

15/04/2013

Gruppo Azzurro Mariarosaria Acampora

Domenico Alampi

Giuseppe Gravina

Roberta Orlando

Matteo Sabatini

**Documento Baseline v 1.0**

|  |  |
| --- | --- |
| ProjectWork2013 | **Sistema Editoriale** |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 15/04/2013 | 1.0 | Baseline 1 | Gruppo Azzurro |
| 22/04/2013 | 1.1 | Baseline 2 | Gruppo Azzurro |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Descrizione generale di Business 4

1.1 Motivazione e scopo del sistema 4

1.2 Funzionalità del sistema 4

1.3 Definizioni, acronimi e abbreviazioni 4

2. Architettura Logica 4

2.1 Presentation Layer 6

2.2 Business Layer Interface 7

2.3 Business Layer 7

2.4 Data Layer Interface 8

2.5 Data Layer 8

1.6 Entity 8

3. Architettura Fisica 10

4. Scelte Tecnologiche 11

4.1 Linguaggio Utilizzato 11

4.2 Framework *Apache* *Struts 1.3* 11

4.3 *Apache Axis 2* 11

4.4 Web Server *Apache Tomcat* 12

4.5 Framework *Quartz* 12

4.6 *Log4J* 12

4.7 *Database* 12

5. Modello Dati 12

5.1 Progettazione Concettuale 13

5.2 Progettazione Logica 14

5.3 Tabelle 14

5.3.1 ACCOUNT 14

5.3.2 GRUPPO 14

5.3.3 ACCOUNT\_GRUPPO 15

5.3.4 FUNZIONALITA 15

15

5.3.5 GRUPPO\_FUNZIONALITA 15

5.3.6 NOTIZIA 16

5.4 Descrizione degli Attributi delle Tabelle 16

5.4.1 Account 16

5.5 Script 19

20

# Descrizione generale di Business

## Motivazione e scopo del sistema

Il sistema software si occupa della gestione del ciclo di vita delle notizie per una redazione giornalistica di un’agenzia di stampa, il cui business primario è la fornitura di news verso clienti esterni. Il sistema è multiutente ed è accessibile via web mediante l’utilizzo di un browser.

## Funzionalità del sistema

Il sistema è accessibile solo da utenti registrati, dunque è richiesta la registrazione al sistema da parte dell’utente. Sono previsti due tipi di utenti : Amministratore e Giornalista.

Il sistema software mette a disposizione dell’utente Amministratore la possibilità di gestire gli Account degli utenti, in particolare può creare, modificare, visualizzare e cancellare Accounts.

L’utente Giornalista invece può accedere alla parte del sistema relativa alla gestione delle notizie, in particolare si occupa della Gestione delle Notizie Online (creazione, modifica, versionamento, visualizzazione, ricerca, cancellazione e archiviazione di una notizia da parte della redazione) e della Gestione delle Notizie in modalità Batch (ricezione automatica di notizie da fornitori esterni ed invio verso client esterni).

## Definizioni, acronimi e abbreviazioni

|  |  |
| --- | --- |
| Nome | Descrizione |
| PL | Presentation layer |
| BLI | Business Logic Interface |
| BL | Business Logic |
| DLI | Data Layer Interface |
| DL | Data Layer |

# Architettura Logica

Il sistema Editoriale si compone di diversi strati applicativi (*layer*), ciascuno dei quali caratterizzato da una forte focalizzazione e specializzazione funzionale. Ogni livello assolve compiti specifici e omogenei, è in grado di comunicare con gli altri *layer* e demanda ad essi le eventuali azioni non di propria pertinenza. Tra i diversi strati esiste una certa gerarchia, nel senso che il rapporto tra i *layer* in generale non è paritetico, ma è regolato da un insieme di dipendenze che permettono di individuare un ordine di relazione. Questo significa che ogni livello "si appoggia" ad uno o più strati per poter eseguire i suoi compiti, dipende da essi e comunica unicamente con loro.

Inoltre l'approccio basato su *layer* consente di limitare l'accoppiamento e le duplicazioni funzionali, promuovendo la creazione di oggetti fortemente specializzati e coesi. La distribuzione delle responsabilità applicative su più oggetti implica che le funzionalità più complesse vengono in generale ottenute dalla collaborazione di più elementi a minore complessità. La scomposizione dell'applicazione in oggetti raggruppati in modo omogeneo a formare i diversi *layer* comporta una migliore organizzazione e strutturazione logica del codice, con vantaggi enormi in termini di manutenibilità e scalabilità.

