

ODD Object Design Document

TutoratoSmart

|  |  |
| --- | --- |
| Riferimento |  |
| Versione |  |
| Data |  |
| Destinatario | Prof.ssa F. Ferrucci |
| Presentato da | Marco Delle Cave, Francesco Pagano,  Manuel Pisciotta, Alessia Olivieri |
| Approvato da |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autori |
| 11/12/19 | 0.1 | Definizione ed introduzione ODD, scelta dei trade-off | Delle Cave Marco, Pisciotta Manuel, Pagano Francesco, Olivieri Alessia |
| 14/12/19 | 0.2 | Scelta design pattern, definizione interfacce delle classi e package | Delle Cave Marco, Pisciotta Manuel, Pagano Francesco, Olivieri Alessia |
| 15/12/19 | 0.3 | Revisione generale ODD | Delle Cave Marco, Pisciotta Manuel, Pagano Francesco, Olivieri Alessia |

Sommario

[1. Introduzione 3](#_Toc27407719)

[1.1 Trade-off 3](#_Toc27407720)

[1.1.1 Comprensibilità vs costi 3](#_Toc27407721)

[1.1.2 Interfaccia vs Easy-use 3](#_Toc27407722)

[1.1.3 Manutenibilità vs efficienza 3](#_Toc27407723)

[1.2 Componenti off-the-shelf 3](#_Toc27407724)

[1.3 Linee guida per la documentazione dell’interfaccia 4](#_Toc27407725)

[1.4 Design pattern 6](#_Toc27407726)

[1.5 Definizioni, acronimi e abbreviazioni 7](#_Toc27407727)

[1.6 Riferimenti 7](#_Toc27407728)

[2 Packages 8](#_Toc27407729)

[3 Interfacce delle classi 9](#_Toc27407730)

# 1. Introduzione

## Trade-off

### 1.1.1 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.

### 1.1.2 Interfaccia vs Easy-use

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

### 1.1.3 Manutenibilità vs efficienza

Si preferisce avere una maggiore manutenibilità del sistema facendo in modo che ogni sottosistema non acceda direttamente allo storage, ma che l’accesso ai dati viene gestito da un sottosistema intermedio a discapito dell’efficienza e delle prestazioni generali

## 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.

Per il sistema che si vuole realizzare ci interessa un framework per applicazioni web e librerie per la gestione delle interfacce grafiche.

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. Inoltre bisogna commentare, giustificare delle decisioni particolari o dei calcoli.
* La convenzione che deve essere adottata da tutti i team member per quanto riguarda i nomi delle variabili, è la nota Camel Notation.

**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 della Richiesta Tutorato (compilazione richiesta, visualizzazione stato richiesta, modifica prenotazione, etc.) deve essere implementata in file separati;
* La convenzione per quanto riguarda i nomi dei file, delle operazioni e delle variabili è quella di avere nomi evocativi, ma soprattutto in lingua inglese.
* Organizzare in una cartella i file delle librerie usate e le altre risorse scaricate necessarie per lo sviluppo del progetto.

**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;

**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.

*Es.*

<html>

<head>

</head>

<body>

</body>

</html>

Deve essere sostituita da:

<html>

<head>

</head>

<body>

</body>

</html>

Questa pratica deve essere usata soprattutto per le istruzioni FOR, IF.

È buona pratica scendere di livello.

**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**

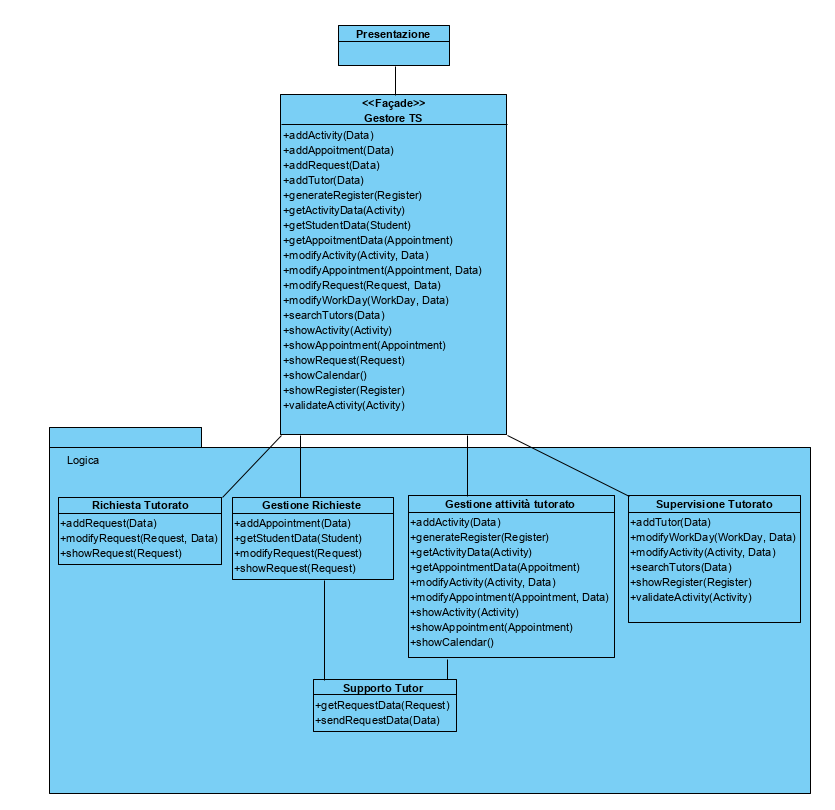
A prescindere dalle istruzioni che seguono un IF, è necessario, laddove ci fosse anche una sola istruzione, riportare il blocco di istruzioni tra parentesi graffe.

