UNIVERSITÀ DEGLI STUDI DI SALERNO

UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Corso di Ingegneria del Software

Progetto FantaFootball

ODD

Partecipanti	Matricola
Maria Natale	05121 05084
Gaetano Casillo	05121 05510
Pasquale Caramante	05121 05334
Mattia Della Sala	05121 05306

0 SOMMARIO

1 INTRODUZIONE

1.1 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, intuitivo e di facile utilizzo poiché ha un'interfaccia chiara e intuitiva.

1.2 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 per la creazione di siti e applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, bottoni e navigazione, e altri componenti dell'interfaccia, così come alcune estensioni opzionali di JavaScript.

1.3 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)

1.4 DESIGN PATTERN

1.5 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.

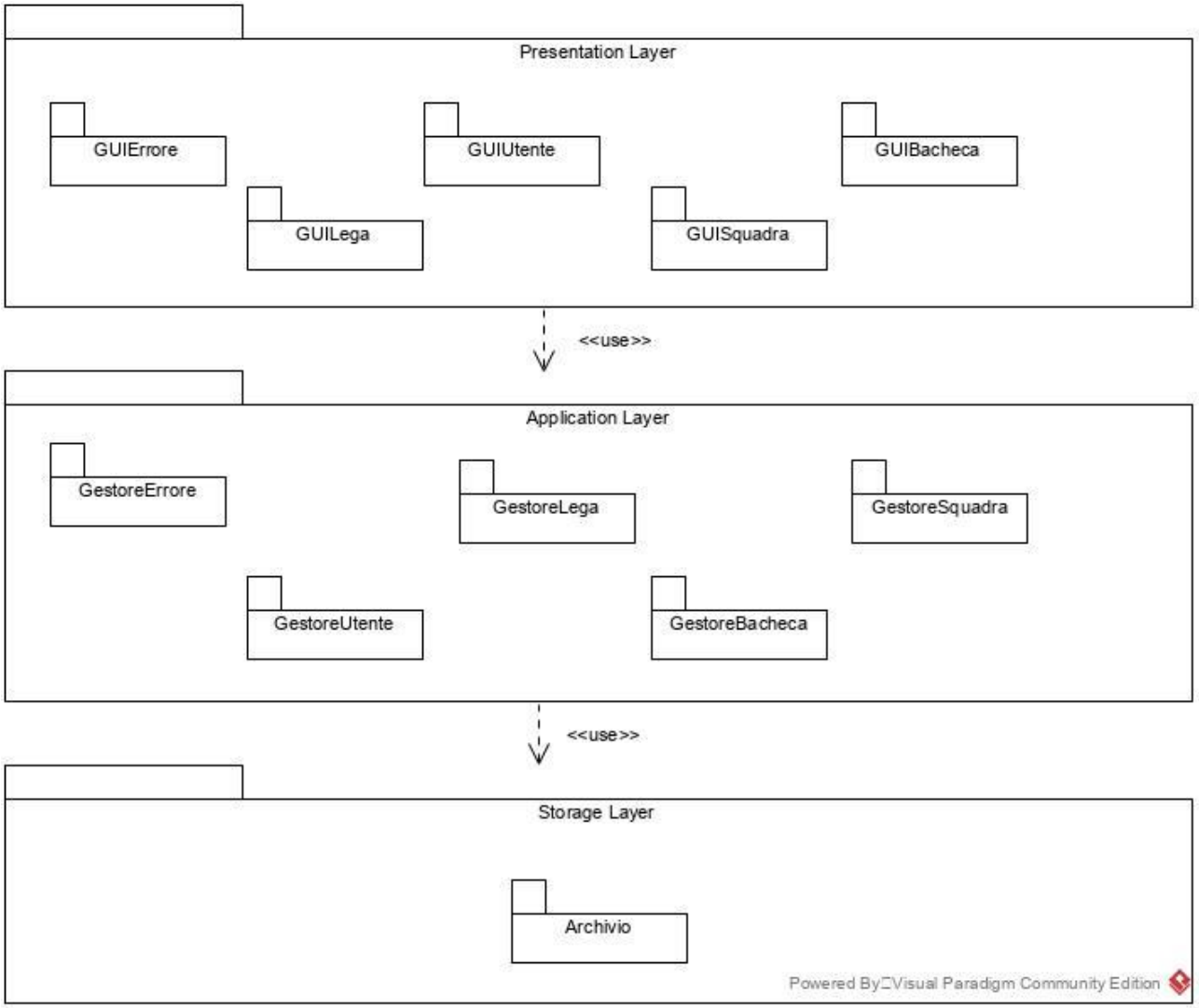
JavaScript: Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente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

