

*Università degli studi di Salerno*  
*Dipartimento di Informatica*  
*Corso di Laurea in Informatica*

# Object Design Document

## *“TomMASO”*

### **Docente:**

Andrea De Lucia

### **Tutor:**

Manuel De Stefano

### **Studenti:**

<i>Nome</i>	<i>Matricola</i>
Antonio Allocca	0512106933
Pasquale Di Costanzo	0512106675
Vincenzo Esposito	0512108127
Antonio Toppi	0512106882

*Anno Accademico: 2021/22*

## Revision History

Data	Versione	Descrizione
18/11/2021	0.1	Prima stesura
20/11/2021	0.2	Aggiunti i pacchetti
22/11/2021	0.3	Aggiunte Scelte Progettuali
20/12/2021	0.4	Revisione finale

# Indice

1. Introduzione.....	4
1.1. Trade-offs .....	4
1.2. Interface documentation guidelines .....	4
1.3. Definizioni, acronimi e abbreviazioni .....	5
1.4. Riferimenti .....	5
2. Pacchetti .....	6
3. Scelte progettuali.....	7
4. Specifiche delle interfacce.....	8

# 1. Introduzione

## 1.1. Trade-offs

<b>Funzionalità vs Usabilità</b>	Si è data maggiore importanza all'usabilità del sistema, in quanto un numero troppo elevato di funzionalità può scoraggiare l'utilizzo da parte di utenti con scarse competenze digitali
<b>Costi vs Portabilità</b>	Siccome il sistema potrà essere installato su server eterogenei, è stato previsto l'utilizzo di tecnologie che potessero garantire la portabilità, nonostante comportassero un incremento nei costi
<b>Sicurezza vs Efficienza</b>	Si è data maggior importanza alla sicurezza del sistema e sono stati previsti vari controlli per ogni funzionalità, a discapito delle performance
<b>Comprensibilità vs Tempo</b>	Nonostante la quantità di tempo richiesta, si è scelto di rendere quanto più leggibile il codice tramite un'attenta documentazione

## 1.2. Interface documentation guidelines

- Le classi hanno nomi al singolare (es. class Blog)
- I nomi delle classi cominciano con lettera maiuscola
- I metodi hanno nomi formati da espressioni verbali con campi e parametri a cui fanno riferimento (es. findBlogById)
- Gli errori vengono mostrati all'utente con pagine apposite e tramite gli Status Codes di HTTP
- Per i nomi di variabili e metodi viene utilizzato il camelCase
- Per i nomi delle costanti vengono usate stringhe in maiuscolo
- I nomi dei pacchetti sono stringhe in minuscolo
- I nomi dei metodi d'accesso alle variabili sono del tipo "getNomeVariabile"
- I nomi dei metodi di modifica alle variabili sono del tipo "setNomeVariabile"

- I nomi delle pagine JSP devono essere in camelCase e devono descrivere l'interfaccia che offrono all'utente
- Per poter distinguere le servlet dalle classi che implementano logica applicativa, esse sono raggruppate in sottopacchetti

### 1.3. Definizioni, acronimi e abbreviazioni

**ODD** - Object Design Document

**RAD** – Requirements Analysis Document

**SDD** – System Design Document

**API** – Application Programming Interface

**URL rewriting** – Tecnica con la quale si appone un token di sessione all'URL in modo tale che il server possa recuperarlo senza bisogno di cookie












**HTML** – HyperText Markup Language

### 1.4. Riferimenti

RAD di TomMASO

SSD di TomMASO

## 2. Pacchetti

- >  access
- >  api
- ▼  blog
  - >  management
  - >  visualization
- >  chat
- >  general
- >  report
- ▼  storage
  - >  model
- >  utility

Il pacchetto **access** contiene gli elementi necessari al funzionamento del sistema di login e di registrazione

Il pacchetto **api** contiene le classi necessarie al funzionamento delle API

Il pacchetto **blog** contiene le classi che si occupano della logica applicativa e della visualizzazione dei blog nei rispettivi sotto-pacchetti **management** e **visualization**

Il pacchetto **chat** contiene le classi per la gestione delle chat

Il pacchetto **general** contiene le classi che gestiscono le pagine base del sito

Il pacchetto **report** contiene le classi per la gestione dei report

Il pacchetto **storage** contiene le classi che si interfacciano con il database e nel sotto-pacchetto **model** le classi che rappresentano le entità del sistema

