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 | 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 |
| 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  +getNomeSquadra2(): String  +setNomeSquadra2(nomeSquadra2: 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 |  |

**Servlet**

|  |  |
| --- | --- |
| Nome classe | RegistrazioneServlet |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni |  |
| Post-condizioni |  |

**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**:  **Context** AllenatoreDAO::checkLogin(username, password)  **Post**:  **Context** AllenatoreDAO::deleteAllenatore(username)  **Post**:  **Context** AllenatoreDAO::updateAllenatore (nome, cognome, email, username, password)  **Post**:  **Context** AllenatoreDAO::getAllAllenatori()  **Post**:  **Context** AllenatoreDAO::getAllenatoreByUsername(username)  **Post**: |
| 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**:  **Context** LegaDAO::getLegaByNome(nome)  **Post**: |
| 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**:  **Context** InvitoDAO::updateInvito(allenatore, nomeLega, risposta)  **Pre**:  **Context** InvitoDAO::getAllInvitoAllenatore(allenatore)  **Pre**: |
| Post-condizioni | **Context** InvitoDAO::addInvito(allenatore, nomeLega)  **Post**:  **Context** InvitoDAO::updateInvito(allenatore, nomeLega, risposta)  **Post**:  **Context** InvitoDAO::getAllInvitoAllenatore(allenatore)  **Post**: |
| 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(squadra: String, giocatore: String): boolean  + deleteGiocatoreSquadra(squadra: String, giocatore: String): void  + updateGiocatoreSquadra(squadra: String, giocatore1: String, giocatore2: String): boolean  + 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(squadra, giocatore)  **Pre**: squadra!=null and giocatore!=null  **Context** SquadraDAO:: deleteGiocatoreSquadra(squadra, giocatore)  **Pre**: squadra!=null and giocatore!=null  **Context** SquadraDAO::updateGiocatoreSquadra(squadra, giocatore1, giocatore2)  **Pre**: squadra!=null and giocatore1!=null and giocatore2!=null  **Context** SquadraDAO::getSquadreByAllenatore(allenatore)  **Pre**: allenatore!=null  **Context** SquadraDAO::getSquadraByNome(nome)  **Pre**: nome!=null  **Context** SquadraDAO::getSquadreByLega(lega)  **Pre**: lega!=null |
| Post-condizioni |  |
| 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 |  |
| Pre-condizioni | **Context** GestorePartita::addPartita(squadra1, squadra2, giornata)  **Pre**: squadra1!=null and squadra2!=null and giornata!=null  **Context** GestorePartita::updatePartita (goal1, goal2)  **Pre**: goal1!=null and goal2!=null  **Context** GestorePartita::getAllPartiteLega(lega)  **Pre**: lega!=null  **Context** GestorePartita::getAllPartiteSquadra(squadra)  **Pre**: squadra!=null  **Context** GestorePartita::getAllPartiteByGiornata (giornata)  **Pre**: giornata!=null |
| Post-condizioni | **Context** GestorePartita::creaPartita(squadra1, squadra2, giornata)  **Post**:  **Context** GestorePartita::setRisultato(goal1, goal2)  **Post**:  **Context** GestorePartita::getAllPartiteLega(lega)  **Post**: |
| 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 |  |
| Pre-condizioni | **Context** OffertaDAO::addOfferta ()  **Pre**: squadra1!=null and squadra2!=null and giornata!=null  deleteOfferta  updateOfferta  getAllOfferteByAllenatore  getAllOfferteByAsta  getOfferteGiocatoreAsta() |
| Post-condizioni |  |
| 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 |  |
| Pre-condizioni | **Context** AstaDAO::addAsta()  **Pre**: squadra1!=null and squadra2!=null and giornata!=null  **Context** AstaDAO::getAsteByAllenatore()  **Pre**: |
| Post-condizioni | **Context** GestorePartita::getAllPartiteLega(lega)  **Post**: |
| 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 |  |
| Pre-condizioni | **Context** GiocatoreDAO::getSquadreGiocatore()  **Pre**:  **Context** GiocatoreDAO::aggiornaGiocatore()  **Pre**: |
| Post-condizioni | **Context** GestorePartita::getAllPartiteLega(lega)  **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 |  |
| Pre-condizioni | **Context** FormazioneDAO::addFormazione (modulo, giornata, schierata)  **Pre**:  **Context** FormazioneDAO::addGiocatoreFormazione()  **Pre**:  **Context** FormazioneDAO::deleteGiocatoreFormazione()  **Pre**:  **Context** FormazioneDAO::updateGiocatoreFormazione()  **Pre**:  **Context** FormazioneDAO::updateFormazione (modulo, giornata, schierata)  **Pre**:  **Context** FormazioneDAO::aggiornaPanchina ()  **Pre**: |
| Post-condizioni | **Context** GestorePartita::getAllPartiteLega(lega)  **Post**: |
| 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 |  |
| Pre-condizioni | **Context** ScambioDAO::creaScambio ()  **Pre**:  **Context** ScambioDAO::accettaScambio ()  **Pre**:  **Context** ScambioDAO::rifiutaScambio ()  **Pre**:  **Context** ScambioDAO::getScambiNonAccettatiAllenatore ()  **Pre**: |
| Post-condizioni | **Context** GestorePartita::getAllPartiteLega(lega)  **Post**: |
| 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 |  |
| Pre-condizioni | **Context** ScoutDAO::checkLogin()  **Pre**:  **Context** ScoutDAO::logout()  **Pre**:  **Context** ScoutDAO::updateScout ()  **Pre**: |
| Post-condizioni | **Context** GestorePartita::getAllPartiteLega(lega)  **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 |  |
| Pre-condizioni | **Context** PostDAO::addPost()  **Pre**:  **Context** PostDAO::removePost()  **Pre**:  **Context** PostDAO::updatePost()  **Pre**:  **Context** PostDAO::getPostByScout()  **Pre**:  **Context** PostDAO::getAllPost()  **Pre**: |
| Post-condizioni | **Context** GestorePartita::getAllPartiteLega(lega)  **Post**: |
| Invariante |  |