La suddivisione in *layer* si rivela una soluzione vincente anche in un'ottica di sicurezza applicativa, in quanto, suddividendo l'applicazione in più parti (non solo logiche, ma anche fisiche), è possibile associare a ciascuna di esse le credenziali di sicurezza strettamente necessarie, limitando in questo modo la superficie di attacco e la sensibilità del sistema a vulnerabilità derivanti da bachi interni all'applicazione o legati all'ambiente in cui essa viene eseguita.

La struttura *Multi-Layer* sulla quale si basa il sistema progettato è mostrata nella figura che segue:

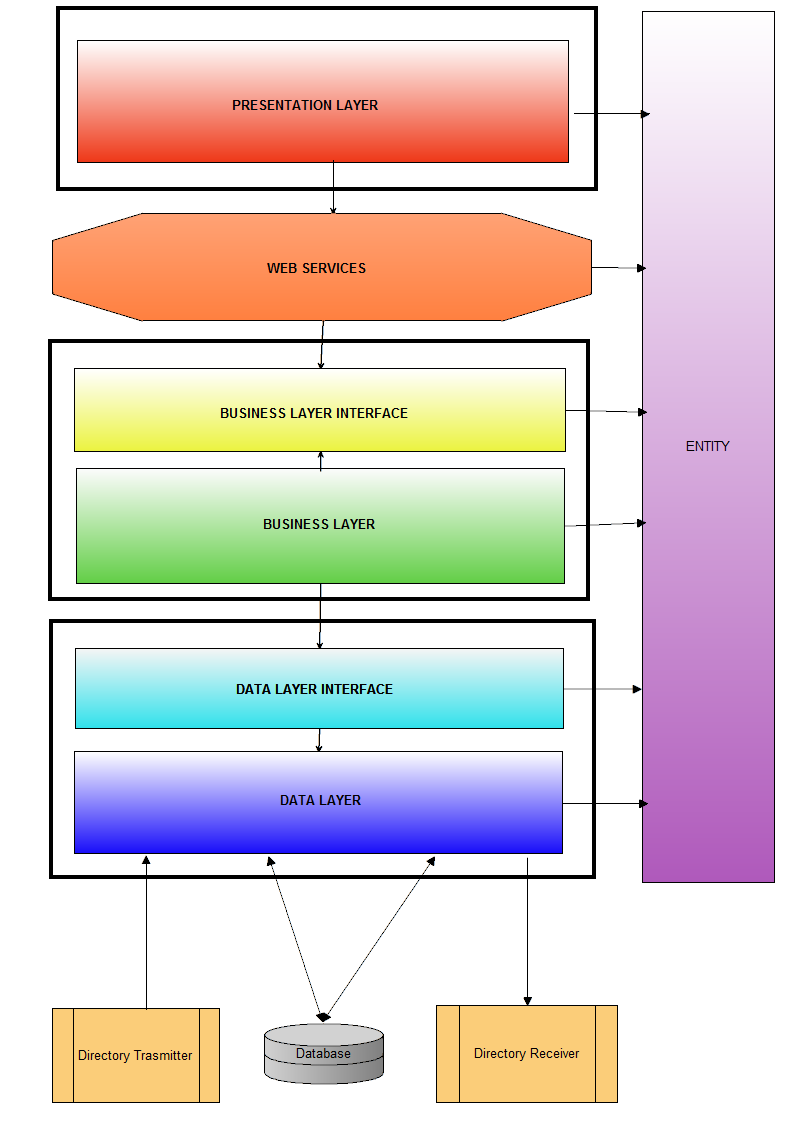


Figura 1 - Architettura Logica *Multi-Layer*

## Presentation Layer

Ha lo scopo di gestire l'interazione del sistema con il mondo esterno, in particolare con gli utenti. Include le maschere per la visualizzazione e l'inserimento dei dati, i controlli, dai più semplici ai più complessi, e i meccanismi per intercettare e trattare opportunamente gli eventi che sono scatenati in funzione delle azioni svolte dagli utenti.

Questo livello è realizzato attraverso l’uso di Servlet, pagine dinamiche JSP, costruite e gestite con l’ausilio del framework *Apache Struts,* il quale permette di realizzare il *Design Pattern MVC*, l’internazionalizzazione e la validazione delle form.

## Business Layer Interface

Strato software che mette a disposizione del livello superiore (*Presentation Layer*) delle interfacce per comunicare con il livello sottostante (*Business Layer*). Tale meccanismo consente di disaccoppiare il *Front-End* dalla componente di logica applicativa.