Il pacchetto **utility** contiene classi con metodi utili per il sistema

### 3. Scelte progettuali

- 3.1. La sessione è rappresentata da una classe realizzata ad hoc che contiene tutti gli oggetti della sessione per evitare continui casting ai valori restituiti dal metodo *request.getSession().getAttribute(x)*, che potrebbero rendere il codice poco leggibile
- 3.2. Per poter aver a disposizione la nostra sessione, ogni richiesta verrà rappresentata con una classe creata ad hoc basata su *HttpServlet*, ma che contenga la sessione
- 3.3. La sessione dell'utente memorizza i blog visitati per assicurare che la classifica dei blog più visitati sia accurata, e non influenzata da continue visite da parte dello stesso utente
- 3.4. Poiché per alcune funzionalità del sistema non erano necessarie servlet specifiche (inviano dati al server e non forniscono un'interfaccia), si è scelto di implementare alcune funzionalità come API con risultato in JSON, ciò rende possibile anche la futura pubblicazione di una lista di API pubbliche per permettere anche a client esterni di fare richieste al sistema, oltre a ridurre i dati che il server deve trasferire al client
- 3.5. Per permettere anche agli utenti con cookie disabilitati di mantenere una sessione, il sistema prevede l'utilizzo dell'URL rewriting
- 3.6. Per motivi di sicurezza in ogni reindirizzamento ad una JSP si passa prima per una servlet dove vengono effettuati i controlli necessari
- 3.7. Per semplificare il processo di debugging delle query è stato progettato un proxy per i *PreparedStatement*
- 3.8. Per evitare che il sistema crei chat diverse per gli stessi utenti i membri vengono ordinati per id ( $utente1.id < utente2.id$ )
- 3.9. Il sistema di ranking dei blogs si basa sul numero di visite ricevute dal blog
- 3.10. Per ridurre al minimo i dati sensibili dell'utente vengono memorizzati unicamente username e password
- 3.11. Il server mantiene la versione Markdown degli articoli, la quale viene inviata al client che provvederà a convertire in HTML e a sanificarne il contenuto, permettendo così di alleggerire il carico del server oltre che a garantire maggiore sicurezza
- 3.12. Per fornire un'interfaccia più accattivante si usano icone personalizzate per utenti e blog si utilizzano icone generate automaticamente dalla libreria *jdenticon*

## 4. Specifiche delle interfacce

### Il sottosistema ChatSystem

- List<Messaggio> fetchMessages(chat, amount, offset) throws SQLException
- Messaggio sendTextToChat(chat, mittente, testo) throws SQLException
- Chat createChat(user1, user2) throws SQLException
- Chat findChatById(idChat) throws SQLException
- Messaggio findMessageById(idMessaggio) throws SQLException
- List<Chat> findUserChats(user) throws SQLException
- Chat findChatByUsers(u1, u2) throws SQLException
- List<Messaggio> fetchMessageFromId(chat, fromId) throws SQLException
- void deleteMessage(messaggio) throws SQLException

### Il sottosistema StorageSystem

- Blog fromPathInfo(pathInfo) throws SQLException
- void writeFile(input, output) throws IOException
- void recursiveDelete(file)
- String relativeUrl(file)
- String escapeForMarked(string)
- String headFile(file, numRows) throws IOException
- List<File> getPages(context, log)
- FileType getFileType(context, file)
- File blogPathToFile(pathInfo)
- String escapeMDFile(file) throws IOException
- void init() throws SQLException
- void closeConnection() throws SQLException
- void close()
- Generic findById(entity, id) throws SQLException
- List<Generico> resultSetToList(entity, rs) throws SQLException

### Il sottosistema PermissionManageSystem

- void changePermissions(utente, permessi) throws SQLException



## Il sottosistema AccessSystem

- String crypt(pwd)
- boolean verify(cryped, pwd)
- Session loadSession(req) throws SQLException
- Session createSession()
- void updateUser() throws SQLException
- void visitedBlog(blog) throws SQLException
- boolean isLogged()
- Utente getUtente()
- Session setUtente(utente)
- Utente findUserById(idUtente) throws SQLException
- Utente registerUser(password, username) throws SQLException

## Il sottosistema BlogContentManagement

- Blog createBlog(utente, nome)
- void deleteBlog(blog)
- void uploadMdFileOnServlet(session,file,path,resp)
- boolean uploadFileOnServlet(part, url)

## Il sottosistema BlogContentVisualization

- Blog findBlogById(idBlog)
- void incrementVisit(blog)
- List<Blog> getBlogsUser(u)
- List<Blog> topBlogs(count)

## Il sottosistema ReportSystem

- Report report(comment, user, url, reason, target) throws SQLException
- Report findReportById(idReport)
- List<Report> fetchUnreviewedOfType(type) throws SQLException
- void reviewReport(report, approved) throws SQLException

## Il sottosistema SearchSystem

- Blog findBlogByName(name)
- Utente findUserByUsername(username) throws SQLException