

ODD

Object

Design

Document

Easy Placement

ODD 1.0 – ultima modifica 15/12/2016

Destinatario del documento:

Top Manager – Prof.ssa F. Ferrucci

Presentato da:

|  |  |
| --- | --- |
| Chiara Signore | Gregorio Saggese |
| Ciro De Martino |  |

Approvato da:

Storia dei cambiamenti

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Cambiamenti | Autori |
| 27/11/2017 | 0.1 | Struttura del documento e inserimento delle sezioni relative all’Introduzione. | Tutti |
| 05/12/2017 | 0.2 | Inserimento delle sezioni relative ai Packages e alle Interfacce delle classi. | Tutti |
| 14/12/2017 | 0.3 | Inserimento della sezione relativa ai Design Pattern. | Tutti |
| 15/12/2017 | 1.0 | Revisione. | Tutti |

Sommario

[**1. Introduzione 4**](#_Toc469064002)

[**1.1 Trade-offs 4**](#_Toc469064003)

[**1.2 Componenti off-the-shelf 5**](#_Toc469064004)

[**1.3 Linee guida per la documentazione dell’interfaccia 5**](#_Toc469064005)

[**1.4 Design Pattern 7**](#_Toc469064006)

[**1.5 Definizioni, acronimi e abbreviazioni 9**](#_Toc469064007)

[**1.6 Riferimenti 11**](#_Toc469064008)

[**2. Packages 11**](#_Toc469064009)

[**3. Interfacce delle classi 12**](#_Toc469064010)

# 1. Introduzione

## **1.1 Trade-offs**

* *Comprensibilità vs costi*

La comprensibilità del codice è un aspetto molto importante, soprattutto per la fase di testing.

Ogni classe e metodo deve essere facilmente interpretabile anche dalle persone non coinvolte

nel progetto o dalle 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 processo di modifica. Ovviamente questa caratteristica aggiungerà dei

costi allo sviluppo del nostro progetto.

* *Prestazioni vs Costi*

Non avendo a disposizione alcun budget, utilizziamo materiale open source per la realizzazione

del software EP con lo scopo di rendere il sistema efficiente.

* *Costi vs Mantenimento*

L’utilizzo di materiale open source rende il sistema facilmente modificabile e manutenibile, per

questo i costi di manutenzione non saranno molto alti.

* *Interfaccia vs Easy-use*

L’interfaccia si presenta semplice ed intuitiva, permettendo una facile gestione del database

(Easy-Use).

* *Memoria vs efficienza*

Si ha una leggera perdita di velocità iniziale per recuperare le informazioni che andrà poi a favore

dell’efficienza nell’utilizzo del sistema al fine di rendere subito disponibili i dati. Quindi

preferiamo le prestazioni invece della memoria.

* *Sicurezza vs efficienza*

Nel sistema ogni richiesta del client viene validata controllando la validità di username e

password. Questa caratteristica potrebbe far aumentare il tempo di risposta del sistema

soprattutto in situazioni di carico elevato, ma è necessario effettuare tali controlli per rispettare

i requisiti iniziali del sistema.

* *Tempo di risposta vs Spazio di memoria*

La scelta di utilizzare un database relazionale è scaturita dai diversi vantaggi che ne derivano:

- gestione consistente dei dati;

- tempo di risposta basso rispetto all’utilizzo del file system;

- accesso veloce e concorrente ai dati;

Lo svantaggio nell’utilizzo di un database relazionale è che richiede il triplo dello spazio di

memoria rispetto ad un file system.

## **1.2 Componenti off-the-shelf**

Per il progetto facciamo uso di componenti off-the-shelf, ovvero componenti software disponibili sul mercato per facilitare la creazione del progetto. Per il sistema che si vuole realizzare, siamo interessati ad un framework per applicazioni web. Il framework che andremo ad utilizzare è Bootstrap, un framework open source che contiene una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Questo framework 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. Bootstrap è compatibile con le ultime versioni di tutti i principali browser. Dalla versione 2.0 supporta anche il responsive web design. Ciò significa che il layout delle pagine web si regola dinamicamente, tenendo conto delle caratteristiche del dispositivo utilizzato, sia esso desktop, tablet o telefono cellulare.

## **1.3 Linee guida per la documentazione dell’interfaccia**

Prima dell’implementazione della logica è opportuno sottometterla al Project Manager, l’algoritmo che si intende seguire per l’implementazione, in modo che eventuali correzioni della logica da seguire possano essere apportate prima di imbattersi nella sintassi degli strumenti scelti. 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 inserire un underscore per separare più parole che compongono un unico nome.