In particolare, le interfacce messe a disposizione da questo livello sono:

* **NotiziaService** -> consente l’accesso da parte di un giornalista a tutte le funzionalità che si occupano dell’inserimento di una nuova notizia nel database del Sistema Editoriale e della gestione di quelle già presenti.
* **AccountService** -> permette ad un amministratore del Sistema di creare, cancellare o modificare un Account e di visualizzare la lista completa degli account registrati nel Sistema.
* **TrasmissionService** -> offre la possibilità ad un giornalista di trasmettere una notizia ad un cliente.

## Business Layer

Il livello di Business rappresenta il vero e proprio cuore del sistema e racchiude la logica applicativa.

Il compito principale di questa componente è quello di ricevere richieste provenienti dalla componente di *Presentation*, dopo eventuali verifiche (ad esempio l’autorizzazione).

Sulla base delle richieste pervenute avverrà la chiamata agli opportuni moduli del livello sottostante per l’accesso ai dati o ai servizi necessari alla composizione della risposta per l’utente finale.

Tale livello architetturale contiene i *web services* che implementano le interfacce del livello superiore:

* **NotiziaServiceImpl** -> Servizio Web che fornisce le funzionalità messe a disposizione dei giornalisti per la gestione delle Notizie;
* **AccountServiceImpl ->** Servizio Web che fornisce le funzionalità messe a disposizione degli Amministratori per la gestione degli Account;
* **TrasmissionServiceImpl ->** Servizio Web che fornisce le funzionalità messe a disposizione dei Giornalisti per la gestione della trasmissione delle Notizie.

Inoltre, nella *Business Logic* è presente un altro componente, **Validator**, il quale si occupa di controllare che il file XML recuperato dal processo Receiver sia conforme allo schema contenuto nel file “NotiziaReceivedSchema.xsd”; nel caso di esito positivo, richiama il modulo del livello sottostante che si occupa dell’inserimento della nuova notizia nel DB altrimenti sposta il file XML in un’altra cartella rinomando opportunamente il file.

## Data Layer Interface

Strato software che mette a disposizione del livello superiore (*Business Layer*) le interfacce per comunicare con il livello sottostante (*Data Layer*). Tale meccanismo consente di disaccoppiare la logica applicativa dal livello di accesso alle risorse.

In particolare, le interfacce messe a disposizione da questo livello sono:

* **DAO** -> consente l’accesso al *database,* permettendo il disaccoppiamento tra la *Business Logic* e il *Database*, in maniera tale che non occorre conoscere dettagli del database stesso per interrogarlo.
* **Trasmitter** -> permette l’inserimento di una notizia da trasmettere in una Directory condivisa con i Clienti**.**

## Data Layer

Il livello *Data Layer* permette al livelli di *Business* di reperire le informazioni sulle quali effettuare le elaborazioni; i moduli del *Data Layer* contengono la logica che permette sia l’interfacciamento verso sistemi di memorizzazione dati (ad esempio, *database*) che verso *file system* gerarchici.

Il nostro *DataLayer* contiene i *moduli* che implementano le interfacce del livello superiore:

* **DAOImpl** -> Modulo che consente la comunicazione con il database
* **TrasmitterProcess ->** processo che effettua il monitoraggio del *database* per verificare l’esistenza di notizie da trasmettere (il cui stato sia Q); nel caso esse siano presenti, *TrasmitterProcess* genera un file XML, seguendo un determinato schema, da immettere nella cartella condivisa
* **ReceiverProcess ->** processo che effettua il monitoraggio della Directory condivisa con i Fornitori delle notizie al fine di recuperare i file XML presenti in esso.

## 1.6 Entity

Insieme infrastrutturale di classi che vengono utilizzate da ogni livello dell’applicazione. Questa componente dunque comunica con tutti gli altri *Layer* dell’architettura.

In *Figura 2* è stata riportata l’Architettura Logica del Sistema con le componenti previste per ciascun Layer ed i collegamenti tra le stesse.

