Sommario

[Introduzione 1](#_Toc26693365)

[Trade-off 1](#_Toc26693366)

[Componenti off-the-shelf 1](#_Toc26693367)

[Linee guida per la documentazione dell’interfaccia 1](#_Toc26693368)

[Design pattern 1](#_Toc26693369)

[Definizioni, acronimi e abbreviazioni 1](#_Toc26693370)

[Riferimenti 1](#_Toc26693371)

[Packages 1](#_Toc26693372)

[Interfacce delle classi 1](#_Toc26693373)

# Introduzione

## Trade-off

**Comprensibilità vs costi**

Si preferisce aggiungere costi per la documentazione al fine di rendere il codice comprensibile anche alle persone non coinvolte nel progetto o le persone coinvolte che non hanno lavorato a quella parte in particolare. Commenti diffusi nel codice facilitano la comprensione, di conseguenza migliorare la comprensibilità agevola il mantenimento e anche il processo di modifica.

**Interfaccia vs Easy-use**

Il sistema è molto semplice e di facile utilizzo poiché ha un’interfaccia chiara e intuitiva.

## Componenti off-the-shelf

Per il progetto software che si vuole realizzare facciamo uso di componenti***off-the-shelf****,* che sono componenti software disponibili sul mercato per facilitare la creazione del progetto. In particolare, il framework che andremo ad utilizzare è Bootstrap, che è un framework open source che contiene una raccolta di strumenti [liberi](https://it.wikipedia.org/wiki/Software_libero) per la creazione di [siti](https://it.wikipedia.org/wiki/Sito_web) e [applicazioni](https://it.wikipedia.org/wiki/Applicazione_web) per il [Web](https://it.wikipedia.org/wiki/Web). Essa contiene [modelli](https://it.wikipedia.org/wiki/Template) di progettazione basati su [HTML](https://it.wikipedia.org/wiki/HTML) e [CSS](https://it.wikipedia.org/wiki/CSS), sia per la [tipografia](https://it.wikipedia.org/wiki/Tipografia), che per le varie componenti dell'interfaccia, come [moduli](https://it.wikipedia.org/wiki/Form), bottoni e navigazione, e altri componenti dell'interfaccia, così come alcune estensioni opzionali di [JavaScript](https://it.wikipedia.org/wiki/JavaScript).

## Linee guida per la documentazione dell’interfaccia

* Ogni metodo e ogni file devono essere preceduti da un commento, o più precisamente da una documentazione che riporti l’obiettivo che si vuole e deve raggiungere con il nome/i dell’autore/i.
* La convenzione che deve essere adottata da tutti i team member per quanto riguarda i nomi delle variabili, è la notazione CamelCase.

**Organizzazione dei file**

Ogni file deve essere:

* Sviluppato e diviso in base alla categoria di appartenenza, ovvero deve essere correlato ad un’unica funzionalità che persegue. Ogni pagina di FantaFootball (login, AreaPersonale, visualizzaMatch etc.) deve essere implementata in file separati;
* Diviso in più file, se raggiunge una lunghezza tale da divenire difficile da leggere e comprendere.

**Spostamento di linee**

Quando un’espressione supera la lunghezza della linea, occorre spezzarla secondo i seguenti principi generali:

* Interrompere la linea dopo una virgola;
* Interrompere la linea prima di un operatore;
* Preferire interruzioni di alto livello rispetto ad interruzioni di basso livello (interrompere laddove non si interrompe un discorso logico, discorso valido soprattutto per le formule es. (3+4) \* 2 interrompere prima della moltiplicazione senza spezzare gli operandi in parentesi);
* Allineare la nuova linea con l’inizio dell’espressione nella linea precedente;
* Se le regole precedenti rendono il codice più confuso o il codice è troppo spostato verso il margine destro, utilizzare solo otto spazi di indentazione.

**Indentazione**

L’indentazione deve essere effettuata con un TAB e qualunque sia il linguaggio usato per la produzione di codice, ogni istruzione deve essere opportunamente indentata.

**Inizializzazione**

Inizializzare le variabili locali nel punto in cui sono state dichiarate a meno che il suo valore iniziale non dipenda da un calcolo che occorre eseguire prima.

**Posizione**

Mettere le dichiarazioni all’inizio dei blocchi. Non aspettare di dichiarare le variabili al loro primo uso: può confondere il programmatore inesperto e impedire la portabilità del codice dentro lo scope. L’unica eccezione a questa regola sono gli indici dei cicli for che in Java possono essere dichiarati nell’ istruzione stessa. Evitare dichiarazioni locali che nascondono dichiarazioni a più alto livello. Ad esempio, non dichiarare una variabile con lo stesso nome in un blocco interno.

**Parentesi**

A prescindere dalle istruzioni che seguono un IF, è necessario, laddove ci fosse anche una sola istruzione, riportare il blocco di istruzioni tra parentesi graffe. Ogni tag di apertura deve essere necessariamente seguito dall’apposito tag di chiusura (eccetto i tag self-closing)

## Design pattern

## Definizioni, acronimi e abbreviazioni

**ODD**: Object Design Document

**DBMS**: Database Management System

**HTML**: Linguaggio di mark-up per pagine web.

**CSS**: Linguaggio usato per definire la formattazione di pagine web.

**JavaScript**: Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso.

**Off-The-Shelf:** Servizi esterni di cui viene fatto utilizzo da terzi.

**CamelCase**: Consiste nello scrivere più parole insieme delimitando la fine e l’inizio di una nuova parola con una lettera maiuscola.

## Riferimenti

# Packages



Il diagramma descrive la natura three-layer dell’applicazione mostrandone i tre package principali:

* PresentationLayer: contiene i package: GUIUtente, GUIBacheca, GUILega, GUISquadra e GUIErrore
* ApplicationLayer: contiene i package principali GestoreLega, GestoreSquadra, GestoreUtente, GestoreBacheca e il GestoreErrore
* StorageLayer: contiene un package Archivio che gestisce l’interazione con il database.