1.6 RIFERIMENTI

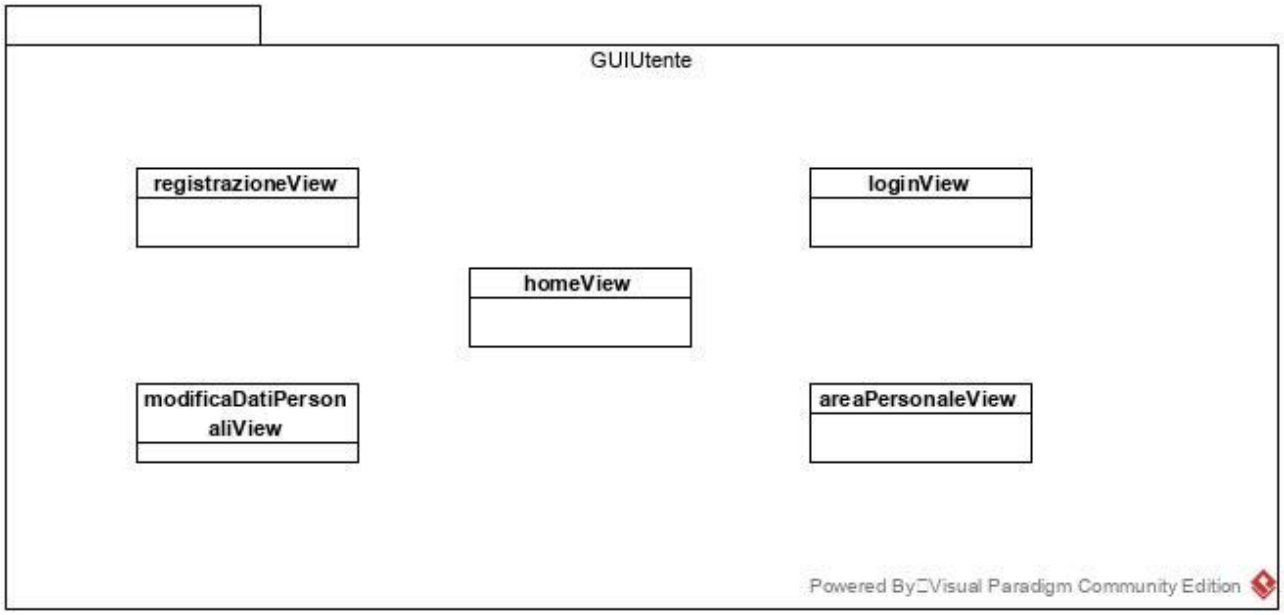
2 PACKAGES



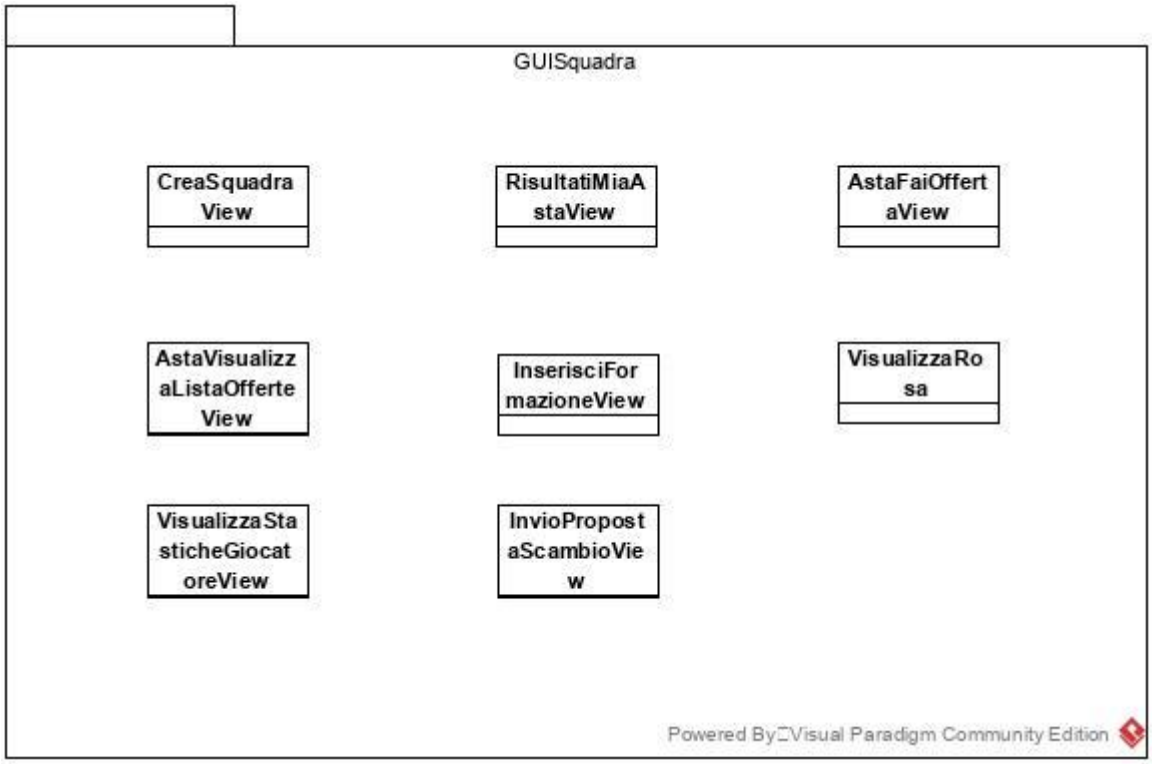
Il diagramma descrive la natura three-layer dell'applicazione mostrandone i tre package principali:

- 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.