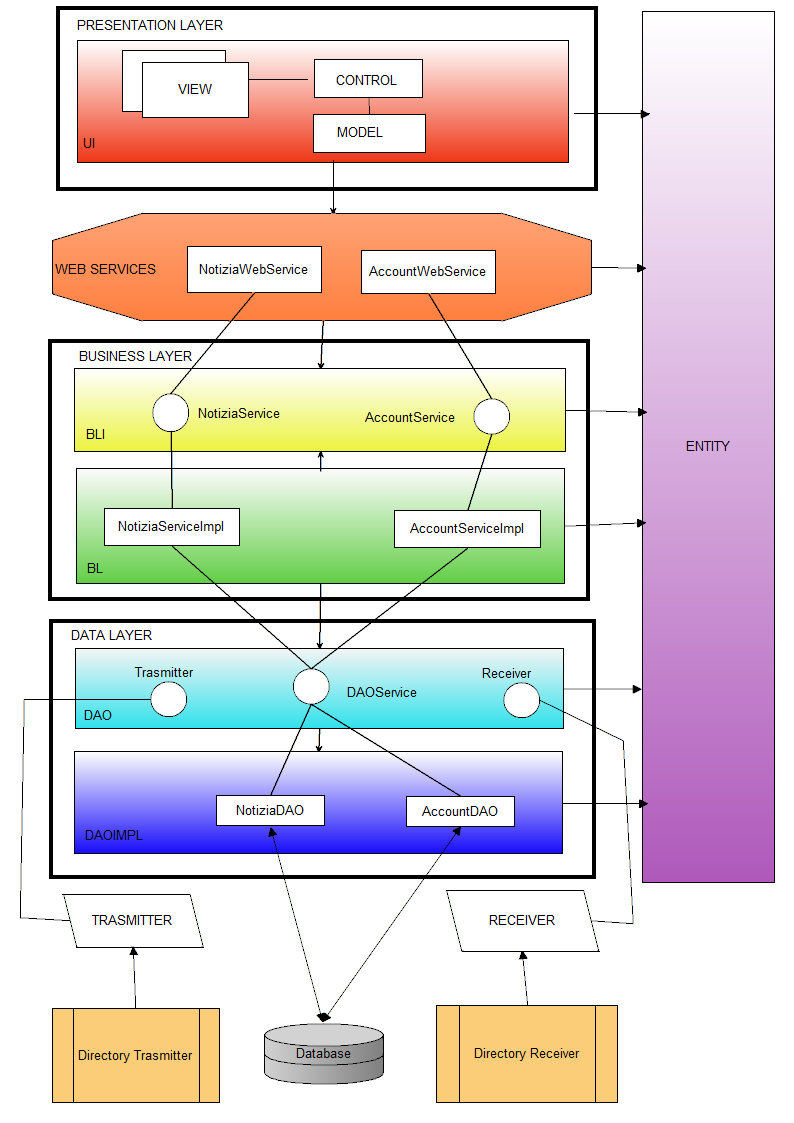


Figura 2 - Architettura Logica con Componenti

# Architettura Fisica

L’architettura dell’applicazione è stata progettata in modo da garantire le seguenti condizioni:

• Alta affidabilità

• Fruibilità con interfacce semplici utilizzabili anche da utenti non esperti

• Elevati livelli di performance per garantire tempi di risposta adeguati

• Scalabilità lineare senza modifiche alla struttura software

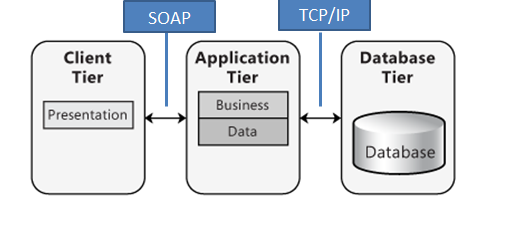
Per fare ciò è stata realizzata un’architettura tecnologica che soddisfi i requisiti di espansibilità sia orizzontale che verticale.

L’espansibilità verticale è data da un sistema che sia in grado di essere messo

in produzione su un unico livello hardware e sia di espandersi sui i n livelli definiti; l’espandibilità orizzontale invece deve essere garantita dalla gestione dei servizi su macchine dedicate.

L’applicazione è stata progettata e implementata in modo da poter essere gestita, in relazione alle esigenze di servizio, da un unica macchina o da una macchina per i diversi livelli infrastrutturale.

In quest’ultimo caso si è pensato di effettuare il deploy dell’applicazione su tre macchine come mostrato in figura :



Client Tier : macchina server su cui risiede il Presentation Layer da cui accede l’utente;

Application Tier : macchina server su cui risiede la logica applicativa: in particolare il Business Layer e Data Layer. Esso offre un’infrastruttura di supporto allo sviluppo della logica intesa come integrazione di più servizi.

Database Tier : macchina server su cui risiedono i dati e le risorse usate dal sistema.