Ogni tag di apertura deve essere necessariamente seguito dall’apposito tag di chiusura (eccetto i tag self-closing).

Una convenzione importante, per quanto riguarda l’inserimento di numeri o di valori costanti, è quella di non usare una codifica fissa (hard coding) che è fortemente sconsigliata ma di associare sempre il valore ad una variabile o semplicemente definire una macro che può essere richiamata da eventi ed essere parametrizzata. In questo modo si facilita la modifica, sostituendo solo il valore della variabile o macro, in un unico posto.

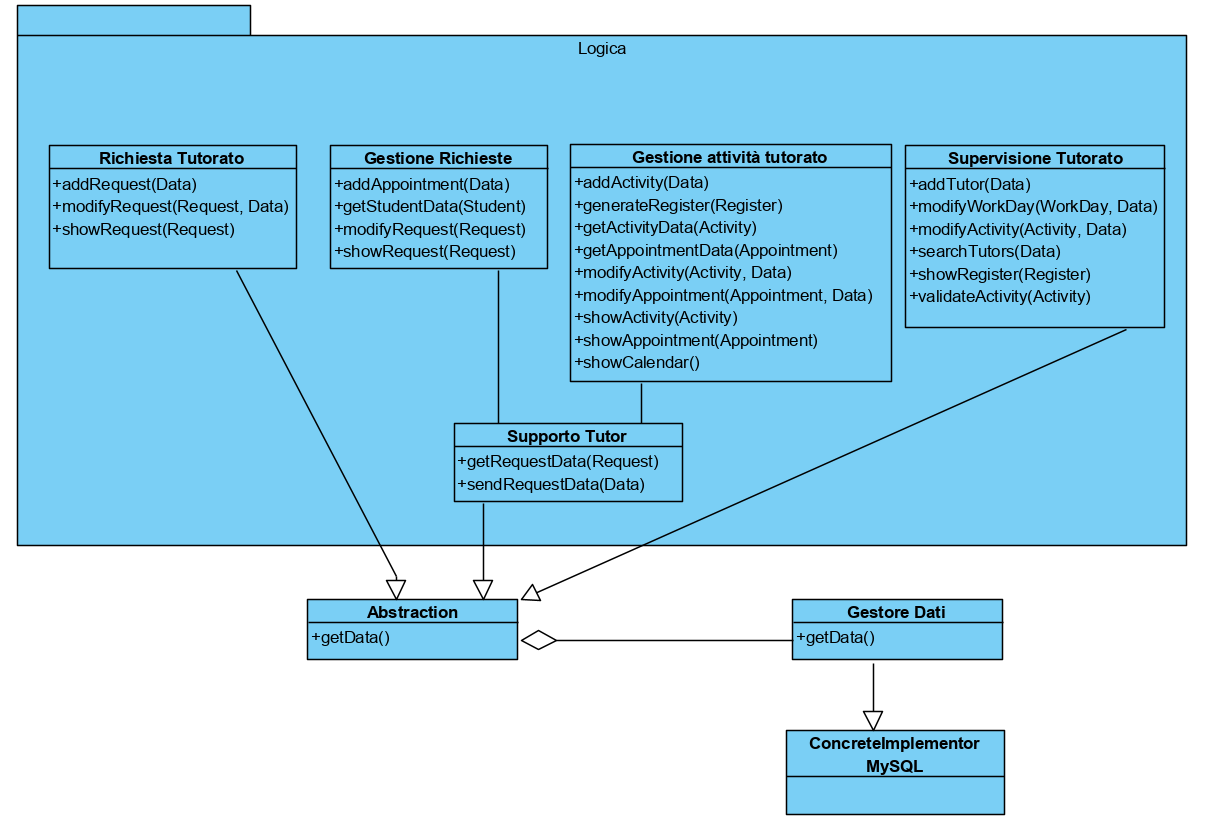
## Design pattern

**Façade Pattern**



TutoratoSmart fa uso del Façade Pattern per definire un’unica interfaccia a livello di logica che permette all’utente di interagire, attraverso l’interfaccia, con le funzionalità del sistema vedendole come un unico sistema.