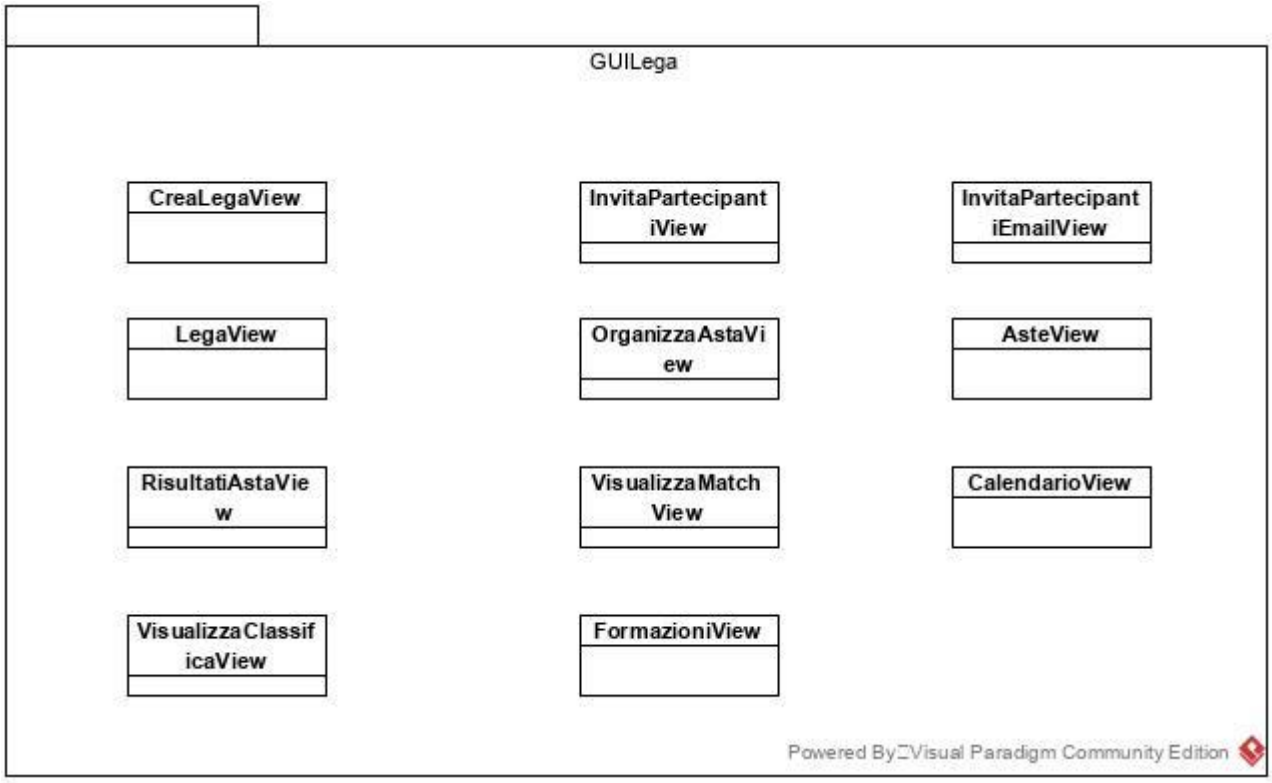
2.1 PACKAGE INTERFACE GESTOREUTENTE



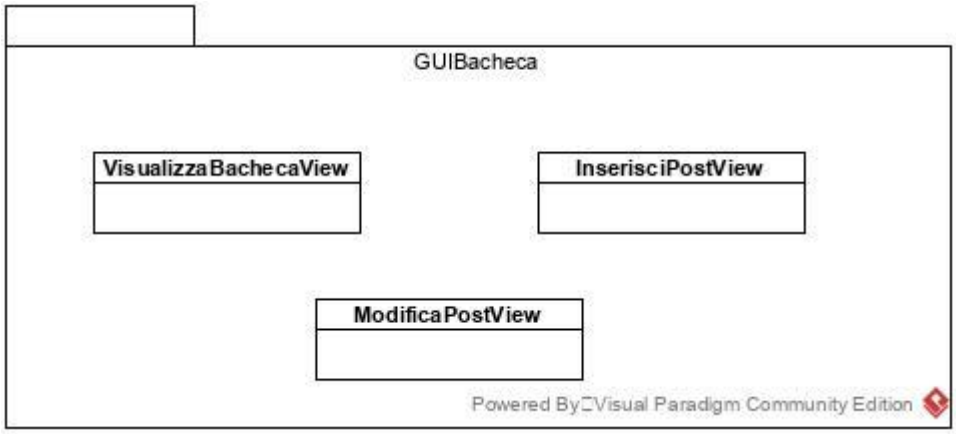
2.2 PACKAGE INTERFACE GESTORELEGA



2.3 PACKAGE INTERFACE GESTORELEGA

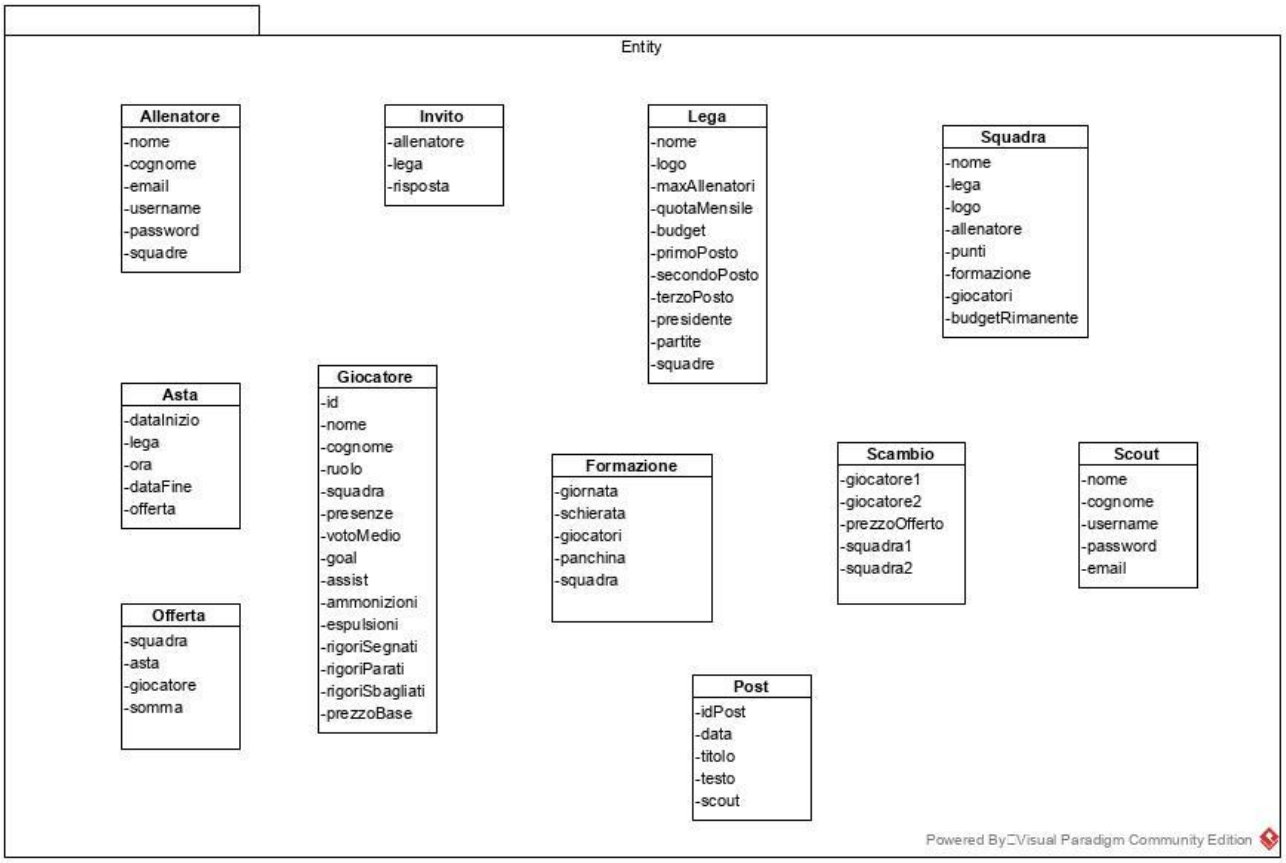


2.4 PACKAGE INTERFACE GESTOREBACHECA



3 INTERFACCE DELLE CLASSI

3.1 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
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) **Post**: email è presente nel database **Context** Allenatore::setUsername (username) **Post**: 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 + setSquadra(squadra: Squadra): void +getAsta():Asta +setAsta(asta: Asta): 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
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 + getLega (): Lega + setLega (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 + getLega (): Lega + setLega (Lega: Lega): 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 +getPrezzoBase(): int +setPrezzoBase(prezzoBase: 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	+getSquadra (): Squadra +setSquadra(nomeSquadra: String): void +getModulo(): String +setModulo(modulo: String): void +getGiornata(): int +setGiornata(giornata: int): void +getSchierata(): boolean +setSchierata(schierata: boolean): void +getGiocatoriSchierati(): Collection +setGiocatoriSchierati(giocatoriSchierati: Collection<Giocatore>): void +getPanchina(): Collection +setPanchina(panchina: Collection<Giocatore>): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Partita
Descrizione	Questa classe rappresenta l'oggetto Formazione
Signature dei metodi	+getSquadra1 (): Squadra +setSquadra1(nomeSquadra1: Squadra): void +getGiornata(): int +setGiornata(giornata: int): void +getSquadra2 (): Squadra +setSquadra2(nomeSquadra2: Squadra): void +getGoal1 (): int +setGoal1(goal1: int): void +getGoal2(): int +setGoal2(goal2: int): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe	Scambio
Descrizione	Questa classe rappresenta l'oggetto Scambio
Signature dei metodi	+getGiocatore1(): Giocatore +setGiocatore1(Giocatore1: Giocatore): void +getGiocatore2(): Giocatore +setGiocatore2(Giocatore2: int): void +getSquadra1(): Squadra +setSquadra1(squadra1: Squadra): void +getSquadra2(): Squadra +setSquadra2(squadra2: Squadra): 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	

3.2 MODEL

Nome classe	AllenatoreDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Allenatore.
Signature dei metodi	+addAllenatore(allenatore: Allenatore): boolean +checkLogin (username: String, password: String): boolean +deleteAllenatore(username: String): boolean +updateAllenatore(allenatore: Allenatore): boolean +getAllAllenatori(): List<Giocatore> +getAllenatoreByUsername(username: String): Allenatore +getAllenatoreByEmail(email: String): Allenatore
Pre-condizioni	**Context** AllenatoreDAO::addAllenatore (nome, cognome, email, username, password) **Pre**: allenatore!=null **Context** AllenatoreDAO::checkLogin(username, password) **Pre**: username!=null and password!=null **Context** AllenatoreDAO::deleteAllenatore (username) **Pre**: username!=null **Context** AllenatoreDAO::updateAllenatore(allenatore) **Pre**: allenatore!=null **Context** AllenatoreDAO::getAllAllenatori() **Pre**: **Context** AllenatoreDAO::getAllenatoreByUsername(username) **Pre**: username!=null **Context** AllenatoreDAO::getAllenatoreByEmail(email) **Pre**: email!=null
Post-condizioni	**Context** AllenatoreDAO::addAllenatore (allenatore) **Post**: database.allenatore->includes(select(a|allenatore.username=allenatore.getUsername())) **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 (allenatore)

