserServlet*		
Descrizione	Servlet che gestisce gli utenti di iPlay.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
# doPost(HttpServletRequest, HttpServletResponse) : void **action:** ○ **disattiva** ○ **modifica**			

3.4. database

Nome classe	*DriverManagerConnectionPool*		
Descrizione	Classe che permette l'interazione col database.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ createDBConnection () : Connection			
+ getConnection () : Connection			
+ releaseConnection (Connection) : void			

3.5. handler

Nome classe	*FiltroEventiByLuogoData*		
Descrizione	Classe filtro che ricerca gli eventi per luogo e data.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ setSuccessivo(RicercaHandlerInterface) : void			
+ doRicerca(String, HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			
# doRicercaEventiByDataProvincia(HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			

Nome classe	*FiltroEventiByLuogoDataTipo*		
Descrizione	Classe filtro che ricerca gli eventi per il luogo, la data e il tipo di sport.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ setSuccessivo(RicercaHandlerInterface) : void			
+ doRicerca(String, HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			
# doRicercaEventiByLuogoDataTipo (HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			

Nome classe	*FiltroFasceOraByStruttSportData*		
Descrizione	Classe filtro che ricerca le fasce orarie per la struttura sportiva e la data.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ setSuccessivo(RicercaHandlerInterface) : void			
+ doRicerca(String, HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			
# doFasceOraByStruttSportData (HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			

Nome classe	*FiltroStrutturaByLuogoDataTipo*		
Descrizione	Classe filtro che ricerca i Proprietari Strutture Sportive per il luofo, la data e il tipo di sport.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ setSuccessivo(RicercaHandlerInterface) : void			
+ doRicerca(String, HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			
# doRicercaProprietriStruttureSportiveByLuogoDataTipo (HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			

Nome classe	*RicercaHandlerInterface*

Descrizione	Interfaccia per i filtri di ricerca.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ setSuccessivo(RicercaHandlerInterface) : void			
+ doRicerca(String, HttpServletRequest, HttpServletResponse) : DizionarioRichiestaPagina			

3.6. utils

Nome classe	*DataChecker*		
Descrizione	Classe di utility che permette il controllo dei dati.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ checkEmptyStrings(Map<String, String>) : boolean			
+ checkData(String) : boolean			

Nome classe	*PasswordHasher*		
Descrizione	Classe di utility che permette la crittografia e la decodifica dei dati sensibili.		
Signature dei metodi	**Pre-condizioni**	**Post-condizioni**	**Invariante**
+ hash(String) : String			
+ prepareSecreteKey(String) : void			
+ encrypt (String) : String			
+ decryptString (String) : String			
+ getAlphaNumericString (int) : String			

4. DESIGN PATTERNS

4.1. Chain of Responsability

Descrizione del problema e soluzione:

Conseguenze:

5. PRIORITÀ DI SVILUPPO

In questa sezione vengono presentate le priorità per l'dentificativo del requisito funzionale, presenti all'interno del sistema.

- I requisiti con priorità "**High Priority**" sono quelli che vanno implementati sin da subito, in quanto senza di essi il sistema sarebbe inutilizzabile per l'utente finale.

- I requisiti con priorità "**Low Priority**" sono quelli che non compromettono l'utilizzo del sistema da parte dell'utente finale.

Identificativo requisito	***Priorità***
[RF1]	High Priority
[RF2]	High Priority
[RF3]	High Priority
[RF4]	High Priority
[RF5]	High Priority
[RF6]	Low Priority
[RF7]	High Priority
[RF8]	Low Priority
[RF9]	High Priority
[RF10]	High Priority
[RF11]	High Priority
[RF12]	High Priority
[RF13]	High Priority
[RF14]	High Priority
[RF15]	High Priority
[RF16]	High Priority
[RF17]	Low Priority