# Interfacce delle classi

## Entity

|  |  |
| --- | --- |
| Nome classe | Allenatore |
| Descrizione | Questa classe rappresenta l’oggetto Allenatore |
| Signature dei metodi | + getNome(): String  + setNome(nome: String): void  + getCognome(): String  + setCognome(cognome: String): void  + getEmail (): String  + setEmail (email: String): void  + getUsername (): String  + setUsername (username: String): void  + getPassword (): String  + setPassword (password: String): void  + getSquadre (): Collection<Squadra>  + setSquadre (squadre: Collection<Squadra>): void |
| Pre-condizioni | **Context** Allenatore::setEmail(email)  **Pre**: email non deve avere altre corrispondenze nel database  **Context** Allenatore::setUsername (username)  **Pre**: username non deve avere altre corrispondenze nel database |
| Post-condizioni | **Context** Allenatore::setEmail(email)  **Pre**: email è presente nel database  **Context** Allenatore::setUsername (username)  **Pre**: username è presente nel database |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Invito |
| Descrizione | Questa classe rappresenta l’oggetto Invito |
| Signature dei metodi | + getAllenatore (): Allenatore  + setAllenatore(allenatore: Allenatore): void  + getLega (): Lega  + setLega(lega: Lega): void  + getRisposta (): boolean  + setRisposta(risposta: boolean): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Offerta |
| Descrizione | Questa classe rappresenta l’oggetto offerta |
| Signature dei metodi | + getSquadra(): Squadra  + setSquadre(squadra: Squadra): void  +getDataAsta():Date  +setDataAsta(dataAsta: Date): void  +getLega(): Lega  +setLega(lega: Lega): void  +getGiocatore(): Giocatore  +setGiocatore(giocatore: Giocatore): void  +getSomma(): int  +setSomma(somma: int): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Lega |
| Descrizione | Questa classe rappresenta l’oggetto Lega |
| Signature dei metodi | + getNome (): String  + setNome(nome: String): void  + getLogo (): String  + setLogo(logo: String): void  + getMaxAllenatori (): int  + setMaxAllenatori(maxAllenatori: int): void  + getQuotaMensile (): int  + setQuotaMensile(quotaMensile: int): void  + getBudget (): int  + setBudget(budget: int): void  + getPrimoPosto (): int  + setPrimoPosto(primoPosto: int): void  + getSecondoPosto (): int  + setSecondoPosto(secondoPosto: int): void  + getTerzoPosto (): int  + setTerzoPosto(primoPosto: int): void  + getPresidente (): Allenatore  + setPresidente(presidente: Allenatore): void  + getPartite (): collection<Partita>  + setPartite (partite: collection<Partita>): void  + getSquadre (): collection<Squadra>  + setSquadre(squadre: collection<Squadra>): void |
| Pre-condizioni | **Context** Lega::setNome(nome)  **Pre**: nome non deve avere altre corrispondenze nel database |
| Post-condizioni | **Context** Lega::setNome(nome)  **Pre**: nome è presente nel database |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Squadra |
| Descrizione | Questa classe rappresenta l’oggetto Squadra |
| Signature dei metodi | + getNome (): String  + setNome(nome: String): void  + getLogo (): String  + setLogo(logo: String): void  + getAllenatore (): Allenatore  + setAllenatore(allenatore: Allenatore): void  + getNomeLega (): Lega  + setNomeLega (lega: Lega): void  + getPunti (): int  + setPunti (punti: int): void  + getFormazione (): Formazione  + setFormazione (formazione: Formazione): void  + getGiocatori (): array<Giocatore>  + setGiocatori (giocatori: array<Giocatore>): void  + getBudgetRimanente(): int  + setBudgetRimanente (budgetRimanente: int): void |
| Pre-condizioni | **Context** Squadra::setNome(nome)  **Pre**: nome non deve avere altre corrispondenze nel database |
| Post-condizioni | **Context** Squadra::setNome(nome)  **Pre**: nome è presente nel database |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Asta |
| Descrizione | Questa classe rappresenta l’oggetto Asta |
| Signature dei metodi | + getDataInizio (): Date  + setDataInizio (dataInizio: Date): void  + getOra (): Time  + setOra (ora: Time): void  + getDataFine (): Date  + setDataFine (dataFine: Date): void  + getNomeLega (): Lega  + setNomeLega (nomeLega: Lega): void  + getOfferte (): collection<Offerte>  + setOfferte (offerte: collection<Offerte>): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Giocatore |
| Descrizione | Questa classe rappresenta l’oggetto Giocatore |
| Signature dei metodi | + getIdGiocatore(): int  +setIdGiocatore(id: int): void  +getNome(): String  +setNome(nome: String): void  +getCognome(): String  +setCognome(cognome: String): void  +getRuolo(): String  +setRuolo(ruolo: String): void  +getSquadra(): Squadra  +setSquadra(squadra: Squadra): void  +getPresenze(): int  +setPresenze(presenze: int): void  +getVotoMedio(): float  +setVotoMedio(votoMedio: float): void  +getGoal(): int  +setGoal(goal: int): void  +getAssist(): int  +setAssist(assist: int): void  +getAmmonizioni(): int  +setAmmonizioni(ammonizioni: int): void  +getEspulsioni(): int  +setEspulsioni(espulsioni: int): void  +getRigoriSegnati(): int  +setRigoriSegnati(rigoriSegnati: int): void  +getRigoriSbagliati(): int  +setRigoriSbagliati(rigoriSbagliati: int): void  +getRigoriParati(): int  +setRigoriParati(rigoriParati: int): void |
| Pre-condizioni | **Context** Giocatore::setIdGiocatore(idGiocatore)  **Pre**: idGiocatore non deve avere altre corrispondenze nel database  **Context** Giocatore::setRuolo (ruolo)  **Pre**: ruolo == “POR” or ruolo == “DIF” or ruolo == “CEN” or ruolo == “ATT” |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Formazione |
| Descrizione | Questa classe rappresenta l’oggetto Formazione |
| Signature dei metodi | +getModulo(): String  +setModulo(modulo: String): void  +getGiornata(): int  +setGiornata(giornata: int): void  +getSquadra(): Squadra  +setSquadra(squadra: Squadra): void  +getSchierata(): boolean  +setSchierata(schierata: boolean): void  +getGiocatoriSchierati(): Collection  +setGiocatoriSchierati(giocatoriSchierati: Collection): void  +getPanchina(): Collection  +setPanchina(panchina: Collection): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Scambio |
| Descrizione | Questa classe rappresenta l’oggetto Scambio |
| Signature dei metodi | +getIdGiocatore1(): int  +setIdGiocatore1(idGiocatore1: int): void  +getIdGiocatore2(): int  +setIdGiocatore2(idGiocatore2: int): void  +getNomeSquadra1(): String  +setNomeSquadra1(nomeSquadra1: String): void  +getNomeLega1(): String  +setNomeLega1(nomeSquadra1: String): void  +getNomeSquadra2(): String  +setNomeSquadra2(nomeSquadra2: String): void  +getNomeLega2(): String  +setNomeLega2(nomeSquadra1: String): void  +getPrezzoOfferto(): float  +setPrezzoOfferto(prezzoOfferto: float): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Scout |
| Descrizione | Questa classe rappresenta l’oggetto Scout |
| Signature dei metodi | +getNome(): String  +setNome(nome: String): void  +getCognome(): String  +setCognome(cognome: String): void  +getUsername(): String  +setUsername(username: String): void  +getEmail(): String  +setEmail(email: String): void  +getPassword(): String  +setPassword(password: String): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | Post |
| Descrizione | Questa classe rappresenta l’oggetto Post |
| Signature dei metodi | +getIdPost(): int  +setIdPost(idPost: int): void  +getData(): Date  +setData(data: Date): void  +getTitolo(): String  +setTitolo(titolo: String): void  +getTesto(): String  +setTesto(testo: String): void  +getScout(): Scout  +setScout( scout: Scout): void |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

