|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Requirement Analisys Document  Progetto  STORYTELLING   |  |  | | --- | --- | | Riferimento |  | | Versione | 0.8 | | Data | 10/12/2021 | | Destinatario | Prof. Gravino | | Presentato da | Alessandro Marigliano, Antonio Scotellaro, Emmanuele Virginio Coppola, Muriel Rossi. | | Approvato da |  | |

# RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 19/12/2021 | 0.1 | Prima stesura | Emmanuele Virginio Coppola |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Team Members

|  |  |  |
| --- | --- | --- |
| **Nome** | **Acronimo** | **Informazioni di contatto** |
| Emmanuele Virginio Coppola | EVC | e.coppola37@studenti.unisa.it |
| Alessandro Marigliano | AM |  |
| Antonio Scotellaro | AS |  |
| Muriel Rossi | MR |  |

Sommario

[1.1.Obiettivo Sistema 4](#__RefHeading___Toc3718_787960820)

[1.2. Ambito del Sistema 4](#__RefHeading___Toc3720_787960820)

[1.3. Obiettivi e criteri di successo del Sistema 4](#__RefHeading___Toc3722_787960820)

[1.4 Definizioni, Acronimi e Abbreviazioni 5](#__RefHeading___Toc3735_3709428043)

[3 Sistema Proposto 6](#__RefHeading___Toc3727_3709428043)

[3.2 Requisiti Funzionali 7](#__RefHeading___Toc2980_1234530958)

[3.2.1 Gestione Account 7](#__RefHeading___Toc2982_1234530958)

[3.2.2 Gestione bacheca 7](#__RefHeading___Toc3557_2010560777)

[3.3 Requisiti Non Funzionali 9](#__RefHeading___Toc3559_2010560777)

[3.3.1 RNF\_1: Usabilità 9](#__RefHeading___Toc3561_2010560777)

[3.3.2 RNF\_2: Affidabilità 9](#__RefHeading___Toc3563_2010560777)

[3.3.3 RNF\_3: Legali 9](#__RefHeading___Toc3565_2010560777)

[3.4.1 Scenari 10](#__RefHeading___Toc3567_2010560777)

[GESTIONE ACCOUNT 11](#__RefHeading___Toc3307_2763684958)

[GESTIONE BACHECA 15](#__RefHeading___Toc3729_3709428043)

[3.1 Casi d’Uso/Use Case 19](#__RefHeading___Toc3569_2010560777)

[3.4.2 Modelli Caso D’uso 19](#__RefHeading___Toc3309_2763684958)

[3.3. Modello ad Oggetti 36](#__RefHeading___Toc3609_2010560777)

[3.3.1. *Tabella riassuntiva di tutti gli oggetti* 36](#__RefHeading___Toc3611_2010560777)

[3.3.2 Object Diagram 38](#__RefHeading___Toc5511_2533412105)

[3.3.3 Class Diagram 40](#__RefHeading___Toc3311_2763684958)

[3.3.4 Modello Dinamico 41](#__RefHeading___Toc3311_27636849581)

[3.3.4.1 Activity DIagram 41](#__RefHeading___Toc5513_2533412105)

[3.3.4.2 Sequence Diagram 44](#__RefHeading___Toc5515_2533412105)

[3.3.4.3 State Chart Diagram 49](#__RefHeading___Toc4224_1458941559)

[3.4.4 Navigational path 50](#__RefHeading___Toc3731_3709428043)

[3.4.5 Esempi di Interfacce (Mock-up) 51](#__RefHeading___Toc3724_787960820)

# 1 Introduzione

Storytelling si propone di creare una piattaforma libera di pubblicità e …

In questa prima sezione del documento verranno descritti i trade-offs e le linee guida per l’implementazione , riguardante la nomenclatura, la documentazione e le convenzioni sui formati.

## 1.1 Object design goals

**Modularità:**

Il sistema Storytelling deve avere la possibilità di poter cambiare e aggiungere funzionalità senza dover modificare molto codice.

**Robustezza:**

Il Sistema deve risultare robusto, reagendo in maniera adeguata alle situazioni impreviste grazie alla gestione delle eccezioni evitando di bloccare le componenti non coinvolte del sistema.

**Riusabilità:**

Il Sistema deve basarsi sulla riusabilità del codice attraverso l’utilizzo delle interfacce e dei design pattern

## 1.2 Object Trade-Off

**Modularità vs Tempi di esecuzione:**

Il sistema sarà costruito in maniera tale da preferire ove possibile la modularità rispetto al tempo di Esecuzione.

**Robustezza vs Tempo di esecuzione:**

Il sistema è costruito per preferire effettuare i controlli necessari per rendere il sistema più robusto anche se c’è un perdita in tempi di esecuzione.