**ROBA DI GAETANO CHE HA MESSO LE SERVLET. FRA DOBBIAMO COPIARE TUTTO**

|  |  |
| --- | --- |
| Nome classe | RegistrazioneServlet |
| Descrizione | Questa classe è un control che si occupa di verificare se le credenziali inserite dall’utente sono valide per poi passarle al DAO addetto all’inserimento di quest’ultime nel database. |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RegistrazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“nome”)!=null and rispetta il format “^[A-Za-z ]{2,50}$” request.getParamter(“cognome”)!=null and rispetta il formato “^[A-Za-z ]{2,50}$”. Request.getParamter(“username”)!=null and rispetta il formato “^(?=.{4,20}$)(?![\_.])(?!.\*[\_.]{2})[a-zA-Z0-9.\_]$”and non dev’essere presente nel database. Request.getParameter(“password”)!=null and rispetta il formato “^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.{4,10})$”. Request.getParameter(“email”)!=null and rispetta il formato “^[A-Za-z0-9. \_%+-] @ [A-Za-z] [.A-Za-z]. [A-Za-z]\*$” and non dev’essere presente nel database. |
| Post-condizioni | **Context** RegistrazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: true if AllenatoreDAO.getAllenatoreByUsername(username)!=null, else altrimenti |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | LoginServlet |
| Descrizione | Questa classe è un control che si occupa di passare le credenziali di login al DAO addetto alla verifica di queste e se queste sono verificate provvederà a creare |
| Signature dei metodi | +doGet(request: HttpServletRequest, response: HttpServletRequest):void |
| Pre-condizioni | **Context** RegistrazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest)  **Pre**: request.getParameter(“nome”)!=null and rispetta il format “^[A-Za-z ]{2,50}$” request.getParamter(“cognome”)!=null and rispetta il formato “^[A-Za-z ]{2,50}$”. Request.getParamter(“username”)!=null and rispetta il formato “^(?=.{4,20}$)(?![\_.])(?!.\*[\_.]{2})[a-zA-Z0-9.\_]$”and non dev’essere presente nel database. Request.getParameter(“password”)!=null and rispetta il formato “^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?=.{4,10})$”. Request.getParameter(“email”)!=null and rispetta il formato “^[A-Za-z0-9. \_%+-] @ [A-Za-z] [.A-Za-z]. [A-Za-z]\*$” and non dev’essere presente nel database. |
| Post-condizioni | **Context** RegistrazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest)  **Post**: true if AllenatoreDAO.getAllenatoreByUsername(username)!=null, else altrimenti |
| Invariante |  |