## Model

|  |  |
| --- | --- |
| Nome classe | AllenatoreDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Allenatore. |
| Signature dei metodi | +addAllenatore(nome: String, cognome: String, email: String, username:String, password: String): boolean  +checkLogin (username: String, password: String): boolean  +deleteAllenatore(username: String): boolean  +updateAllenatore(nome: String, cognome: String, email: String, username: String, password: String): boolean  +getAllAllenatori(): List<Giocatore>  +getAllenatoreByUsername(username: String): Allenatore |
| Pre-condizioni | **Context** AllenatoreDAO::addAllenatore (nome, cognome, email, username, password)  **Pre**: nome!=null and cognome!=null and email!=null and username!=null and password!=null  **Context** AllenatoreDAO::checkLogin(username, password)  **Pre**: username!=null and password!=null  **Context** AllenatoreDAO::deleteAllenatore (username)  **Pre**: username!=null  **Context** AllenatoreDAO::updateAllenatore(nome, cognome, email, username, password)  **Pre**: nome!=null and cognome!=null and email!=null and username!=null and password!=null  **Context** AllenatoreDAO::getAllAllenatori()  **Pre**:  **Context** AllenatoreDAO::getAllenatoreByUsername(username)  **Pre**: username!=null |
| Post-condizioni | **Context** AllenatoreDAO::addAllenatore (nome, cognome, email, username, password)  **Post**: database.allenatore->includes(select(a|allenatore.username=username))  **Context** AllenatoreDAO::checkLogin(username, password)  **Post**: true if database.allenatore->includes (select(a|allenatore.username=username and allenatore.password=password)), false altrimenti  **Context** AllenatoreDAO::deleteAllenatore(username)  **Post**: database.allenatore-> not includes(select(a|allenatore.username=username))  **Context** AllenatoreDAO::updateAllenatore (nome, cognome, email, username, password)  **Post**:  **Context** AllenatoreDAO::getAllAllenatori()  **Post**: database.allenatore  **Context** AllenatoreDAO::getAllenatoreByUsername(username)  **Post**: allenatore->select(a|allenatore.username=username)) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | LegaDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Lega. |
| Signature dei metodi | +addLega(nome: String, logo: String, maxAllenatori: int, quotaMensile: int , budget: int, primoPosto: int, secondoPosto: int, presidente: String): boolean  +getLegaByNome(nome: String): Lega |
| Pre-condizioni | **Context** LegaDAO:: addLega(nome, logo, maxAllenatori, quotaMensile, budget, primoPosto, secondoPosto, presidente)  **Pre**: nome!=null and 4 <= maxAllenatori <=10 and quotaMensile!=null and budget!=null and primoPosto!=null and secondoPosto!=null  **Context** LegaDAO::getLegaByNome(nome)  **Pre**: lega!=null |
| Post-condizioni | **Context** LegaDAO::addLega(nome, logo, maxAllenatori, quotaMensile, budget, primoPosto, secondoPosto, presidente)  **Post**: database.lega->includes(select(l|lega.nome=nome))  **Context** LegaDAO::getLegaByNome(nome)  **Post**: lega->select(l|lega.l=nome)) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | InvitoDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Invito. |
| Signature dei metodi | +addInvito(allenatore: String, nomeLega: String): boolean  +updateInvito(allenatore:String, nomeLega: String, risposta: boolean): void  +getAllInvitoAllenatore(allenatore: String): List<Invito> |
| Pre-condizioni | **Context** InvitoDAO::addInvito(allenatore, nomeLega)  **Pre**: allenatore!=null and nomeLega!=null  **Context** InvitoDAO::updateInvito(allenatore, nomeLega, risposta)  **Pre**: allenatore!=null and nomeLega!=null and risposta!=null  **Context** InvitoDAO::getAllInvitoAllenatore(allenatore)  **Pre**: allenatore!=null |
| Post-condizioni | **Context** InvitoDAO::addInvito(allenatore, nomeLega)  **Post**: database.invito->includes(select(i|invito.allenatore=allenatore and invito.nomeLega=nomeLega))  **Context** InvitoDAO::updateInvito(allenatore, nomeLega, risposta)  **Post**:  **Context** InvitoDAO::getAllInvitoAllenatore(allenatore)  **Post**: inviti->select(i!invito.allenatore=allenatore) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | SquadraDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Squadra. |
| Signature dei metodi | + creaSquadra(nome: String, allenatore: String, logo: String, nomeLega: String): boolean  + addGiocatoreSquadra(nomeSquadra: String, nomeLega: String, giocatore: String): boolean  + deleteGiocatoreSquadra(nomeSquadra: String, nomeLega: String, giocatore: String): void  + getSquadreByAllenatore(allenatore: String): List<Squadra>  + getSquadraByNome(nome: String): Squadra  + getSquadreByLega(lega: String): List<Squadra> |
| Pre-condizioni | **Context** SquadraDAO::creaSquadra(nome, allenatore, logo, nomeLega)  **Pre**: nome!=null and allenatore!=null and nomeLega!=null  **Context** SquadraDAO::addGiocatoreSquadra(nomeSquadra, nomeLega, giocatore)  **Pre**: nomeSquadra!=null and nomeLega!=null and giocatore!=null  **Context** SquadraDAO:: deleteGiocatoreSquadra(nomeSquadra, nomeLega, giocatore)  **Pre**: nomeSquadra!=null and nomeLega!=null and giocatore!=null  **Context** SquadraDAO::getSquadreByAllenatore(allenatore)  **Pre**: allenatore!=null  **Context** SquadraDAO::getSquadraByNome(nome)  **Pre**: nome!=null  **Context** SquadraDAO::getSquadreByLega(lega)  **Pre**: lega!=null |