**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 EP (login, Questionario, Convalida Documento ecc.) deve essere implementata in file separati;
* Diviso in più file, se raggiunge una lunghezza tale da divenire difficile da leggere e comprendere.

Se sono presenti diversi script per le singole funzionalità è opportuno organizzarli in una apposita cartella. La convenzione per quanto riguarda i nomi dei file, delle operazioni e delle variabili è quella di avere nomi evocativi, ma soprattutto in lingua italiana. Le uniche eccezioni sono legate al linguaggio di programmazione utilizzato. 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;
* 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.

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.

**Inizializzazione**

Inizializzare le variabili locali nel punto in cui sono state dichiarate a meno che il loro 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**

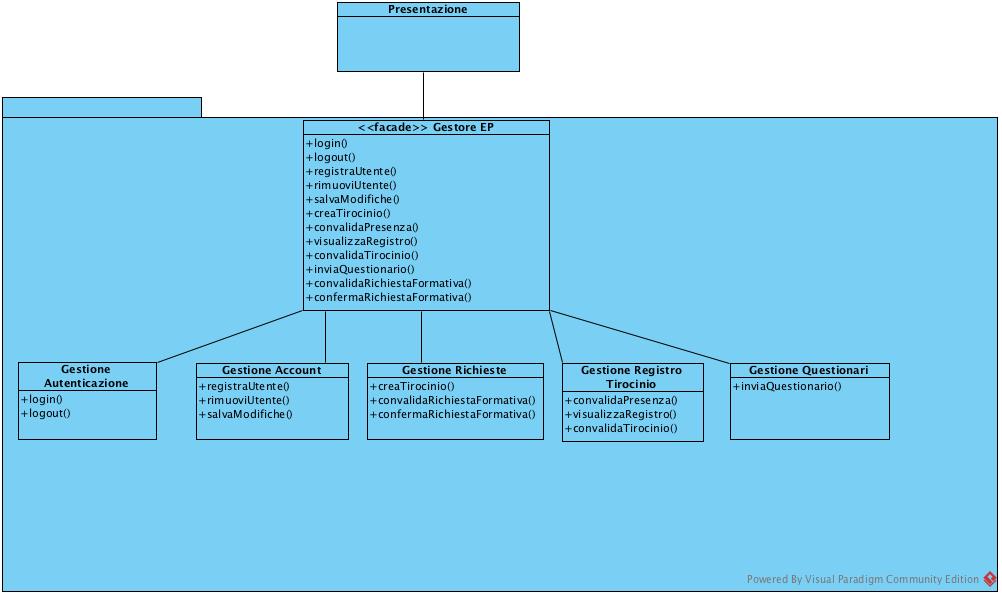
A prescindere dalle istruzioni che seguono un IF, è necessario, laddove ci fosse anche una sola istruzione, riportare il blocco di istruzioni tra parentesi graffe.

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

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.

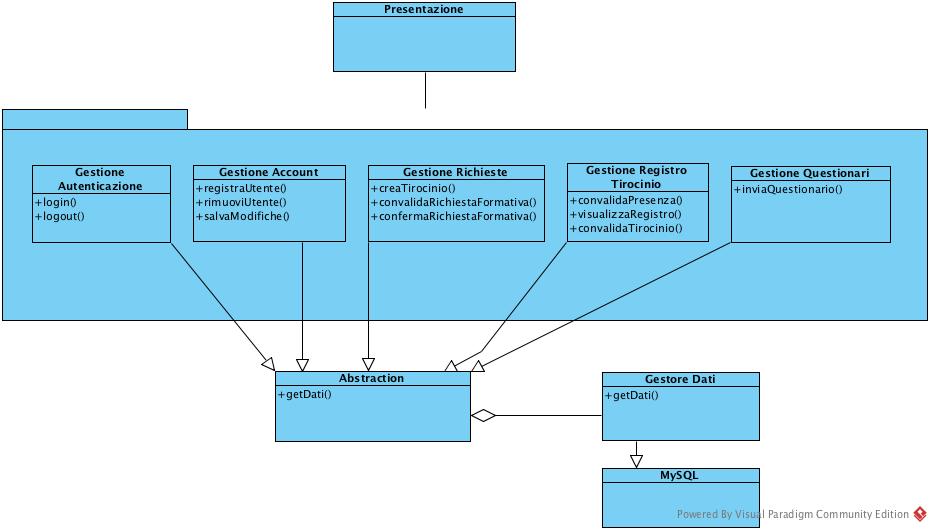
## **1.4 Design Pattern**

## **Façade Pattern**

****

EP 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**

****

EP fa uso del Bridge 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.