Il Client Tier e Application Tier comunicano tra loro tramite protocollo SOAP basato su file XML.

L’Application Tier e Database Tier comunicano tra loro tramite il protocollo di rete TCP/IP.

# Scelte Tecnologiche

In questo capitolo vengono riportate le Tecnologie utilizzate per l’implementazione del Sistema.

## Linguaggio Utilizzato

È stato utilizzato il linguaggio Java per sviluppare l’applicazione sulla piattaforma Java EE.

## Framework *Apache* *Struts 1.3*

Si è deciso di sviluppare il presentation layer utilizzando *Apache Struts 1.3:* un [framework](http://it.wikipedia.org/wiki/Framework) per lo sviluppo di [applicazioni web](http://it.wikipedia.org/wiki/Applicazione_web) su [piattaforma](http://it.wikipedia.org/wiki/Piattaforma_(informatica)) [Java EE](http://it.wikipedia.org/wiki/Java_EE) conformi al Design Pattern MVC.

L’utilizzo di questo framework supporta vantaggi significativi in termini del progetto:

* Controller centralizzato: gestire le varie richieste dell’utente in un unico file centralizzato
* Internazionalizzazione: permette la visualizzazione del sito in base alla lingua impostata nel proprio browser web dall’utente
* Validazione delle form: consente di validare i campi delle form di inserimento dei dati
* Modularità e Riusabilità: i diversi ruoli dell’applicazione sono affidati a diversi componenti; questo consente di sviluppare codice modulare e più facilmente riutilizzabile.
* Manutenibilità: l’applicazione è costituita da livelli logici ben distinti. Una modifica in uno dei livelli non comporta modifiche negli altri.
* Rapidità di sviluppo: è possibile sviluppare in parallelo le varie parti dell’applicazione, logica di business e di view.

## *Apache Axis 2*

I servizi del business layer sono stati implementati come Web Services utilizzando la tecnologia Apache Axis 2, un’infrastruttura di Apache Software Foundation, mediante la quale è possibile creare, pubblicare ed utilizzare Web Services.

## Web Server *Apache Tomcat*

Sia del presentation layer che del business layer è stato effettuato il deploy su Apache Tomcat: un [contenitore Servlet](http://it.wikipedia.org/wiki/Servlet) [open-source](http://it.wikipedia.org/wiki/Open_source) sviluppato dalla [Apache Software Foundation](http://it.wikipedia.org/wiki/Apache_Software_Foundation) che implementa le specifiche [JavaServer Pages (JSP)](http://it.wikipedia.org/wiki/JavaServer_Pages" \o "JavaServer Pages) e [Servlet](http://it.wikipedia.org/wiki/Servlet" \o "Servlet) di [Sun Microsystems](http://it.wikipedia.org/wiki/Sun_Microsystems" \o "Sun Microsystems), fornendo quindi una [piattaforma software](http://it.wikipedia.org/wiki/Piattaforma_(informatica)) per l'[esecuzione](http://it.wikipedia.org/wiki/Esecuzione_(informatica)) di [applicazioni Web](http://it.wikipedia.org/wiki/Applicazione_Web) sviluppate in linguaggio [Java](http://it.wikipedia.org/wiki/Java_(linguaggio)).

Per gestire l’autenticazione e autorizzazione è stato utilizzato Tomcat Security Realm.

All’interno delle JSP sono state utilizzate le *Tag Libraries;* inoltre è stata usata la *Display Tag Library* per la paginazione degli Account.

## Framework *Quartz*

*Quartz* è una libreria *open-source* che permette di schedulare i propri processi da una qualsiasi applicazione Java; esso è utilizzato per schedulare allo *start-up* i processi *Receiver* e *Trasmitter* del Sistema.

Questo *framework* offre numerosi vantaggi, quali:

* i processi possono essere configurati molto semplicemente;
* i processi possono essere resi persistenti facilmente, perché possono essere memorizzati da qualche parte (ad esempio in un database), ed essere rischedulati in fase di *start-up* dell’applicazione;
* i processi schedulati e il loro flusso di esecuzione può essere monitorato facilmente.

## *Log4J*

Per tenere sotto controllo il comportamento della nostra applicazione in fase di sviluppo, di test ed infine di go-live è stato utilizzato Log4J: una libreria Java sviluppata dall’Apache Software Foundation che consente di mettere a punto un sistema di logging.

## *Database*

Per la persistenza dei dati è stato creato un database utilizzando Oracle DataBase 11g.