| Post-condizioni | **Context** SquadraDAO::creaSquadra(nome, allenatore, logo, nomeLega)  **Post**: database.squadra->includes(select(s|squadra.nome=nome and squadra.nomeLega=lega))  **Context** SquadraDAO::addGiocatoreSquadra(nomeSquadra, nomeLega, giocatore)  **Post**: database.squadraGiocatore->includes(select(x|squadraGiocatore.giocatore=giocatore and squadraGiocatore.squadra=nomeSquadra and squadraGiocatore.nomeLega=nomeLega))  **Context** SquadraDAO:: deleteGiocatoreSquadra(nomeSquadra, nomeLega, giocatore)  **Post**: database.squadraGiocatore->not includes(select(x|squadraGiocatore.giocatore=giocatore and squadraGiocatore.squadra=nomeSquadra and squadraGiocatore.nomeLega=nomeLega))  **Context** SquadraDAO::getSquadreByAllenatore(allenatore)  **Post**: squadre->select(s!squadra.allenatore=allenatore)  **Context** SquadraDAO::getSquadraByNome(nome)  **Post**: squadre->select(s!squadra.nome=nome)  **Context** SquadraDAO::getSquadreByLega(lega)  **Post**: squadre->select(s!squadra.lega=lega) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | PartitaDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Partita. |
| Signature dei metodi | +addPartita(squadra1: String, squadra2: String, giornata: int): boolean  +updatePartita (goal1: int, goal2: int): void  +getAllPartiteLega(lega: String): List<Partita>  +getAllPartiteSquadra(squadra: String): List<Partita>  + getAllPartiteByGiornata (giornata: int): List<Partita> |
| Pre-condizioni | **Context** PartitaDAO::addPartita(squadra1, squadra2, giornata)  **Pre**: squadra1!=null and squadra2!=null and giornata!=null  **Context** PartitaDAO::updatePartita (goal1, goal2)  **Pre**: goal1!=null and goal2!=null  **Context** PartitaDAO::getAllPartiteLega(lega)  **Pre**: lega!=null  **Context** PartitaDAO::getAllPartiteSquadra(squadra)  **Pre**: squadra!=null  **Context** PartitaDAO:: getAllPartiteByGiornata (giornata)  **Pre**: giornata!=null |
| Post-condizioni | **Context** PartitaDAO::addPartita(squadra1, squadra2, giornata)  **Post**: database.partita->includes(select(p|partita.squadra1=squadra1 and partita.squadra2=squadra2 and partita.giornata=giornata))  **Context** PartitaDAO::updatePartita (goal1, goal2)  **Post**:  **Context** PartitaDAO::getAllPartiteLega(lega)  **Post**: partite->select(p|partita.squadra1.lega=lega)  **Context** PartitaDAO::getAllPartiteSquadra(squadra)  **Post**: partite-> select(p|partita.squadra1=squadra or partita.squadra2=squadra)  **Context** PartitaDAO::getAllPartiteByGiornata (giornata)  **Post**: select(p|partita.giornata=giornata) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | OffertaDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Offerta. |
| Signature dei metodi | + addOfferta (nomeSquadra: String, dataAsta: Date, nomeLega: String, giocatore: String, somma: int): boolean  + deleteOfferta(nomeSquadra: String, dataAsta: Date, nomeLega: String, giocatore: String): boolean  + updateOfferta(nomeSquadra: String, dataAsta: Date, nomeLega: String, giocatore: String, somma: int): void  + getAllOfferteByAllenatore(allenatore: String): List<Offerta>  + getAllOfferteByAsta(dataInizioAsta: Date, nomeLega: String): List<Offerta>  + getAllOfferteByAstaAllenatore(dataInizioAsta: Date, nomeLega: String, allenatore: String): List<Offerta>  + getOfferteGiocatoreAsta(giocatore, dataInizioAsta: Date, nomeLega: String): List<Offerta>  + getOffertaGiocatoreAllenatore(giocatore, dataInizioAsta: Date, nomeLega: String,allenatore): Offerta |
| Pre-condizioni | **Context** OffertaDAO::addOfferta (nomeSquadra, dataAsta, nomeLega, giocatore, somma)  **Pre**: nomeSquadra!=null and dataAsta!=null and nomeLega!=null and giocatore!=null and somma!=null  **Context** OffertaDAO:: deleteOfferta(nomeSquadra, dataAsta, nomeLega, giocatore)  **Pre**: nomeSquadra!=null and dataAsta!=null and nomeLega!=null and giocatore!=null  **Context** OffertaDAO:: updateOfferta(nomeSquadra, dataAsta, nomeLega, giocatore, somma)  **Pre**: nomeSquadra!=null and dataAsta!=null and nomeLega!=null and giocatore!=null and somma!=null  **Context** OffertaDAO:: getAllOfferteBySquadra(squadra)  **Pre**: squadra!=null  **Context** OffertaDAO:: getAllOfferteByAsta(dataInizioAsta, nomeLega)  **Pre**: asta!=null and nomeLega!=null  **Context** OffertaDAO:: getAllOfferteByAstaSquadra(dataInizioAsta, nomeLega, squadra)  **Pre**: asta!=null and nomeLega!=null and squadra!=null  **Context** OffertaDAO:: getOfferteGiocatoreAsta(giocatore, dataInizioAsta, nomeLega)  **Pre**: giocatore!=null and dataInizioAsta!=null and nomeLega!=null  **Context** OffertaDAO:: getOffertaGiocatoreSquadra(giocatore, dataInizioAsta, nomeLega,squadra)  **Pre**: giocatore!=null and dataInizioAsta!=null and nomeLega!=null and squadra != null |
| Post-condizioni | **Context** OffertaDAO:: addOfferta (nomeSquadra, dataAsta, nomeLega, giocatore, somma)  **Post**: database.offerta->includes(select(o|offerta.nomeSquadra=nomeSquadra and offerta.dataAsta=dataAsta and offerta.nomeLega=nomeLega and offerta.giocatore=giocatore and offerta.somma=somma))  **Context** OffertaDAO:: deleteOfferta(nomeSquadra, dataAsta, nomeLega, giocatore)  **Post**: database.offerta-> not includes(select(o|offerta.nomeSquadra=nomeSquadra and offerta.dataAsta=dataAsta and offerta.nomeLega=nomeLega and offerta.giocatore=giocatore))  **Context** OffertaDAO:: updateOfferta(nomeSquadra, dataAsta, nomeLega, giocatore, somma)  **Post**:  **Context** OffertaDAO:: getAllOfferteBySquadra(squadra)  **Post**: offerte-> select(o|offerta.squadra=squadra)  **Context** OffertaDAO:: getAllOfferteByAstaSquadra(dataInizioAsta, nomeLega, squadra)  **Post**: offerte-> select(o|offerta.dataInizioAsta=dataInizioAsta and offerta.nomeLega=nomeLega and offerta.squadra=squadra)  **Context** OffertaDAO:: getAllOfferteByAsta(dataInizioAsta, nomeLega)  **Post**: offerte-> select(o|offerta.dataInizioAsta=dataInizioAsta and offerta.nomeLega=nomeLega)  **Context** OffertaDAO:: getOfferteGiocatoreAsta(giocatore, dataInizioAsta, nomeLega)  **Post**: offerte-> select(o|offerta.giocatore=giocatore and offerta.dataInizioAsta=dataInizioAsta and offerta.nomeLega=nomeLega)  **Context** OffertaDAO:: getOffertaGiocatoreSquadra(giocatore, dataInizioAsta, nomeLega,squadra)  **Pre**: offerte-> select(o|offerta.giocatore=giocatore and offerta.dataInizioAsta=dataInizioAsta and offerta.nomeLega=nomeLega and offerta.squadra = squadra) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | AstaDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Asta. |