## 1.4 Definizioni, Acronimi e Abbreviazioni

***Definizioni***

**Azienda:** Organismo composto di persone e beni, diretto al raggiungimento di un fine economico, d’interesse sia al pubblico sia privato.

**Parametri Azienda:** rappresentano le informazioni relative ad ogni azienda:

1. **Nome**;
2. **Tutor Aziendale**;
3. **Responsabile Aziendale**;
4. **Sede legale:** È il luogo in cui dall’atto costitutivo essa risulta avere il centro dei propri affari;
5. **Sede operativa:** È il luogo in cui si trovano gli uffici dell’azienda, nella quale vengono svolte attività politiche e gestionali;
6. **Ambito:** Il settore per cui si distingue l’azienda.

**Studente:** Persona iscritta ad un corso di studi o ad una facoltà universitaria.

**Parametri Studente:** rappresentano le informazioni relative ad ogni studente:

1. **Nome;**
2. **Cognome;**
3. **Email Istituzionale:** è un indirizzo email che come dominio riporta quello universitario;
4. **Username;**
5. **Numero di telefono;**

**Ufficio Stage e Tirocini:** Ufficio universitario che si occupa della gestione amministrativa per quanto riguarda la stipula di convenzioni con Aziende/Enti pubblici e progetti formativi.

**Parametri Ufficio Stage e Tirocini:** rappresentano le informazioni relative all’ufficio:

1. **Nome;**
2. **Cognome;**

**Tutor Aziendale:** Persona che si occupa di tenere traccia delle presenze dello studente durante il tirocinio.

**Parametri Tutor Aziendale:** rappresentano le informazioni relative al Tutor aziendale:

1. **Nome;**
2. **Cognome;**

**Presidente Consiglio Didattico:** Persona che si occupa di presiedere il consiglio didattico e che ha il compito di convalidare il tirocinio effettuato dallo studente;

**Parametri Presidente Consiglio Didattico:** rappresentano le informazioni relative al Presidente del Consiglio Didattico:

1. Nome;
2. Cognome;

**Tutor Accademico:** Persona che si occupa delle esigenze del tirocinio dello studente.

**Parametri Tutor Accademico:** rappresentano le informazioni relative al Tutor Accademico:

1. Nome;
2. Cognome;

**Progetto Formativo:** Documento contenente tutti i dettagli e i ruoli del tirocinio dello studente.

**Parametri Progetto Formativo:** rappresentano le informazioni relative al Progetto Formativo:

1. Data Stipula;
2. Contratto: Accordo contenente tutte le informazioni relative al tirocinio che lo studente dovrà svolgere.
3. Studente;
4. Tutor Aziendale;
5. Tutor Accademico;
6. Presidente Consiglio Didattico;

**Tirocinio:** addestramento pratico dello studente in un ambito lavorativo.

**Parametri Tirocinio:** rappresentano le informazioni relative del Tirocinio:

1. Studente;
2. Azienda;
3. Luogo di Svolgimento;

***Acronimi***

EP = Easy Placement

RAD = Requirement Analysis Document.

UC = Use Case.

UCD = Use Case Diagram.

RF = Requisito Funzionale.

RNF = Requisito Non Funzionale

AUT = Autenticazione.

ACC = Account.

RIC = Richiesta

RT = Registro Tirocini

QUEST = Questionario

CD = Class Diagram

SD = Sequence Diagram

SCD = Statechart Diagram

MP = MockUp

RAD = Requirement Analysis Document.

SDD = System Design Document.

ODD = Object Design Document.

MVC = Model View Controller.

HTML = HyperText Markup Language.

CSS = Cascading Style Sheets.

## **1.6 Riferimenti**

* corsi.unisa.it/informatica/attivita-e-servizi/tirocini
* Bernd Bruegge & Allen H. Dutoit, Object-Oriented Software Engineering: Using UML, Patterns and Java, (2nd edition), Prentice-Hall, 2004.
* Ian Sommerville, Ingegneria del software, (8a edizione), Pearson, 2007.
* EP\_RAD\_Vers3.0
* EP\_SDD\_Vers1.0

2. Packages