# Modello Dati

In questo paragrafo viene illustrata in maniera dettagliata la progettazione del nostro modello dei dati, ricavato a partire dalle specifiche funzionali fornite.

## Progettazione Concettuale

Dai requisiti sono state estratte quattro entità principali: Account, Notizia, Gruppo e Funzionalità. Il seguente modello Entità-Relazione (E-R) fornisce le loro proprietà e le relazioni che le legano:

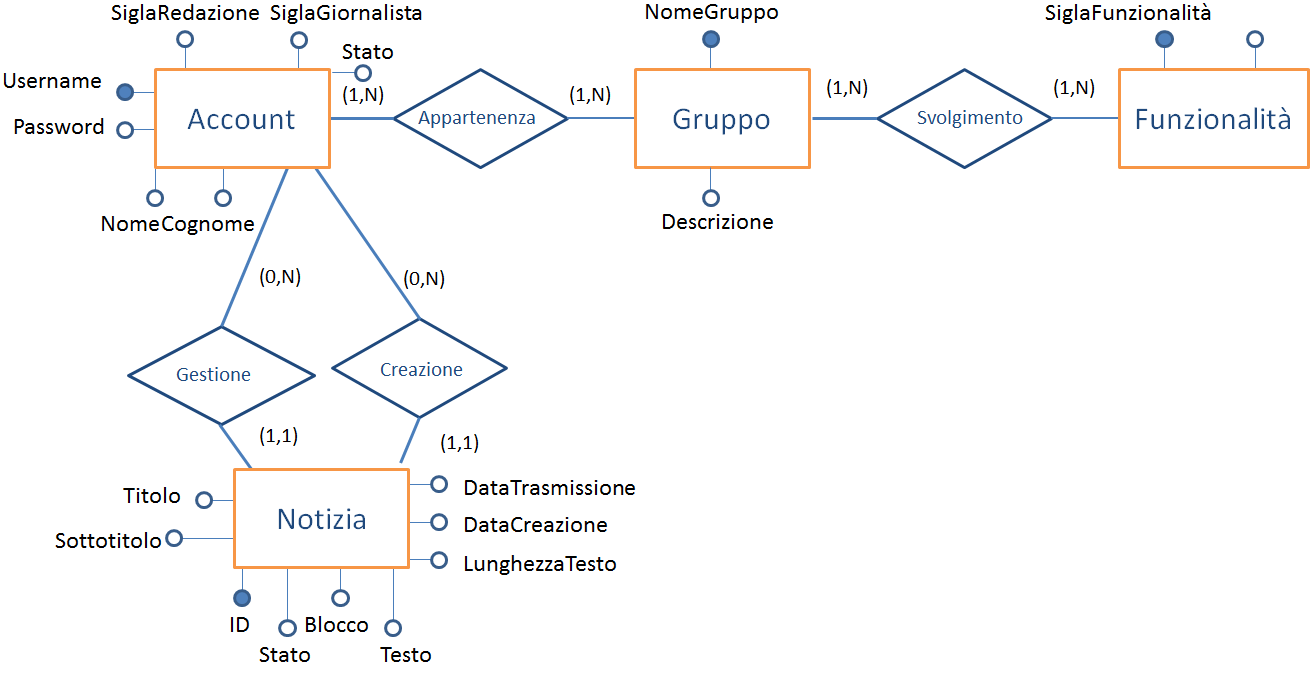


Figura 3 - Schema Entità-Relazione

Assunzioni relative alle relazioni tra le varie Entità:

* **Appartenenza ->** Si suppone che un Account possa appartenere ad uno o più gruppi. Allo stato attuale del Sistema, sono previsti soltanto due tipologie di gruppo: **Giornalisti** ed **Amministratori.** Ad un gruppo devono essere associati uno o più utenti.
* **Svolgimento ->** Tale relazione indica che ad un particolare gruppo sono associate una o più funzionalità specifiche e che una data funzionalità possa essere svolta da uno o più gruppi. Nel nostro caso specifico, in realtà, ciascuna funzionalità è caratteristica di un gruppo soltanto ma la cardinalità è stata impostata molti-a-molti per permette al sistema di essere “estendibile” in termini di possibili aggiunte di ulteriori gruppi di utenti che possono eventualmente utilizzare alcune funzionalità già presenti nel sistema.
* **Creazione** -> Tale relazione mette in corrispondenza una Notizia con l’Account del giornalista che l’ha scritta: una Notizia può essere scritta da un solo giornalista; ed un giornalista può scrivere più notizie.
* **Gestione** -> Tale relazione mette in corrispondenza una Notizia con l’Account del giornalista che effettua sulla stessa operazioni di modifica, cancellazione e trasmissione.