| Signature dei metodi | + addAsta(dataInizio: Date, ora: Time, dataFine: Date, nomeLega: String): boolean  + getAsteByAllenatore(allenatore: String): List<Asta> |
| Pre-condizioni | **Context** AstaDAO::addAsta(dataInizio, ora, dataFine, nomeLega)  **Pre**: dataInizio!=null and ora!=null and dataFine!=null and nomeLega!=null and dataInizio<dataFine  **Context** AstaDAO::getAsteByAllenatore(allenatore)  **Pre**: allenatore!=null |
| Post-condizioni | **Context** AstaDAO::addAsta(dataInizio, ora, dataFine, nomeLega)  **Post**: database.asta->includes(select(a|asta.dataInizio=dataInizio and asta.ora=ora and asta.dataFine=dataFine and asta.nomeLega=nomeLega))  **Context** AstaDAO:: getAsteByAllenatore(allenatore)  **Post**: aste->select(a|offerta.allenatore=allenatore) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | GiocatoreDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Giocatore. |
| Signature dei metodi | + getSquadreGiocatore(giocatore: Giocatore)  + aggiornaGiocatore(giocatore: Giocatore) |
| Pre-condizioni | **Context** GiocatoreDAO::getSquadreGiocatore(giocatore)  **Pre**: giocatore!=null  **Context** GiocatoreDAO::aggiornaGiocatore(giocatore)  **Pre**: |
| Post-condizioni | **Context** GiocatoreDAO::getSquadreGiocatore(giocatore)  **Post**: giocatori->select(g|squadragiocatore.giocatore=giocatore.id)  **Context** GiocatoreDAO::aggiornaGiocatore(giocatore)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | FormazioneDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Formazione. |
| Signature dei metodi | + addFormazione (nomeSquadra: String, nomeLega: String, modulo: String, giornata: int, schierata: boolean): boolean  + addGiocatoreFormazione(nomeSquadra: String, nomeLega: String, giornata: int, giocatore: String): boolean  + deleteGiocatoreFormazione(nomeSquadra: String, nomeLega: String, giornata: int, giocatore: String): boolean  + updateGiocatoreFormazione(nomeSquadra: String, nomeLega: String, giornata: int, giocatore1: String, giocatore2: String): boolean  + updateFormazione (nomeSquadra: String, nomeLega: String, modulo: String, giornata: int, schierata: boolean): boolean  + aggiornaPanchina (nomeSquadra: String, nomeLega: String, giornata: int, giocatore: String, numero: int): boolean  + getFormazioniBySquadra(nomeSquadra: String, nomeLega: String): List<Formazione>  + getFormazioneBySquadraGiornata (nomeSquadra: String, nomeLega: String, giornata: int): Formazione |
| Pre-condizioni | **Context** FormazioneDAO::addFormazione (nomeSquadra, nomeLega, modulo, giornata, schierata)  **Pre**: nomeSquadra!=null and nomeLega!=null and modulo!=null and giornata!=null and schierata!=null  **Context** FormazioneDAO::addGiocatoreFormazione(nomeSquadra, nomeLega, giornata, giocatore)  **Pre**: nomeSquadra!=null and nomeLega!=null and giornata!=null and giocatore!=null  **Context** FormazioneDAO::deleteGiocatoreFormazione(nomeSquadra, nomeLega, giornata, giocatore)  **Pre**: nomeSquadra!=null and nomeLega!=null and giornata!=null and giocatore!=null  **Context** FormazioneDAO::updateGiocatoreFormazione(nomeSquadra, nomeLega, giornata, giocatore1, giocatore2)  **Pre**: nomeSquadra!=null and nomeLega!=null and giornata!=null and giocatore1!=null and giocatore2!=null  **Context** FormazioneDAO::updateFormazione (nomeSquadra, nomeLega, modulo, giornata, schierata)  **Pre**: nomeSquadra!=null and nomeLega!=null and modulo!=null giornata!=null and schierata!=null  **Context** FormazioneDAO::aggiornaPanchina (nomeSquadra, nomeLega, giornata, giocatore, numero)  **Pre**: nomeSquadra!=null and nomeLega!=null and giornata!=null and giocatore!=null and numero!=null  **Context** FormazioneDAO::getFormazioniBySquadra(nomeSquadra, nomeLega)  **Pre**: nomeSquadra!=null and nomeLega!=null  **Context**FormazioneDAO::getFormazioneBySquadraGiornata (nomeSquadra, nomeLega, giornata)  **Pre**: nomeSquadra!=null and nomeLega!=null and giornata!=null |
| Post-condizioni | **Context** FormazioneDAO::addFormazione (nomeSquadra, nomeLega, modulo, giornata, schierata)  **Post**: database.formazione->includes(select(f|formazione.nomeSquadra=nomeSquadra and formazione.nomeLega=nomeLega and formazione.modulo=modulo and formazione.giornata=giornata and formazione.schierata=schierata))  **Context** FormazioneDAO::addGiocatoreFormazione(nomeSquadra, nomeLega, giornata, giocatore)  **Post**: database.giocatoreFormazione->includes(select(f|giocatoreFormazione.nomeSquadra=nomeSquadra and giocatoreFormazione.nomeLega=nomeLega and giocatoreFormazione.giornata=giornata and giocatoreFormazione.giocatore=giocatore))  **Context** FormazioneDAO::deleteGiocatoreFormazione(nomeSquadra, nomeLega, giornata, giocatore)  **Post**: database.giocatoreFormazione-> not includes(select(f|giocatoreFormazione.nomeSquadra=nomeSquadra and giocatoreFormazione.nomeLega=nomeLega and giocatoreFormazione.giornata=giornata and giocatoreFormazione.giocatore=giocatore))  **Context** FormazioneDAO::updateGiocatoreFormazione(nomeSquadra, nomeLega, giornata, giocatore1, giocatore2)  **Post**: database.giocatoreFormazione-> includes(select(f|giocatoreFormazione.nomeSquadra=nomeSquadra and giocatoreFormazione.nomeLega=nomeLega and giocatoreFormazione.giornata=giornata and giocatoreFormazione.giocatore=giocatore2))  **Context** FormazioneDAO::updateFormazione (nomeSquadra, nomeLega, modulo, giornata, schierata)  **Post**:  **Context** FormazioneDAO::aggiornaPanchina (nomeSquadra, nomeLega, giornata, giocatore, numero)  **Post**: database.giocatoreFormazione-> includes(select(f|giocatoreFormazione.nomeSquadra=nomeSquadra and giocatoreFormazione.nomeLega=nomeLega and giocatoreFormazione.giornata=giornata and giocatoreFormazione.giocatore=giocatore and giocatoreFormazione.numero=numero))  **Context** FormazioneDAO::getFormazioniBySquadra(nomeSquadra, nomeLega)  **Post**: formazioni-> select(f|formazione.nomeSquadra=nomeSquadra and formazione.nomeLega=nomeLega)  **Context**FormazioneDAO::getFormazioneBySquadraGiornata (nomeSquadra, nomeLega, giornata)  **Post**: formazione-> select(f|formazione.nomeSquadra=nomeSquadra and formazione.nomeLega=nomeLega and formazione.giornata=giornata) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ScambioDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Scambio. |