	Post: **Context** AllenatoreDAO::getAllAllenatori() **Post**: database.allenatore **Context** AllenatoreDAO::getAllenatoreByUsername(username) **Post**: allenatore->select(a|allenatore.username=username)) **Context** AllenatoreDAO::getAllenatoreByEmail(email) **Post**: allenatore->select(a|allenatore.email=email))
Invariante	

Nome classe	LegaDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Lega.
Signature dei metodi	+addLega(lega: Lega): boolean +getLegaByNome(nome: String): Lega +getLegheByPresidente(allenatore: Allenatore): Collection<Lega> +deleteLega(lega: Lega): boolean
Pre-condizioni	**Context** LegaDAO:: addLega(lega) **Pre**: lega!=null **Context** LegaDAO::getLegaByNome(nome) **Pre**: nome!=null **Context** LegaDAO::getLegheByPresidente(allenatore) **Pre**: allenatore!=null **Context** LegaDAO:: deleteLega(lega) **Pre**: lega!=null
Post-condizioni	**Context** LegaDAO::addLega(lega) **Post**: database.lega->includes(select(l|lega.nome=lega.getNome())) **Context** LegaDAO::getLegaByNome(nome) **Post**: lega->select(l|lega.l=nome)) **Context** LegaDAO::getLegheByPresidente(allenatore) **Post**: leghe->select(l|lega.presidente=allenatore.getUsername())) **Context** LegaDAO:: deleteLega(lega) **Post:** database.lega-> not includes(select(l|lega.nome=lega.getNome()))
Invariante	

Nome classe	InvitoDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Invito.
Signature dei metodi	+addInvito(invito: Invito): boolean +updateInvito(invito: Invito): void +getInvitoByAllenatore(allenatore: String): List<Invito> +deleteInvito(invito: Invito): boolean +getInvitoById(allenatore: Allenatore, lega: Lega): Invito
Pre-condizioni	**Context** InvitoDAO::addInvito(invito) **Pre**: invito!=null **Context** InvitoDAO::updateInvito(invito) **Pre**: invito!=null

	Context InvitoDAO::getAllInvitoAllenatore(allenatore) **Pre**: allenatore!=null **Context** InvitoDAO::deleteInvito(invito) **Pre**: invito!=null **Context** InvitoDAO::getInvitoById(allenatore, lega) **Pre**: allenatore!=null and lega!=null
Post-condizioni	**Context** InvitoDAO::addInvito(invito) **Post**: database.invito->includes(select(i|invito.allenatore=invito.getAllenatore().getUsername() and invito.nomeLega=invito.getLega().getNome())) **Context** InvitoDAO::updateInvito(invito) **Post**: **Context** InvitoDAO::getInvitoByAllenatore(allenatore) **Post**: inviti->select(i|invito.allenatore=allenatore) **Context** InvitoDAO::deleteInvito(invito) **Post**: database.invito-> not includes(select(i|invito.allenatore=invito.getAllenatore().getUsername() and invito.nomeLega=invito.getLega().getNome())) **Context** InvitoDAO::getInvitoById(allenatore, lega) **Post:** invito->select (i| invito.allenatore=allenatore and invito.lega=lega)
Invariante	

Nome classe	SquadraDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Squadra.
Signature dei metodi	+ creaSquadra(squadra: Squadra): boolean +updateSquadra(squadra: Squadra): boolean + addGiocatoreSquadra(nomeSquadra: String, nomeLega: String, giocatore: int): boolean + deleteGiocatoreSquadra(nomeSquadra: String, nomeLega: String, giocatore: int): void + getSquadreByAllenatore(allenatore: String): List<Squadra> + getSquadraById (nome: String, lega: String): Squadra + getSquadreByLega(lega: String): List<Squadra> + getSquadreGiocatore(giocatore: Giocatore): Collection<Squadra > + getSquadraByUserELega(user: String, nomeLega: String): Squadra
Pre-condizioni	**Context** SquadraDAO::creaSquadra(squadra) **Pre**: squadra!=null **Context** SquadraDAO::updateSquadra(squadra)

	Pre: squadra!=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::getSquadraById(nome, lega) **Pre**: nome!=null and lega!=null **Context** SquadraDAO::getSquadreByLega(lega) **Pre**: lega!=null **Context** SquadraDAO::getSquadreGiocatore(giocatore) **Pre**: giocatore!=null **Context** SquadraDAO::getSquadraByUserELega(user, nomeLega) **Pre**: user!=null and nomeLega!=null
Post-condizioni	**Context** SquadraDAO::creaSquadra(squadra) **Post**: database.squadra->includes(select(s|squadra.nome=squadra.getNome() and squadra.nomeLega=squadra.getLega.getNome())) **Context** SquadraDAO::updateSquadra(squadra) **Post:** **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::getSquadraById(nome, lega) **Post**: squadre->select(s|squadra.nome=nome and squadra.nomeLega=lega) **Context** SquadraDAO::getSquadreByLega(lega) **Post**: squadre->select(s|squadra.lega=lega) **Context** SquadraDAO::getSquadreGiocatore(giocatore) **Post**: squadre->select(s|squadragiocatore.giocatore=giocatore.id) **Context** SquadraDAO::getSquadraByUserELega(user, nomeLega) **Post**: squadra->select(s|squadra.allenatore=user and squadra.lega=nomeLega)
Invariante	

Nome classe	PartitaDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Partita.
Signature dei metodi	+addPartita(partita: Partita): boolean + getPartitaById (squadra1: Squadra, squadra2: Squadra, giornata: int): Partita +updatePartita (partita: Partita): boolean +getAllPartiteLega(lega: String): List<Partita> +getAllPartiteSquadra(squadra: Squadra): List<Partita> + getAllPartiteByGiornataLega(giornata: int, lega:String): List<Partita>
Pre-condizioni	**Context** PartitaDAO::addPartita(partita) **Pre**: partita!=null **Context** PartitaDAO::getPartitaById (squadra1, squadra2, giornata) **Pre**: squadra1!=null and squadra2!=null and giornata!=null **Context** PartitaDAO::updatePartita (partita) **Pre**: partita!=null **Context** PartitaDAO::getAllPartiteLega(lega) **Pre**: lega!=null **Context** PartitaDAO::getAllPartiteSquadra(squadra) **Pre**: squadra!=null **Context** PartitaDAO:: getAllPartiteByGiornataLega(giornata, lega) **Pre**: giornata!=null and lega!=null
Post-condizioni	**Context** PartitaDAO::addPartita(partita) **Post**: database.partita-

