



Università degli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate

Laboratorio Interdisciplinare B

Documentazione di Progetto

Loris Bozzato

Dipartimento di Scienze Teoriche e Applicate

loris.bozzato@uninsubria.it



Documentazione di Progetto

1. Manuale Utente
2. Manuale Tecnico
3. JavaDoc

- **Manuale Utente**
 - Manuale di alto livello del funzionamento del programma
 - Come usare il programma (quale ambiente usare, come installare l'ambiente, come installare il programma, come eseguirlo, ...)
 - Aggiungere screenshot per semplificare la comprensione del funzionamento / uso del programma
 - Utilizzare un livello di astrazione e un linguaggio adeguati ad un pubblico non esperto e non tecnico
- Seguendo le specifiche del progetto, distinguere le sezioni delle funzioni per utenti registrati e non registrati

- Manuale Utente – Struttura
 - Frontespizio (Titolo, Autori, Data e Versione documento)
 - Indice
 - Installazione
 - Requisiti di sistema
 - Setup ambiente
 - Installazione programma
 - Esecuzione ed uso
 - Setup e lancio del programma
 - Uso delle funzionalità
 - Data set di test
 - Limiti della soluzione sviluppata
 - Sitografia / Bibliografia
 - es. QtSpim Simulator, Online: www.xyz.com
 - es. D. Tosi, "Come fare un progetto software", nome rivista, anno

- **Manuale Tecnico**
 - dettagliare dal punto di vista tecnico la struttura dell'applicazione
 - **le scelte progettuali: progettazione SW e DB, con diagrammi UML e ER**
 - le scelte architettureali
 - le strutture dati utilizzate
 - le scelte algoritmiche
 - il formato dei file e la loro gestione (es. path)
 - l'uso eventuale di pattern, riportando parti di codice significativo
 - utilizzare un livello di astrazione e un linguaggio adeguati a esperti tecnici del dominio
- la documentazione prodotta con JavaDoc è considerata parte del manuale tecnico (il codice deve quindi essere commentato opportunamente)

- Manuale Tecnico - Struttura
 - Si segua la struttura del Manuale Utente, quindi:
 - Frontespizio (Titolo, Autori, Data e Versione documento)
 - Indice
 - **Progettazione della soluzione**
(progettazione software + progettazione DB)
 - Report tecnico della soluzione sviluppata
(scelte di architettura, strutture dati, algoritmi utilizzati)
 - Limiti della soluzione sviluppata
 - Sitografia / Bibliografia

Documentazione di Progetto

Documentazione del sistema client/server con UML:

- UML deve far vedere la **struttura statica** del sistema ad un livello di astrazione maggiore del codice sorgente (**Class Diagram**).
- UML deve far vedere la **struttura dinamica** del sistema, catturando le principali interazioni fra classi attraverso **Sequence Diagrams, Interaction Diagram e State diagrams**.
- **NB: non è necessario usare tutti questi diagrammi**, a patto che l'insieme prodotto sia sufficiente a rappresentare chiaramente il comportamento del sistema (sia dal punto di vista strutturale che comportamentale)

Documentazione del database:

Documento di analisi dei requisiti ristrutturato e documentazione associata allo schema ER (ristrutturato e non), con eventuale specifica di vincoli in linguaggio naturale, schema finale tabelle, query di creazione e interrogazione

- **JavaDoc** ¹

- è uno strumento che permette di documentare i file sorgente di un programma usando le informazioni riportate al loro interno
- il programmatore inserisce i commenti al codice in un formato predefinito
- i commenti vengono riconosciuti dal programma javadoc che li converte in un formato di consultazione (html, pdf, ...)
- un commento Javadoc è un testo HTML racchiuso tra i tag `/**` e `*/`
- Javadoc è pensato solo per descrivere le funzionalità di Package, Classi, interfacce, Metodi (non può essere usato per commentare il codice)

¹ <https://www.oracle.com/technical-resources/articles/java/javadoc-tool.html>

Documentazione di Progetto

Esempio Javadoc

```
/**  
 * Questo &egrave; un commento Javadoc.  
 * Gli spazi e gli asterischi a inizio riga  
 * sono sempre ignorati.  
 */
```

L'effetto è il seguente:

Questo è un commento *Javadoc*. Gli spazi e gli asterischi a inizio riga **sono** sempre ignorati.

Documentazione di Progetto

- Formato generale di un tag: **@name comment**
dove nome specifica quale tipo di informazione si sta dando e
il commento è l'informazione.
Esempio: **@author William Shakespeare**
- Ogni tag deve essere su una riga nuova.
- I commenti possono estendersi su più righe, ma non ci devono essere righe vuote

- **JavaDoc (cont.)**
 - la documentazione deve quindi comprendere la descrizione di ogni package, classe, interfaccia, metodo e attributo
 - Il commento è sempre posto subito prima della dichiarazione della classe, interfaccia, metodo, attributo
 - può contenere tag HTML per aiutare la formattazione (es. tag di struttura come ``, ``, ecc)

- **JavaDoc - Classi**
 - deve descrivere lo scopo della classe
 - al termine della descrizione si mette il tag `@author Nome Cognome` per indicare l'autore del codice
 - se ci sono più autori, si mettono più tag su righe separate, uno di seguito all'altro

Tag minimo per le classi

```
/** ...  
 *  
 * @author William Shakespeare  
 * @author Christopher "Kit" Marlowe  
 */  
public class Drama {...}
```

- **JavaDoc – Attributi**

- deve descrivere a cosa serve l'attributo
- solo il primo paragrafo verrà riportato nella descrizione generale. Gli altri andranno nella descrizione puntuale
- un nuovo paragrafo inizia con il tag `<p>` a inizio linea
- usare i tag `<code>` e `</code>` per racchiudere il nome degli attributi
 - vengono visualizzati nel font usato per il codice

- **JavaDoc – Metodi**

- per ogni metodo usare sempre i tag a inizio linea:
 - `@param <nome parametro>` per fornire una breve descrizione del parametro (se più parametri, rispettare l'ordine con cui sono dichiarati)
 - `@return` per fornire una breve descrizione di ciò che il metodo ritorna
 - `@throws <nome eccezione>` per fornire una descrizione delle circostanze che determinano il sollevamento dell'eccezione

- **JavaDoc – Package**
 - deve riportare la documentazione generale del package
 - (il commento JavaDoc va inserito in `package-info.java` nel package)
 - usare i tag `@see` per riferirsi alle classi, metodi, attributi in esso contenuti
- Potete fare riferimento alla documentazione di JavaDoc per l'uso di tag aggiuntivi (e.g. `@version`, `@since`, `@deprecated`, `@serialData...`)
- Per generare la documentazione si usa il comando `javadoc` che fa parte della distribuzione di Java Development Kit (JDK)

Aspetti Organizzativi del Progetto

- Gli aspetti organizzativi di un progetto sono relativi a:
 - Ruoli delle persone
 - Struttura del progetto e project management

Aspetti Organizzativi del Progetto: Gantt



Aspetti Metodologici del Progetto

Cosa occorre fare:

1. Individuare i ruoli e assegnarli nel gruppo
(vedi slide ruoli)
2. Decidere i tempi di esecuzione delle fasi del progetto
(vedi slide precedente)
3. **Progettare la Soluzione Software**
(quali componenti, come strutturare il DB, quali patterns, come gestire architettura client / server e concorrenza)
4. Sviluppare la Soluzione Software
(implementare il codice/testare il codice)
5. Documentare il progetto e controllo qualità