## Progettazione Logica

Lo Schema E-R illustrato in precedenza è stato tradotto nel seguente Modello Relazionale.

**ACCOUNT (***Username*, Password, Nome, Cognome, Sigla\_Redazione, Sigla\_Giornalista, Stato**)**

**GRUPPO (***Nome\_Gruppo, Descrizione***)**

**ACCOUNT\_GRUPPO (***Username : ref Account, Nome\_Gruppo: ref Gruppo***)**

**FUNZIONALITA (***Sigla\_Funzionalità, Nome\_Funzionalità***)**

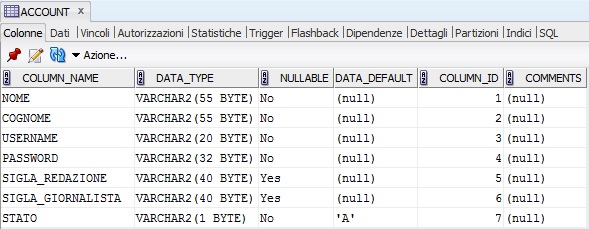
**GRUPPO\_FUNZIONALITA (***Sigla\_Funzionalit : ref Funzionalita, Nome\_Gruppo: ref Gruppo***)**

**NOTIZIA (***ID*, Autore : ref Account, Ultimo\_Digitatore : ref Account, Stato, Blocco, Testo, Lunghezza\_Testo, Titolo, Sottotitolo, Data\_Creazione, Data\_Trasmissione**)**

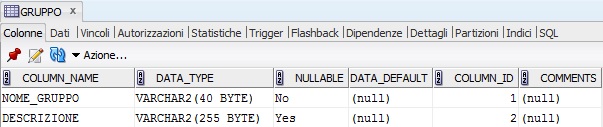
Sono state create quattro tabelle relative alle Entità del Modello E-R, con l’aggiunta di due tabelle associative che tenessero conto delle relazioni molti-a-molti tra Account e Gruppo e tra Gruppo e Funzionalita.

## Tabelle

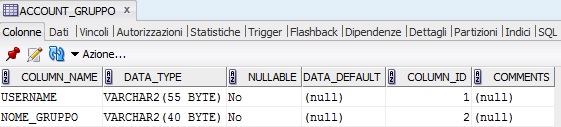
### ACCOUNT



### GRUPPO



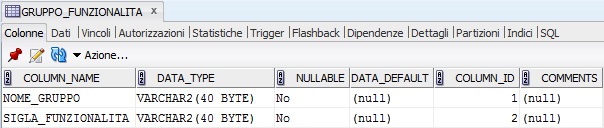
### ACCOUNT\_GRUPPO



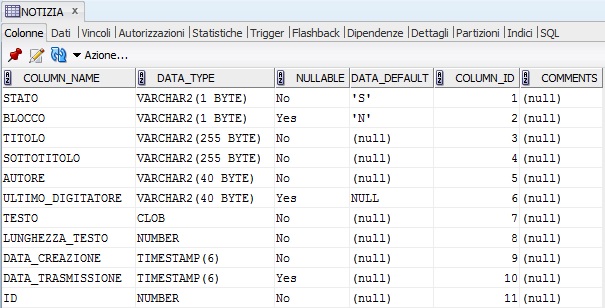
### FUNZIONALITA

### C:\Users\Mary\Dropbox\ProjectWork\tabella.funzionalita.jpg

### GRUPPO\_FUNZIONALITA



### NOTIZIA



## 

## Descrizione degli Attributi delle Tabelle

### Account

|  |  |
| --- | --- |
| **ATTRIBUTO** | **DESCRIZIONE** |
| **username** | PRIMARY KEY – è costituito da un indirizzo e-mail valido dell’utente |
| **nome** | nome del giornalista |
| **cognome** | cognome del giornalista |
| **password** | password associata allo username |
| **siglaRedazione** | sigla univoca di una redazione |
| **siglaGiornalista** | sigla univoca di un giornalista |
| **stato** | può assumere il valore ‘A’(se l’account è attivo) oppure ‘C’ (se l’account è stato cancellato ) |

La scelta per la chiave primaria di questa tabella è ricaduta su **username** (inteso quale indirizzo e-mail valido di un utente) in quanto esso costituisce un valore universalmente univoco. Se avessimo utilizzato, invece, come da specifica, la coppia Nome e Cognome come chiave di Account, non avremmo potuto considerare la possibilità di aggiungere due utenti omonimi all’interno del database.