	>includes(select(p|partita.nomeSquadra1=partita.getSquadra1.getNome() and partita.nomSquadra2=partita.getSquadra2().getNome() and partita.giornata=partita.getGiornata() and partita.nomeLega=partita.getSquadra1().getLega().getNome() and partita.nomeLega= partita.getSquadra2().getLega().getNome())) **Context** PartitaDAO::getPartitaById (squadra1, squadra2, giornata) **Post**: partita->select(p|partita.squadra1=squadra1.getNome() and partita.squadra2=squadra2.getNome() and partita.giornata=giornata and partita.nomeLega=squadra1.getNomeLega() or partita.nomeLega=squadra2.getNome()) **Context** PartitaDAO::updatePartita (partita) **Post**: **Context** PartitaDAO::getAllPartiteLega(lega) **Post**: partite->select(p|partita.squadra1.nomeLega=lega) **Context** PartitaDAO::getAllPartiteSquadra(squadra) **Post**: partite-> select(p|partita.nomeSquadra1=squadra.getNome() and partita.nomeLega=squadra.getLega().getNome() or partita.nomeSquadra2=squadra.getNome()) **Context** PartitaDAO::getAllPartiteByGiornata (giornata, lega) **Post**: partite->select(p|partita.giornata=giornata and partita.nomeLega=lega)
Invariante	

Nome classe	OffertaDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Offerta.
Signature dei metodi	+ addOfferta (offerta: Offerta): boolean + deleteOfferta(offerta: Offerta):void + updateOfferta(offerta: Offerta): void + getAllOfferteByAsta(dataInizioAsta: Date, nomeLega: String): List<Offerta> + getAllOfferteBySquadra (nomeSquadra: String, nomeLega: String): List<Offerta> + getAllOfferteByAstaAllenatore(dataInizioAsta: Date, nomeLega: String, allenatore: String): List<Offerta> + getOfferteGiocatoreAsta(giocatore: int, dataInizioAsta: Date, nomeLega: String): List<Offerta> + getOffertaByKey (giocatore: int, dataInizioAsta: Date, nomeLega: String, nomeSquadra: String): Offerta
Pre-condizioni	**Context** OffertaDAO::addOfferta (offerta)

	Pre: offerta!=null **Context** OffertaDAO:: deleteOfferta(offerta) **Pre**: offerta!=null **Context** OffertaDAO:: updateOfferta(offerta) **Pre**: offerta!=null **Context** OffertaDAO:: getAllOfferteBySquadra(nomeSquadra, nomeLega) **Pre**: nomeSquadra!=null and nomeLega!=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:: getOffertaByKey (giocatore, dataInizioAsta, nomeLega,nomeSquadra, nomeLega) **Pre**: giocatore!=null and dataInizioAsta!=null and nomeLega!=null and nomeSquadra != null
Post-condizioni	**Context** OffertaDAO:: addOfferta (offerta) **Post**: database.offerta->includes(select(o|offerta.nomeSquadra= offerta.getSquadra().getNome() and offerta.dataAsta= offerta.getAsta().getDataInizio() and offerta.nomeLega= offerta.getAsta().getLega().getNome() and offerta.giocatore= offerta.getGiocatore().getId() and offerta.somma= offerta.getSomma())) **Context** OffertaDAO:: deleteOfferta(offerta) **Post**: database.offerta-> not includes(select(o|offerta.nomeSquadra= offerta.getSquadra().getNome() and offerta.dataAsta= offerta.getAsta().getDataInizio() and offerta.nomeLega= offerta.getAsta().getLega().getNome() and offerta.giocatore=gofferta.getGiocatore().getId())) **Context** OffertaDAO:: updateOfferta(offerta) **Post**: **Context** OffertaDAO:: getAllOfferteBySquadra(nomeSquadra, nomeLega) **Post**: offerte-> select(o|offerta.nomeSquadra=nomeSquadra and

	offerta.nomeLega=nomeLega) **Context** OffertaDAO:: getAllOfferteByAstaSquadra(dataInizioAsta, nomeLega, squadra) **Post**: offerte-> select(o|offerta.dataInizioAsta=dataInizioAsta and offerta.nomeLega=nomeLega and offerta.nomSquadra=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:: getOffertaByKey (giocatore, dataInizioAsta, nomeLega, nomeSquadra) **Pre**: offerte-> select(o|offerta.giocatore=giocatore and offerta.dataInizioAsta=dataInizioAsta and offerta.nomeLega=nomeLega and offerta.nomeSquadra = nomeSquadra)
Invariante	

Nome classe	AstaDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Asta.
Signature dei metodi	+ addAsta(asta: Asta): boolean + getAsteByAllenatore(allenatore: String): List<Asta> + getAstaByKey(dataInizio: Date, nomeLega: String): Asta +getAsteByLega(lega: Lega): List<Asta> +getAsteScaduteOggi(): List<Asta>
Pre-condizioni	**Context** AstaDAO::addAsta(asta) **Pre**: asta!=null and asta.getDataInizio()<asta.getDataFine() **Context** AstaDAO::getAsteByAllenatore(allenatore) **Pre**: allenatore!=null **Context** AstaDAO::getAstaByKey (dataInizio, nomeLega) **Pre**: dataInizio!=null and nomeLega!=null **Context** AstaDAO::getAsteByLega(lega) **Pre**: lega!=null **Context** AstaDAO::getAsteScaduteOggi () **Pre**:

Post-condizioni	**Context** AstaDAO::addAsta(asta) **Post**: database.asta->includes(select(a|asta.dataInizio= asta.getDataInizio()< and asta.ora=asta.getOra() and asta.dataFine= asta.getDataFine() and asta.nomeLega=asta.getLega().getNomeLega())) **Context** AstaDAO:: getAsteByAllenatore(allenatore) **Post**: aste->select(a|offerta.allenatore.getNome()=allenatore) **Context** AstaDAO:: getAstaByKey (dataInizio, nomeLega) **Post**: asta->select(a|asta.dataInizio=dataInizio and asta.nomeLega=nomeLega) **Context** AstaDAO::getAsteByLega(lega) **Post**: aste->select(a|asta.nomeLega=lega.getNome()) **Context** AstaDAO::getAsteScaduteOggi () **Post**: aste->select(a|asta.dataFine=Date.today())
Invariante	

Nome classe	GiocatoreDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Giocatore.
Signature dei metodi	+ addGiocatore(giocatore: Giocatore): boolean + aggiornaGiocatore(giocatore: Giocatore): boolean + getGiocatoreById(id: int): Giocatore + getPopular(prezzo: int): List<Giocatore>
Pre-condizioni	**Context** GiocatoreDAO::addGiocatore(giocatore) **Pre**: giocatore!=null **Context** GiocatoreDAO::aggiornaGiocatore(giocatore) **Pre**: **Context** GiocatoreDAO::getGiocatoreById(id) **Pre**: id!=null **Context** GiocatoreDAO::getPopular (prezzo) **Pre**: prezzo!=null
Post-condizioni	**Context** GiocatoreDAO::addGiocatore(giocatore) **Pre**: database.giocatore->includes(select(g|g.id=giocatore.id)) **Context** GiocatoreDAO::aggiornaGiocatore(giocatore) **Post**: **Context** GiocatoreDAO::getGiocatoreById(id) **Pre**: giocatore->select(g|giocatore.id=id)