## 1.3 Linee guida per la documentazione dell’interfaccia

Le linee guida sono una serie di regolamenti usati dagli sviluppatori per la progettazione delle interfacce per facilitarne la comprensione anche per altri team di sviluppatori. Le linee guida sono definite in base alla convenzione Java **Sun Java Coding Conventions** [Sun, 2009].

Link a documentazione ufficiale sulle convenzioni

Di seguito una lista di link alle convenzioni usate per definire le linee guida:

**• Java Sun:** <https://checkstyle.sourceforge.io/sun_style.html>  
**• HTML:**<https://www.w3schools.com/html/html5_syntax.asp>

## 1.4 Definizioni, acronimi e abbrevizioni

Vengono riportati di seguito alcune definizioni presenti nel documento:

* **Package:** raggruppamento di classi, interfacce o file correlati;
* **Design pattern:** template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità;
* **Interfaccia:** insieme di signature delle operazioni offerte dalla classe;
* **View:** nel pattern MVC rappresenta ciò che viene visualizzato a schermo da un utente e che gli  
  permette di interagire con le funzionalità offerte dalla piattaforma;
* **lowerCamelCase:** è la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione nel  
  mezzo della frase inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi;
* **UpperCamelCase:** è la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione inizi  
  con una lettera maiuscola, senza spazi o punteggiatura intermedi;
* **Javadoc:** sistema di documentazione offerto da Java, che viene generato sottoforma di interfaccia  
  in modo da rendere la documentazione accessibile e facilmente leggibile
* **IDE(Integated develoment enviroment):** software che sfrutta l’implementazione di tecnologie diverse per velocizzare implementazione e testing del codice.
* **POJO(Plain Old Java Object):** Oggetti java semplici che non sono legati a restrizioni speciali aumentando quindi leggibilità e riutilizzo del codice.

## 1.5 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:  
• Statement Of Work;  
• Business Case;  
• Requirements Analysis Document;  
• System Design Document;  
• Object Design Document;  
• Test Plan;  
• Matrice di tracciabilità;  
• Manuale di installazione;  
• Manuale utente;

Per definizioni di design pattern applicati in ambienti java:

<https://www.oracle.com/java/technologies>

# 2 Packages

In questa sezione viene mostrata la suddivisione del sistema in package, in base a quanto definito nel documento di System Design.

Tale suddivisione è motivata dalle scelte architetturali prese e ricalca la struttura di directory standard definita da Maven e di quella dei progetti generati dall’IDE **Intellij IDEA .**

# 3 Interfacce delle classi

# 4 Design Patterns

Nella presente sezione si andranno a descrivere e dettagliare i design patterns utilizzati nello sviluppo dell’applicativo Storytelling.

Per ogni pattern si darà:  
• Una brevissima introduzione teorica.  
• Il problema che doveva risolvere all’interno di BiblioNet.  
• Una brevissima spiegazione di come si è risolto il problema in BiblioNet.  
• Un grafico della struttura delle classi che implementano il pattern.

**Adapter**

L’Adapter è un design pattern strutturale , ovvero una soluzione già provata nell’industria informatica che semplifica le relazioni tra le entità. In questo caso l’Adapter è un pattern che facilita la comunicazione tra il sistema Storytelling e il modulo fia.

Infatti il problema riguarda la comunicazione con il suddetto modulo fia che communica con il sistema con messaggi HTTP, rendendo quindi necessaria la creazione di una classe che permettesse di convertire i messaggi inviati dal modulo fia in oggetti java utilizzabili dal sistema e viceversa.

**DAO**

Un DAO (Data Access Object) è un pattern che offre un’interfaccia astratta nascondendo i dettagli implementativi del database usato dagli utilizzatori.

Il problema è la quantità di dati da gestire e la modularità del sistema e quindi permettere di poter cambiare facilmente il database implementato nel sistema non cambiando l’interfaccia mostrato ai sottosistemi utilizzatori del database rendendo quindi la gestione dei dati semplice e anche facilmente modificabile.

I DAO si occuperanno di salvare e leggere i messaggi all’interno del database rendendo maneggevoli le entità Utenti,Commenti, Reazioni, Storie implementandoli con dei POJO che possono essere generati automaticamente durante l’implementazione grazie all’implementazione standard dell’IDE **intellij Idea.**

**Prototype?????**

Il Prototype è un pattern crazionale, ovvero una soluzione già provata nell’industria informatica che semplifica la creazione di oggetti, in questo caso creare un clone dell’oggetto demandano la creazione direttamente all’oggetto stesso che vogliamo creare permettendo quindi di ignorare l’implementazione della logica dell’oggetto abbassando quindi diminuendo l’accoppianmento.

Il problema posto nel sistema è l’accesso ad oggetti che vengono richiamati spesso come le storie pubblicate più recentemente che essendo visualizzate nella pagina principale del sito sono di certo quelle con più accessi e di cui abbiamo bisogno di copie locali.

???????????????????????????????????????????????????

# 5 Glossario