| Signature dei metodi | + creaScambio (giocatore1: Giocatore, squadra1: Squadra, giocatore2: Giocatore, squadra2: Squadra, prezzoOfferto: int): boolean  + accettaScambio (giocatore1: Giocatore, squadra1: Squadra, giocatore2: Giocatore, squadra2: Squadra, risposta: boolean): boolean  + rifiutaScambio (giocatore1: Giocatore, squadra1: Squadra, giocatore2: Giocatore, squadra2: Squadra, risposta: boolean): void  + getScambiNonAccettatiSquadra (squadra: Squadra): List<Scambio>  + getScambioById (giocatore1: Giocatore, squadra1: Squadra, giocatore2: Giocatore, squadra2: Squadra): Scambio |
| Pre-condizioni | **Context** ScambioDAO::creaScambio (giocatore1, squadra1, giocatore2, squadra2, prezzoOfferto)  **Pre**: giocatore1!=null and squadra1!=null and giocatore2!=null and squadra2!=null  **Context** ScambioDAO::accettaScambio (giocatore1, squadra1, giocatore2, squadra2, risposta)  **Pre**: giocatore1!=null and squadra1!=null and giocatore2!=null and squadra2!=null and risposta=true  **Context** ScambioDAO::rifiutaScambio (giocatore1, squadra1, giocatore2, squadra2, risposta)  **Pre**: giocatore1!=null and squadra1!=null and giocatore2!=null and squadra2!=null and risposta=false  **Context** ScambioDAO::getScambiNonAccettatiSquadra (squadra)  **Pre**: squadra!=null  **Context** ScambioDAO::getScambioById (giocatore1, squadra1, giocatore2, squadra2)  **Pre**: giocatore1!=null and squadra1!=null and giocatore2!=null and squadra2!=null |
| Post-condizioni | **Context** ScambioDAO::creaScambio (giocatore1, squadra1, giocatore2, squadra2, prezzoOfferto)  **Post**: database.scambio-> includes(select(s|scambio.giocatore1=giocatore1.id and scambio.nomeSquadra1=squadra1.nome and scambio.nomeLega1=squadra1.nomeLega and scambio.giocatore2=giocatore2.id and scambio.nomeSquadra2=squadra2.nome and scambio.nomeLega2=squadra2.nomeLega ))  **Context** ScambioDAO::accettaScambio (giocatore1, squadra1, giocatore2, squadra2, risposta)  **Post**: database.squadraGiocatore->includes(select(g| squadraGiocatore.nomeSquadra=squadra1.nome and squadraGiocatore.nomeLega=squadra1.nomeLega and squadraGiocatore.giocatore=giocatore2) and (select(g| squadraGiocatore.nomeSquadra=squadra2.nome and squadraGiocatore.nomeLega=squadra2.nomeLega and squadraGiocatore.giocatore=giocatore1)  **Context** ScambioDAO::rifiutaScambio (giocatore1, squadra1, giocatore2, squadra2, risposta)  **Post**:  **Context** ScambioDAO::getScambiNonAccettatiSquadra (squadra)  **Post**: scambi->select(s|scambio.squadra2=squadra.nome and scambio.nomeLega2=squadra.nomeLega)  **Context** ScambioDAO::getScambioById (giocatore1, squadra1, giocatore2, squadra2)  **Post**: scambio->select(s|scambio.squadra1=squadra1.nome and scambio.nomeLega1=squadra1.nomeLega and scambio.giocatore1=giocatore1.id and scambio.squadra2=squadra2.nome and scambio.nomeLega2=squadra2.nomeLega and scambio.giocatore2=giocatore2.id) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ScoutDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Scout. |
| Signature dei metodi | + checkLogin(username: String, password: String): boolean  + logout(): void  + updateScout (nome: String, cognome: String, email: String, username: String, password: String): boolean |
| Pre-condizioni | **Context** ScoutDAO::checkLogin(username, password)  **Pre**: username!=null and password!=null  **Context** ScoutDAO:: updateScout (nome, cognome, email, username, password)  **Pre**: nome!=null and cognome!=null and email!=null and username!=null and password!=null |
| Post-condizioni | **Context** ScoutDAO::checkLogin(username, password)  **Post**: true if database.scout ->includes (select(s|scout.username=username and scout.password=password)), false altrimenti  **Context** ScoutDAO:: updateScout (nome, cognome, email, username, password)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | PostDAO |
| Descrizione | Questa classe è un manager che si occupa di interagire con il database. Gestisce le query rigurdanti Post. |
| Signature dei metodi | + addPost(data: Date, titolo: String, testo: String, scout: String): boolean  + removePost(id: int): boolean  + updatePost(id: int, data: Date, titolo: String, testo: String, scout: String): boolean  + getPostByScout(scout: String): List<Post>  + getAllPost(): List<Post> |
| Pre-condizioni | **Context** PostDAO::addPost(data, titolo, testo, scout)  **Pre**: data!=null and titolo!=null and testo!=null and scout!=null  **Context** PostDAO::removePost(id)  **Pre**: id!=null  **Context** PostDAO::updatePost(id, data, titolo, testo, scout)  **Pre**: id!=null and data!=null and titolo!=null and testo!=null and scout!=null  **Context** PostDAO::getPostByScout(scout)  **Pre**: scout!=null  **Context** PostDAO::getAllPost()  **Pre**: |
| Post-condizioni | **Context** PostDAO::addPost(data, titolo, testo, scout)  **Post:**  **Context** PostDAO::removePost(id)  **Post**: database.post->not includes(select(p| p.id=id))  **Context** PostDAO::updatePost(id, data, titolo, testo, scout)  **Post**:  **Context** PostDAO::getPostByScout(scout)  **Post**: posts->select(p|post.scout=scout)  **Context** PostDAO::getAllPost()  **Post**: database.post |
| Invariante |  |