**Bridge pattern**



TutoratoSmart fa uso del Bridge Design Pattern perché abbiamo bisogno di utilizzare un’unica interfaccia per diversi accessi allo storage: offrendo un’unica interfaccia si garantisce che l’eventuale cambio di implementazione del database usato comporta la modifica solo a una componente e non a svariate componenti del sistema.

## Definizioni, acronimi e abbreviazioni

**DBMS:** Data Base Management System.

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

**Framework:** Software di supporto allo sviluppo web.

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

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

**JavaScript:**  [Linguaggio di scripting](https://it.wikipedia.org/wiki/Linguaggio_di_scripting) [orientato agli oggetti](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_oggetti) e agli [eventi](https://it.wikipedia.org/wiki/Programmazione_orientata_agli_eventi), comunemente utilizzato nella [programmazione Web](https://it.wikipedia.org/wiki/Programmazione_Web) [lato client](https://it.wikipedia.org/wiki/Lato_client) per la creazione, in [siti web](https://it.wikipedia.org/wiki/Sito_web) e [applicazioni web](https://it.wikipedia.org/wiki/Applicazione_web), di effetti dinamici [interattivi](https://it.wikipedia.org/wiki/Interattivit%C3%A0) tramite [funzioni](https://it.wikipedia.org/wiki/Funzione_(informatica)) di [script](https://it.wikipedia.org/wiki/Script) invocate da *eventi* innescati a loro volta in vari modi dall'utente sulla [pagina web](https://it.wikipedia.org/wiki/Pagina_web) in uso

**Bootstrap:** Framework che contiene librerie utili per lo sviluppo responsive di pagine web.

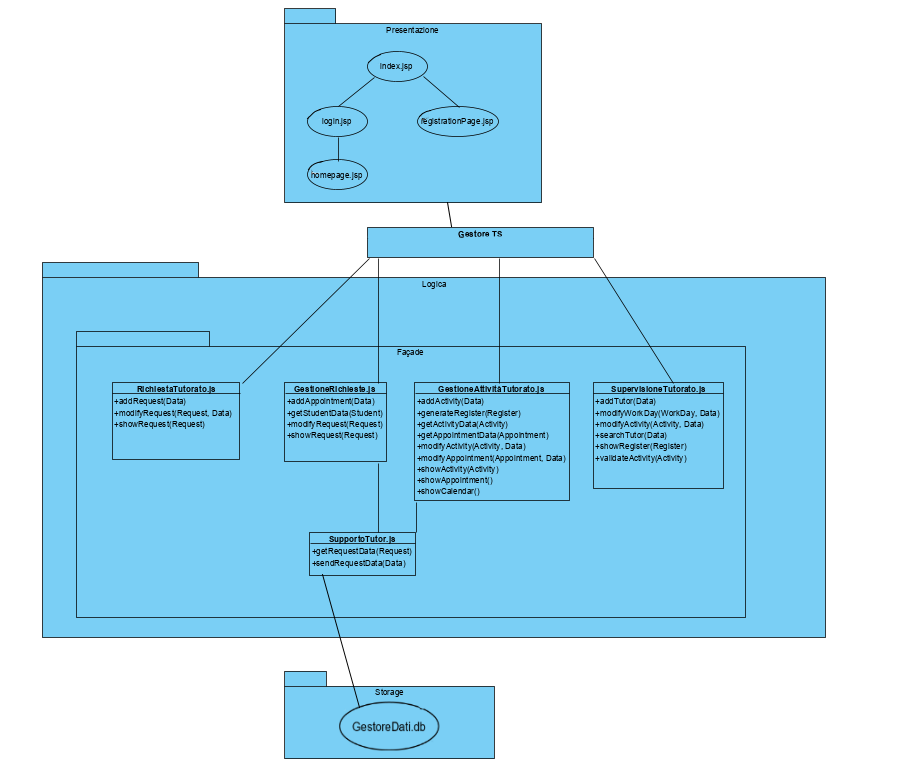
**Camel Notation:** Consiste nello scrivere più parole insieme delimitando la fine e l’inizio di una nuova parola con una lettera maiuscola.