	Context GiocatoreDAO::getPopular (prezzo) **Pre**: giocatori->select(g|giocatore.prezzoBase>prezzo)
Invariante	

Nome classe	FormazioneDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Formazione.
Signature dei metodi	+ addFormazione (formazione: Formazione): boolean + addGiocatoreFormazione(formazione: Formazione, giocatore: Giocatore): boolean + deleteGiocatoreFormazione(formazione: Formazione, giocatore: Giocatore): boolean + updateGiocatoreFormazione(formazione: Formazione, giocatore1: Giocatore, giocatore2: Giocatore): boolean + updateFormazione (formazione: Formazione): boolean + getFormazioneBySquadraGiornata (Squadra: squadra, giornata: int): Formazione
Pre-condizioni	**Context** FormazioneDAO::addFormazione (formazione) **Pre**: formazione!=null **Context** FormazioneDAO::addGiocatoreFormazione(formazione, giocatore) **Pre**: formazione!=null and giocatore!=null **Context** FormazioneDAO::deleteGiocatoreFormazione(formazione, giocatore) **Pre**: formazione!=null and giocatore!=null **Context** FormazioneDAO::updateGiocatoreFormazione(formazione, giocatore) **Pre**: formazione!=null and giocatore!=null **Context** FormazioneDAO::updateFormazione (formazione) **Pre**: formazione!=null **Context**FormazioneDAO::getFormazioneBySquadraGiornata (squadra, giornata) **Pre**: squadra!=null and giornata!=null
Post-condizioni	**Context** FormazioneDAO::addFormazione (formazione) **Post**: database.formazione->includes(select(f|formazione.nomeSquadra=formazione.getSquadra().getNome() and formazione.nomeLega=formazione.getSquadra().getLega().getNome() and formazione.modulo=formazione.getModulo() and formazione.giornata=formazione.getGiornata() and formazione.schierata=formazione.getSchierata())) **Context** FormazioneDAO::addGiocatoreFormazione(formazione, giocatore) **Post**: database.giocatoreFormazione->includes(select(f|giocatoreFormazione.nomeSquadra=formazione.getSquadra().getNome() and giocatoreFormazione.nomeLega= formazione.getSquadra().getLega().getNome() and giocatoreFormazione.giornata= formazione.getGiornata() and giocatoreFormazione.giocatore=giocatore.getId()))

	Context FormazioneDAO::deleteGiocatoreFormazione(formazione, giocatore) **Post**: database.giocatoreFormazione-> not includes(select(f|giocatoreFormazione.nomeSquadra=formazione.getSquadra().getNome() and giocatoreFormazione.nomeLega= formazione.getSquadra().getLega().getNome() and giocatoreFormazione.giornata= formazione.getGiornata() and giocatoreFormazione.giocatore=giocatore.getId())) **Context** FormazioneDAO::updateGiocatoreFormazione(formazione, giocatore1, giocatore2) **Post**: database.giocatoreFormazione-> includes(select(f|giocatoreFormazione.nomeSquadra=formazione.getSquadra().getNome() and giocatoreFormazione.nomeLega= formazione.getSquadra().getLega().getNome() and giocatoreFormazione.giornata= formazione.getGiornata() and giocatoreFormazione.giocatore=giocatore2.getId())) **Context** FormazioneDAO::updateFormazione (formazione) **Post**: **Context**FormazioneDAO::getFormazioneBySquadraGiornata (squadra, giornata) **Post**: formazione-> select(f|formazione.nomeSquadra=squadra.getNome() and formazione.nomeLega=squadra.getLega().getNome() 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 riguardanti Scambio.
Signature dei metodi	+ creaScambio (scambio: Scambio): boolean + accettaScambio (scambio: Scambio): boolean + rifiutaScambio (scambio: Scambio): void + getScambiNonAccettatiSquadra (squadra: Squadra): List<Scambio> + getScambioById (giocatore1: Giocatore, squadra1: Squadra, giocatore2: Giocatore, squadra2: Squadra): Scambio
Pre-condizioni	**Context** ScambioDAO::creaScambio (scambio) **Pre**: scambio!=null **Context** ScambioDAO::accettaScambio (scambio) **Pre**: scambio!=null and scambio.getRisposta==true **Context** ScambioDAO::rifiutaScambio (scambio) **Pre**: scambio!=null and scambio.getRisposta==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 (scambio) **Post**: database.scambio-> includes(select(s|scambio.giocatore1=scambio.getGiocatore1().getId() and scambio.nomeSquadra1=scambio.getSquadra1().getNome() and scambio.nomeLega=scambio.getSquadra1().getLega().getNome() and scambio.giocatore2= scambio.getGiocatore2().getId() and scambio.nomeSquadra2= scambio.getSquadra2().getNome())) **Context** ScambioDAO::accettaScambio (scambio **Post**: database.squadraGiocatore->includes(select(g| squadraGiocatore.nomeSquadra=scambio.getSquadra1().getNome() and squadraGiocatore.nomeLega=scambio.getSquadra1().getLega().getNome() and squadraGiocatore.giocatore=scambio.getGiocatore2().getId()) and (select(g| squadraGiocatore.nomeSquadra= scambio.getSquadra2().getNome() and squadraGiocatore.nomeLega= scambio.getSquadra2().getLega().getNome() and squadraGiocatore.giocatore=scambio.getGiocatore1().getId()) **Context** ScambioDAO::rifiutaScambio (scambio) **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 riguardanti Scout.
Signature dei metodi	+ checkLogin(username: String, password: String): boolean + updateScout (scout: Scout): boolean +getScoutByUsername(username: String): Scout

	+getScoutByEmail(email: String): Scout
Pre-condizioni	**Context** ScoutDAO::checkLogin(username, password) **Pre**: username!=null and password!=null **Context** ScoutDAO:: updateScout (scout) **Pre**: scout!=null **Context** ScoutDAO:: getScoutByUsername (username) **Pre**: username!=null **Context** ScoutDAO:: getScoutByEmail (email) **Pre**: email!=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(scout) **Post**: **Context** ScoutDAO:: getScoutByUsername (username) **Post**: scout->select(s|scout.username=username) **Context** ScoutDAO:: getScoutByEmail (email) **Post**: scout->select(s|scout.email=email)
Invariante	

Nome classe	PostDAO
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti Post.
Signature dei metodi	+ addPost(post: Post): boolean + removePost(id: int): boolean + updatePost(post: Post): boolean + getPostByScout(scout: String): List<Post> + getAllPost(): List<Post> + getPostById(idPost: int): Post
Pre-condizioni	**Context** PostDAO::addPost(post) **Pre**: post!=null **Context** PostDAO::removePost(id) **Pre**: id!=null **Context** PostDAO::updatePost(post) **Pre**: post!=null **Context** PostDAO::getPostByScout(scout)

