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 |  |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invariante |  |

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

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

|  |  |
| --- | --- |
| Nome classe | GestoreAllenatore |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni | **Context** GestoreAllenatore::registrazione(nome, cognome, email, username, password)  **Pre**: nome!=null and cognome!=null and email!=null and username!=null and password!=null  **Context** GestoreAllenatore::login(username, password)  **Pre**: username!=null and password!=null  **Context** GestoreAllenatore::cancellaProfilo(username)  **Pre**: GestoreAllenatore.getAllAllenatori()->include(getAllenatoreByUsername(username))  **Context** GestoreAllenatore::modificaDatiPersonali(nome, cognome, email, username, password)  **Pre**: nome!=null and cognome!=null and email!=null and username!=null and password!=null  **Context** GestoreAllenatore::getAllAllenatori()  **Context** GestoreAllenatore::getAllenatore(username) |
| Post-condizioni | **Context** GestoreAllenatore::registrazione(nome, cognome, email, username, password)  **Post**: allenatore.nome=nome and allenatore.cognome=cognome and allenatore.email=email and allenatore.username=username and allenatore.password=password and Allenatore.getAllAllenatori()->include(allenatore)  **Context** GestoreAllenatore::login(username, password)  **Context** GestoreAllenatore::cancellaProfilo(username)  **Post**: GestoreAllenatore.getAllAllenatori()-> not include(getAllenatoreByUsername(username))  **Context** GestoreAllenatore::modificaDatiPersonali(nome, cognome, email, username, password)  **Post**: email=newEmail or password=newPassword  **Context** GestoreAllenatore::getAllAllenatori()  **Post**: database.allenatore  **Context** GestoreAllenatore::getAllenatoreByUsername(username)  **Post**: allenatore=database.allenatore -> select(allenatore.username=username)) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | GestoreLega |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni | **Context** GestoreLega::creaLega(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** GestoreLega::aggiungiSquadra(squadra, allenatore)  **Pre**: squadra!=null and allenatore!=null  **Context** GestoreLega::aggiungiPartita(partita)  **Pre**: partita!=null |
| Post-condizioni | **Context** GestoreLega::creaLega(nome, logo, maxAllenatori, quotaMensile, budget, primoPosto, secondoPosto, presidente)  **Post**: database.lega -> contains (lega)  **Context** GestoreLega::aggiungiSquadra(squadra, usernameAllenatore)  **Post**: Lega.allenatori -> include (Allenatore.getAllenatoreByUsername(username))  Lega.squadre -> include (squadra)  **Context** GestoreLega::aggiungiPartita(partita)  **Post**: Lega.partite -> include (partita) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | GestoreInvito |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni | **Context** GestoreInvito::inviaInvito(allenatore, nomeLega)  **Pre**:  **Context** GestoreInvito::rispondiInvito()  **Pre**: |
| Post-condizioni | **Context**  **Post**: |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | GestoreSquadra |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni | **Context** GestoreSquadra::creaSquadra(nome, allenatore, logo, nomeLega)  **Pre**: nome!=null and allenatore!=null and nomeLega!=null  **Context** GestoreSquadra::aggiungiGiocatoreSquadra(giocatore)  **Pre**: giocatore!=null  **Context** GestoreSquadra::rimuoviGiocatoreSquadra(giocatore)  **Pre**: giocatore!=null  **Context** GestoreSquadra::sostituisciGiocatoreSquadra(giocatore1, giocatore2)  **Pre**: giocatore!=null |
| Post-condizioni | **Context** GestoreSquadra::creaSquadra()  **Post**: squadra.nome=nome and squadra.allenatore=allenatore and squadra.logo=logo and squadra.nomeLega=nomeLega and squadra.punti=0 and lega.squadre -> include(squadra)  **Context** GestoreSquadra::aggiungiGiocatoreSquadra(giocatore)  **Post**: squadra.giocatori -> include(giocatore)  **Context** GestoreSquadra::rimuoviGiocatoreSquadra(giocatore)  **Post**: squadra.giocatori -> not include (giocatore)  **Context** GestoreSquadra::sostituisciGiocatoreSquadra(giocatore1, giocatore2)  **Post**: squadra.giocatori -> not include (giocatore1) and squadra.giocatori -> include (giocatore2) |
| Invariante |  |

|  |  |
| --- | --- |
| Nome classe | GestorePartita |
| Descrizione |  |
| Signature dei metodi |  |
| Pre-condizioni | **Context** GestorePartita::creaPartita(squadra1, squadra2, giornata)  **Pre**: squadra1!=null and squadra2!=null and giornata!=null  **Context** GestorePartita::setRisultato(goal1, goal2)  **Pre**: goal1!=null and goal2!=null  **Context** GestorePartita::getAllPartiteLega(lega)  **Pre**: lega!=null |
| Post-condizioni | **Context** GestorePartita::creaPartita(squadra1, squadra2, giornata)  **Post**: partita.squadra1=squadra and partita.squadra2=squadra2 and partita.giornata=giornata and database.partita -> include (partita)  **Context** GestorePartita::setRisultato(goal1, goal2)  **Post**: partita.goal1=goal1 and partita.goal2=goal2  If goal1>goal2 [squadra1.punti = @ pre squadra1.punti+3](mailto:squadra1.punti=@presquadra1.punti+3)  If goal2>goal1 [squadra2.punti = @ pre squadra2.punti+3](mailto:squadra1.punti=@presquadra1.punti+3)  If goal1=goal1 squadra1.punti = @pre squadra1.punti+1 and squadra2.punti = @pre squadra2.punti+1  **Context** GestorePartita::getAllPartiteLega(lega)  **Post**: result = getAllPartite -> select (partita.lega=lega) |
| Invariante |  |