**Hard Coding:** Codifica fissa.

## Riferimenti

* Ian Sommerville, Software Engineering, Addison Wesely
* TS\_SDD\_0.4
* <http://getbootstrap.com/>

# Packages



# Interfacce delle classi

|  |  |
| --- | --- |
| Nome classe | RichiestaTutorato |
| Descrizione | Rappresenta il gestore delle funzionalità relate alle richieste di tutorato. |
| Pre-condizione | **context** RichiestaTutorato:: addRequest(Data);  **pre:** Data!=null && isValid(Data)  **context** RichiestaTutorato:: modifyRequest(Request, Data);  **pre:** Request!=null && exists(Request) && Data!=null && isValid(Data)  **context** RichiestaTutorato:: showRequest(Request);  **pre:** Request!=null && exists(Request) |
| Post-condizione | **context** RichiestaTutorato:: addRequest(Data);  **post:** exists(Request) |
| Invarianti |  |

|  |  |
| --- | --- |
| Nome classe | GestioneRichieste |
| Descrizione | Rappresenta il gestore delle funzionalità relate alla gestione delle richieste. |
| Pre-condizione | **context** GestioneRichieste:: addAppointment(Data);  **pre:** Data!=null && isValid(Data)  **context** GestioneRichieste:: getStudentData(Student);  **pre:** Student!=null && exists(Student)  **context** GestioneRichieste:: modifyRequest(Request);  **pre:** Request!=null && exists(Request)  **context** GestioneRichieste:: showRequest(Request);  **pre:** Request!=null && exists(Request) |
| Post-condizione | **context** GestioneRichieste:: addAppointment(Data);  **post:** exists(Appointment) |
| Invarianti |  |

|  |  |
| --- | --- |
| Nome classe | GestioneAttivitàTutorato |
| Descrizione | Rappresenta il gestore delle funzionalità relate alla gestione dell’attività di tutorato. |
| Pre-condizione | **context** GestioneAttivitàTutorato:: addActivity(Data);  **pre:** Data!=null && isValid(Data)  **context** GestioneAttivitàTutorato:: generateRegister(Register);  **pre:** Register!=null && exists(Register)  **context** GestioneAttivitàTutorato:: getActivityData(Activity);  **pre:** Activity!=null && exists(Activity)  **context** GestioneAttivitàTutorato:: getAppointmentData(Appointment);  **pre:** Appointment!=null && exists(Appointment)  **context** GestioneAttivitàTutorato:: modifyActivity(Activity, Data);  **pre:** Activity!=null && exists(Activity) && Data!=null && isValid(Data)  **context** GestioneAttivitàTutorato:: modifyAppointment(Appointment, Data);  **pre:** Appointment!=null && exists(Appointment) && Data!=null && isValid(Data)  **context** GestioneAttivitàTutorato:: showActivity(Activity);  **pre:** Activity!=null && exists(Activity)  **context** GestioneAttivitàTutorato:: showAppointment(Appointment);  **pre:** Appointment!=null && exists(Appointment)  **context** GestioneAttivitàTutorato:: showCalendar(); |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| Nome classe | SupportoTutor |
| Descrizione | Rappresenta il gestore delle funzionalità relate al supporto del tutor. |
| Pre-condizione | **context** SupportoTutor:: getRequestData(Request);  **pre:** Request!=null && exists(Request)  **context** SupportoTutor:: sendRequestData(Data) ;  **pre:** Data!=null && isValid(Data) |
| Post-condizione |  |
| Invarianti |  |

|  |  |
| --- | --- |
| Nome classe | SupervisioneTutorato |
| Descrizione | Rappresenta il gestore delle funzionalità relate alla supervisione dell’attività di tutorato. |
| Pre-condizione | **context** SupervisioneTutorato:: addTutor(Data);  **pre:** Data!=null && isValid(Data)  **context** SupervisioneTutorato:: modifyWorkDay(WorkDay, Data);  **pre:** WorkDay!=null && exists(WorkDay) && Data!=null && isValid(Data)  **context** SupervisioneTutorato:: modifyActivity(Activity, Data);  **pre:** Activity!=null && exists(Activity) && Data!=null && isValid(Data)  **context** SupervisioneTutorato:: searchTutors(Data);  **pre:** Data!=null && isValid(Data)  **context** SupervisioneTutorato:: showRegister(Register);  **pre:** Register!=null && exists(Register)  **context** SupervisioneTutorato:: validateActivity(Activity);  **pre:** Activity!=null && exists(Activity) |
| Post-condizione | **context** SupervisioneTutorato:: addTutor(Data);  **post:** exists(Tutor) |
| Invarianti |  |