## Control

|  |  |
| --- | --- |
| Nome classe | InserisciPostServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PostDAO i dati di un post da pubblicare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** InserisciPostServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“categoria”)!=null and request.getParameter(“testo”)!=null |
| Post-condizioni | **Context** InserisciPostServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: PostDAO.getPostById(idPost)!=null |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RimuoviPostServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PostDAO i dati di un post da rimuovere |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RimuoviPostServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idPost”) != null |
| Post-condizioni | **Context** RimuoviPostServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: PostDAO.getPostById(idPost)==null |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ModificaPostServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PostDAO i dati di un post da modificare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** ModificaPostServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idPost”) != null and request.getParameter(“testo”) != null |
| Post-condizioni | **Context** ModificaPostServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: PostDAO.getPostById(idPost).testo == testo |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | InserisciMatchServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PartitaDAO i dati di una partita da registrare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** InserisciMatchServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“squadra1”) != null and request.getParameter(“squadra2”) != null and request.getParameter(“giornata”) != null and PartitaDAO.getPartitaByKey(squadra1,squadra2,giornata) == null |
| Post-condizioni | **Context** InserisciMatchServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: PartitaDAO.getPartitaById(squadra1,squadra2,giornata)!=null |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | AggiornaMatchServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PartitaDAO i dati di una partita da aggiornare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** AggiornaMatchServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“squadra1”) != null and request.getParameter(“squadra2”) != null and request.getParameter(“giornata”) != null and  request.getParameter(“punteggioSquadra1”) != null and  request.getParameter(“punteggioSquadra2”) != null  PartitaDAO.getPartitaByKey(squadra1,squadra2,giornata) != null |
| Post-condizioni | **Context** AggiornaMatchServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: PartitaDAO.getPartitaById(squadra1,squadra2,giornata).getPunteggioSquadra1()==punteggioSquadra1 and PartitaDAO.getPartitaById(squadra1,squadra2,giornata).getPunteggioSquadra2()==punteggioSquadra2 |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RicercaMatchServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PartitaDAO i dati di una partita da aggiornare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RicercaMatchServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“squadra1”) != null and request.getParameter(“squadra2”) != null and request.getParameter(“giornata”) != null |
| Post-condizioni | **Context** RicercaMatchServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | CalendarioServlet |
| Descrizione | Questa classe è un control che si occupa di passare a PartitaDAO i dati per effettuare la ricerca del calendario richiesto (relativo all’intera Lega o ad una squadra specifica) |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** CalendarioServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“lega”) != null and  LegaDAO.getLegaById(lega) != null |
| Post-condizioni | **Context** CalendarioServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RicercaGiocatoreServlet |
| Descrizione | Questa classe è un control che si occupa di passare a GiocatoreDAO i dati per effettuare la ricerca di un giocatore applicando vari filtri (giocatore specifico, per squadra, per ruolo, ricerca per nome) |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RicercaGiocatoreServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**:request.getParameter(“lega”) != null and (request.getParameter(“idGiocatore”) != null or request.getParameter(“idSquadra”) != null or request.getParameter(“ruolo”) != null or  request.getParameter(“nomeGiocatore”) != null ) |
| Post-condizioni | **Context** RicercaGiocatoreServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | InserisciGiocatoreSquadraServlet |
| Descrizione | Questa classe è un control che si occupa di passare a GiocatoreDAO i dati per inserire un giocatore in una squadra |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** InserisciGiocatoreSquadraServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idGiocatore”) != null and request.getParameter(“squadra”) != null and SquadraDAO.getSquadraById(squadra) != null and GiocatoreDAO.getGiocatoreById(idGiocatore) != null |
| Post-condizioni | **Context** InserisciGiocatoreSquadraServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: SquadraDAO.getSquadraById(squadra).getGiocatori().contains( giocatore) == true |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | ClassificaServlet |
| Descrizione | Questa classe è un control che si occupa di passare a LegaDAO i dati per ricercare la classifica di una Lega |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** ClassificaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“lega”) != null and LegaDAO.getLegaById(lega) != null |