	Pre: scout!=null **Context** PostDAO::getAllPost() **Pre**: **Context** PostDAO::getPostById (idPost) **Pre**:idPost!=null
Post-condizioni	**Context** PostDAO::addPost(post) **Post:** database.post-> includes(select(p| p.id=post.getId())) **Context** PostDAO::removePost(id) **Post**: database.post->not includes(select(p| p.id=id)) **Context** PostDAO::updatePost(post) **Post**: **Context** PostDAO::getPostByScout(scout) **Post**: post->select(p|post.scout=scout) **Context** PostDAO::getAllPost() **Post**: database.post **Context** PostDAO::getPostById (idPost) **Post**: post->select(p|post.idPost=idPost)
Invariante	

3.3 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("testo")!=null and request.getParameter("titolo")!=null And request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("scout")
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 And request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("scout")
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 And request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("scout")
Post-condizioni	**Context** ModificaPostServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: PostDAO.getPostById(idPost).testo == testo
Invariante	

Nome classe	getAllPostServlet
Descrizione	Questa classe è un control che si occupa di prendere tutti i post
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** getAllPostServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**:

Post-condizioni	**Context** getAllPostServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getSession().getAttribute("allPost")!=null
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.getPartitaById(squadra1,squadra2,giornata) == null
Post-condizioni	**Context** InserisciMatchServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: PartitaDAO.getPartitaById(squadra1.nome, squadra1.nomeLega, squadra2.nome, squadra2.nomeLega, 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.getPartitaById(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).getPunteggioS

	quadra2()==punteggioSquadra2
Invariante	

Nome classe	getLegaServlet
Descrizione	Questa classe è un control che si occupa di caricare dati in sessione relativi alla lega che si vuole visualizzare
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** getLegaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getParameter("q")!=null And request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore")
Post-condizioni	**Context** getLegaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getSession().getAttribute("lega")!=null and request.getSession().getAttribute("classifica")!=null and request.getSession().getAttribute("formazioni")!=null and request.getSession().getAttribute("calendario")!=null and request.getSession().getAttribute("aste")!=null
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	getFormazioneSquadraServlet
Descrizione	Questa classe è un control che si occupa di salvare in sessione la formazione attuale della squadra
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** getFormazioneSquadraServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") And request.getSession().getAttribute("lega")!=null
Post-condizioni	**Context** getFormazioneSquadraServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getSession().getAttribute("formazione")!=null
Invariante	

Nome classe	InserisciGiocatoreFormazioneServlet
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** InserisciGiocatoreFormazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and Request.getParameter("modulo")!=null and request.getParameter("titolare")!=null and request.getParameter("giocatore")!=null
Post-condizioni	**Context** InserisciGiocatoreFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).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.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") request.getParameter("giocatore")!=null and

	SquadraDAO.getSquadraByKey(squadra).getGiocatori().contains(giocatore) == true
Post-condizioni	**Context** RimuoviGiocatoreFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: SquadraDAO.getSquadraById(squadra.nome, squadra.nomeLega().getNome()).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.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and request.getParameter("giocatore1")!=null and request.getParameter("giocatore2")!=null and SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).getGiocatori().contains(giocatore1) == true
Post-condizioni	**Context** SostituisciGiocatoreFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).getFormazione().getGiocatori().contains(giocatore1) == false and SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).getFormazione().getGiocatori().contains(giocatore2) == true
Invariante	

Nome classe	SalvaFormazioneServlet
Descrizione	Questa classe è un control che si occupa di passare a FormazioneDAO i dati di una formazione da salvare.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** SalvaFormazioneServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and

	Request.getSession().getAttribute("tipo").equals("allenatore") And SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).getFormazione().getGiocatori().lenght==11 and SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).getFormazione().getPanchina().lenght==7
Post-condizioni	**Context** SalvaFormazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: SquadraDAO.getSquadraById(squadra.getNome(), squadra.getLega().getNome()).getFormazione().getSchierata()==true
Invariante	

Nome classe	getRosaServlet
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** getRosaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and request.getParameter("s") != null
Post-condizioni	**Context** getRosaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getSession().getAttribute("rosa")!=null and request.getSession().getAttribute("squadra")!=null
Invariante	

Nome classe	PrendiOfferteServlet
Descrizione	Questa classe è un control che si occupa di prendere le offerte di una squadra in una certa asta.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** PrendiOfferteServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and Request.getParameter("data")!=null and request.getParameter("lega")!=null
Post-condizioni	**Context** PrendiOfferteServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getAttribute("offerte")!=null
Invariante	

Nome classe	FaiOffertaServlet
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.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and Request.getParameter("data")!= null and request.getParameter("lega")!=null and request.getParameter("giocatore") != null and request.getParameter("sommaOfferta") != null and SquadraDAO.getSquadraById(squadra, lega).getBudgetRimanente()>=sommaOfferta
Post-condizioni	**Context** OffertaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: OffertaDAO.getOffertaGiocatoreAllenatore(giocatore, data, lega, squadra)!=null
Invariante	

Nome classe	ModificaOffertaServlet
Descrizione	Questa classe è un control che si occupa di passare a OffertaDAO i dati di un'offerta da modificare.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** OffertaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and Request.getParameter("data")!= null and request.getParameter("lega")!=null and request.getParameter("giocatore") != null and request.getParameter("sommaOfferta") != null and OffertaDAO.getOffertaGiocatoreAllenatore(giocatore, data, lega, squadra)!=null
Post-condizioni	**Context** OffertaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: OffertaDAO.getOffertaGiocatoreAllenatore(giocatore, data, lega, squadra).getSomma()=sommaOfferta
Invariante	

Nome classe	CancellaOffertaServlet
Descrizione	Questa classe è un control che si occupa di passare a OffertaDAO i dati di un'offerta da cancellare.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** OffertaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and request.getParameter("data")!= null and request.getParameter("lega")!=null and request.getParameter("giocatore") != null and OffertaDAO.getOffertaGiocatoreAllenatore(giocatore, data, lega, squadra)!=null
Post-condizioni	**Context** OffertaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: OffertaDAO.getOffertaGiocatoreAllenatore(giocatore, data, lega, squadra)=null
Invariante	

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 (addAllenatore) 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 formato "^[A-Za-z]{2,50}$"AND Request.getParameter("cognome")!=null e rispetta il formato "^[A-Za-z]{2,50}$" AND Request.getParameter("username")!=null e rispetta il formato ^[a-zA-Z0-9]+([a-zA-Z0-9](_|-|)[a-zA-Z0-9])*[a-zA-Z0-9]+$" AND Request.getParameter("password")!=null and rispetta il formato "^([A-Za-z0-9]){5,}$" AND Request.getParameter("email")!=null e rispetta il formato "^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$" AND allenatoreDAO.getAllenatoreByUsername(username)==null AND allenatoreDAO.getAllenatoreByEmail(email)==null)
Post-condizioni	**Context** RegistrazioneServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: AllenatoreDAO.getAllenatoreByUsername(username)!=null

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 quest'ultime (checkLogin) e se verificate provvederà a creare una sessione per l'utente
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** LoginServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: Request.getParamter("username")!=null Request.getParameter("password")!=null
Post-condizioni	**Context** LoginServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**:AllenatoreDAO.request.getSession().getAttribute("utente")!=null
Invariante	

Nome classe	ModificaDatiPersonaliServlet
Descrizione	Questa classe è un control che si occupa di cambiare le credenziali dell'utente, prima verifica che queste rispettino gli standard e poi chiama il dao addetto all'update delle credenziali.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** ModificaDatiPersonaliServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getParameter("password")!=null and rispetta il formato "^([A-Za-z0-9]){5,}$" and Request.getParameter("email")!=null and rispetta il formato "^([a-zA-Z0-9_\-\.]+)@([a-zA-Z0-9_\-\.]+)\.([a-zA-Z]{2,5})$"and non dev'essere presente nel database.
Post-condizioni	**Context** ModificaDatiPersonaliServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: password e/o email sono stati aggiornati
Invariante	

Nome classe	CancellaProfiloServlet
Descrizione	Questa classe è un control che si occupa di cancellare l'account di un certo utente

Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** CancellaUtenteServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null
Post-condizioni	**Context** CancellaUtenteServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: AllenatoreDAO.getAllenatoreByUsername(username)==null
Invariante	

Nome classe	CreaSquadraServlet
Descrizione	Questa classe è un control che si occupa di verificare se i dati inseriti per la creazione della squadra sono validi, se validi, il control invocherà il DAO adatto alla memorizzazione della squadra. (CreaSquadra)
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** CreaSquadraServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and request.getParameter("Nome")!=null and rispetta il formato "^{4,50}$" and non deve essere presente nella lega. Request.getParameter("logo") rispetta il formato "([^\s]+(\.(?i)(jpg\|png\|img\|))$)" and request.getParameter("nomeLega")!=null
Post-condizioni	**Context** CreaSquadraServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: SquadraDAO.getSquadraById(nome, nomeLega)!=null
Invariante	

Nome classe	CreaLegaServlet
Descrizione	Questa classe è un control che si occupa di verificare se i dati inseriti per la creazione della lega sono validi, se validi, il control invocherà il DAO adatto alla memorizzazione della lega.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** CreaLegaServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and request.getParameter("Nome")!=null and rispetta il formato "^{4,50}$" and non deve essere presente nella lega. Request.getParameter("logo")

	rispetta il formato “([^\s]+(\.(?i)(jpg|png|img|))$)”. Request.getParameter(“Quota”)!=null and rispetta il formato “^[0-9]{0,2}*$”. Request.getParameter(“primoPosto”)!=null and rispetta il formato “^[0-9]{0,2}*$”. Request.getParameter(“secondoPosto”)!=null and rispetta il formato “^[0-9]{0,2}*$”. Request.getParameter(“terzoPosto”)!=null and rispetta il formato “^[0-9]{0,2}*$”. Request.getParameter(“maxAllenatori”)!=null and 4<=maxAllenatori<=10
Post-condizioni	**Context** CreaLegaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: LegaDAO.getLegaByNome(nome)!=null
Invariante	

Nome classe	UniscitiAllaLegaServlet
Descrizione	Questa classe è un control che si occupa di verificare la risposta dell’allenatore all’invito ad una lega. Se positiva, verrà chiamato il DAO addetto all’associazione di un allenatore ad una lega (UpdateInvito), se negativa, verrà chiamato il DAO addetto all’eliminazione dell’invito.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** UniscitiAllaLegaServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute(“utente”)!=null and Request.getSession().getAttribute(“tipo”).equals(“allenatore”) and request.getParameter(“Risposta”)!=null and typeof(Risposta)=boolean. Request.getParameter(“Lega”)!=null.
Post-condizioni	**Context** UniscitiAllaLegaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: InvitoDAO.getInvitoById(utente, Lega).getRisposta()==true
Invariante	

Nome classe	OrganizzaAstaServlet
Descrizione	Questa classe è un control che si occupa di verificare se i dati inseriti dall’utente siano accettabili, se positivi, il control chiamerà il DAO addetto alla creazione dell’asta(AddAsta).
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void

Pre-condizioni	**Context** OrganizzaAstaServlet::doGet(request:HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and request.getParameter("dataInizioAsta")!=null and request.getParameter("oraInizioAsta")!=null and Request.getParamter("dataFineAsta")!=null and dataFineAsta>dataInizioAsta and request.getParameter("nomeLega")!=null
Post-condizioni	**Context** CreLegaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: AstaDAO.getAstaByKey(dataInizioAsta, nomeLega)!=null
Invariante	

Nome classe	filtraGiocatoriServlet
Descrizione	Questa classe è un control che si occupa di filtrare i giocatori
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** filtraGiocatoriServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") and Request.getParameter("prezzoBase")!=null And request.getParameter("squadra")!=null and request.getParamter("ruolo")!=null and request.getParameter("p")!=null
Post-condizioni	**Context** filtraGiocatoriServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getSession().getAttribute("giocatori")!=null
Invariante	

Nome classe	PrendiGiocatoreServlet
Descrizione	Questa classe è un control che si occupa di prendere i dati di un giocatore
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** PrendiGiocatoreServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute("utente")!=null and Request.getSession().getAttribute("tipo").equals("allenatore") And request.getParameter("id")
Post-condizioni	**Context** PrendiGiocatoreServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getAttribute("giocatore")!=null
Invariante	

Nome classe	LogoutServlet
Descrizione	Questa classe è un control che si occupa di effettuare il logout di un utente.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** LogoutServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute(“utente”)!=null
Post-condizioni	**Context** LogoutServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getSession().getAttribute(“utente”)==null
Invariante	

Nome classe	getRisultatiAstaServlet
Descrizione	Questa classe è un control che si occupa di prendere i risultati di una certa Asta
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** getRisultatiAstaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute(“utente”)!=null and Request.getSession().getAttribute(“tipo”).equals(“allenatore”) And request.getParameter(“q”)!=null
Post-condizioni	**Context** getRisultatiAstaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getAttribute(“risultatiAsta”)!=null And request.setAttribute(“asta”)!=null
Invariante	

Nome classe	getRisultatiMiaAstaServlet
Descrizione	Questa classe è un control che si occupa di prendere i risultati di una certa Asta di un certo Allenatore.
Signature dei metodi	+doGet(request: HttpServletRequest, response: HttpServletRequest):void
Pre-condizioni	**Context** getRisultatiMiaAstaServlet:: doGet(request: HttpServletRequest, response: HttpServletRequest) **Pre**: request.getSession().getAttribute(“utente”)!=null and Request.getSession().getAttribute(“tipo”).equals(“allenatore”) And request.getParameter(“q”)!=null and request.getSession().getAttribute(“lega”)!=null
Post-condizioni	**Context** getRisultatiMiaAstaServlet::doGet(request: HttpServletRequest, response: HttpServletRequest) **Post**: request.getAttribute(“giocatoreSquadra”)!=null

	And request.setAttribute(“asta”)!=null
Invariante