| Post-condizioni | **Context** ClassificaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | CreaScambioServlet |
| Descrizione | Questa classe è un control che si occupa di passare a ScambioDAO i dati di una proposta di scambio da inviare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** CreaScambioServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idGiocatore1”) != null and request.getParameter(“idGiocatore2”) != null and request.getParameter(“Squadra1”) != null and request.getParameter(“Squadra2”) != null and request.getParameter(“prezzoOfferto”) != null ScambioDAO.getScambioById(idGiocatore1,idGiocatore2,Allenatore1,Allenatore2) == null |
| Post-condizioni | **Context** CreaScambioServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**:ScambioDAO.getScambioById(idGiocatore1,idGiocatore2,Allenatore1,Allenatore2) != null |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RispostaScambioServlet |
| Descrizione | Questa classe è un control che si occupa di passare a ScambioDAO i dati relativi alla risposta ad una proposta di scambio |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RispostaScambioServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idGiocatore1”) != null and request.getParameter(“idGiocatore2”) != null and request.getParameter(“Squadra1”) != null and  request.getParameter(“Squadra2”) != null and  request.getParameter(“risposta”) != null and  ScambioDAO.getScambioById(idGiocatore1,idGiocatore2,Allenatore1,Allenatore2) != null |
| Post-condizioni | **Context** RispostaScambioServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**:if request.getParameter(“risposta”) == true  SquadraDAO.getSquadraById(squadra1).getGiocatori().contains(giocatore2) == true and  SquadraDAO.getSquadraById(squadra2).getGiocatori().contains(giocatore1) == true and  ScambioDAO.getScambioById(idGiocatore1,idGiocatore2,Allenatore1,Allenatore2) .getRisposta() == true  else  ScambioDAO.getScambioById(idGiocatore1,idGiocatore2,Allenatore1,Allenatore2) .getRisposta() == false |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | AggiungiGiocatoreFormazioneServlet |
| Descrizione | Questa classe è un control che si occupa di passare a FormazioneDAO i dati di un giocatore da aggiungere ad una formazione |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** AggiungiGiocatoreFormazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“idGiocatore”)!=null and  request.getParameter(“squadra”)!=null |
| Post-condizioni | **Context** AggiungiGiocatoreFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: SquadraDAO.getSquadraByKey(squadra).getFormazione().contains(giocatore)==true |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | RimuoviGiocatoreFormazioneServlet |
| Descrizione | Questa classe è un control che si occupa di passare a FormazioneDAO i dati di un giocatore da rimuovere dalla formazione. |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RimuoviGiocatoreFormazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“giocatore”)!=null and  request.getParameter(“squadra”)!=null and  SquadraDAO.getSquadraByKey(squadra).getGiocatori().contains(giocatore) == true |
| Post-condizioni | **Context** RimuoviGiocatoreFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: SquadraDAO.getSquadraByKey(squadra).getFormazione().getGiocatori().contains(giocatore)== false |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | SostituisciGiocatoreFormazioneServlet |
| Descrizione | Questa classe è un control che si occupa di passare a FormazioneDAO i dati due giocatori da scambiare nella formazione |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** SostituisciGiocatoreFormazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“giocatore1”)!=null and  request.getParameter(“giocatore2”)!=null and  request.getParameter(“squadra”)!=null and  SquadraDAO.getSquadraByKey(squadra).getGiocatori().contains(giocatore1) == true |
| Post-condizioni | **Context** SostituisciGiocatoreFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: SquadraDAO.getSquadraByKey(squadra).getFormazione().getGiocatori().contains(giocatore1) == false and  SquadraDAO.getSquadraByKey(squadra).getFormazione().getGiocatori().contains(giocatore2)== true |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | VisualizzaSquadraServlet |
| Descrizione | Questa classe è un control che si occupa di passare a SquadraDAO i dati di una squadra da visualizzare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** VisualizzaSquadraServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“squadra”) != null |
| Post-condizioni | **Context** VisualizzaSquadraServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | OffertaServlet |
| Descrizione | Questa classe è un control che si occupa di passare a OffertaDAO i dati di un’offerta da registrare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** OffertaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**:Request.getParameter(“dataInizioAsta”)!= null and  request.getParameter(“squadra”) != null and  request.getParameter(“giocatore”) != null and  request.getParameter(“sommaOfferta”) != null and  OffertaDAO.getOffertaGiocatoreAllenatore(lega,dataInizioAsta,idAllenatore,giocatore).getSomma() < sommaOfferta |
| Post-condizioni | **Context** OffertaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: OffertaDAO.getOffertaGiocatoreSquadra(lega,dataInizioAsta,squadra,giocatore).getSomma() == sommaOfferta |
| Invariante |  |