L’attributo **stato** ci consente di effettuare una cancellazione “logica” di un Account, settandolo al valore ‘C’. In tal modo, anche se viene eliminato l’Account di un giornalista che ha creato o effettuato delle operazioni su una notizia, il giornalista rimarrà in ogni caso associato ad essa.

***3.4.2 Gruppo***

|  |  |
| --- | --- |
| **ATTRIBUTO** | **DESCRIZIONE** |
| **nome\_gruppo** | PRIMARY KEY – nome del gruppo |
| **descrizione** | descrizione della tipologia di utenti associabili a quel gruppo |

Il database avrà per le tipologie di gruppo due valori preconfigurati:

* Giornalista
* Amministratore

***3.4.3 Account\_Gruppo***

|  |  |
| --- | --- |
| **ATTRIBUTO** | **DESCRIZIONE** |
| **username** | riferimento al campo username di Account |
| **nome\_gruppo** | riferimento al campo nome\_gruppo di Gruppo |

Tale tabella associativa lega le tabelle Account e Gruppo in una relazione molti-a-molti.

La coppia (username, nome\_gruppo) costituisce la chiave primaria (PRIMARY KEY) della tabella.

**3.4.4 *Funzionalita***

|  |  |
| --- | --- |
| **ATTRIBUTO** | **DESCRIZIONE** |
| **Sigla\_funzionalita** | PRIMARY KEY – sigla univoca per ogni funzionalità prevista dal Sistema |
| **Nome\_funzionalita** | nome corrispondente di ciascuna funzionalità |

Nel database devono essere preconfigurate, in accordo alle specifiche forniteci, le seguenti funzionalità:

* CreazioneNotizia
* ModificaNotizia
* Annulla
* CancellazioneNotizia
* TrasmettiNotizia
* VisualizzaNotizia
* RegistraNotizia
* ListaNotizie
* ListaAccount
* CreaAccount
* CancellaAccount
* ModificaAccount

***3.4.5 Gruppo\_Funzionalita***

|  |  |
| --- | --- |
| **ATTRIBUTO** | **DESCRIZIONE** |
| **sigla\_funzionalita** | riferimento al campo sigla\_funzionalita di Funzionalita |
| **nome\_gruppo** | riferimento al campo nome\_gruppo di Gruppo |

Tabella associativa che riporta le relazioni tra un gruppo e le funzionalità del Sistema ad esso associato.

Nel nostro caso specifico, il database deve essere preconfigurato associando ai due gruppi le rispettive funzionalità forniteci nelle specifiche.

***3.4.6 Notizia***

|  |  |
| --- | --- |
| **ATTRIBUTO** | **DESCRIZIONE** |
| **ID** | PRIMARY KEY – codice univoco della notizia |
| **stato** | lettera che indica lo stato della notizia: può assumere uno dei seguenti valori : S (editabile), Q(in trasmissione), T(trasmessa), C(cancellata) |
| **blocco** | Lettera che indica se una notizia è in stata presa in carico da un giornalista (Y) o meno (N). |
| **titolo** | Titolo della notizia |
| **sottotitolo** | Sottotitolo della notizia |
| **autore** | Riferimento al campo username dell’Account del giornalista che ha creato la notizia |
| **ultimo\_digitatore** | Riferimento al campo username dell’Account del giornalista che ha impostato il lock sulla notizia, che l’ha cancellata o l’ha trasmessa. |
| **testo** | Testo della notizia |
| **lunghezza\_testo** | lunghezza del testo della notizia espressa come numero intero |
| **data\_creazione** | Data di creazione di una notizia |
| **data\_trasmissione** | Data in cui una notizia viene trasmessa |

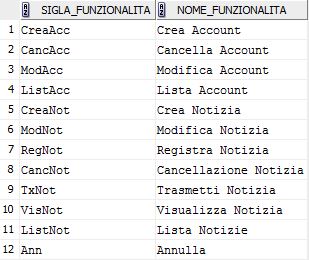
## Script

Per la creazione e la configurazione del database è stato generato uno script che consente di:

* Creare un utente del Database
* Creazione nel database di una Sequenza per la gestione degli ID della tabella Notizia
* Creazione delle tabelle : *Account, Notizia, Gruppo, Funzionalita, Account\_Gruppo, Gruppo\_Funzionalita*
* Creazione del *trigger* per la generazione automatica dell’ID di Notizia e del trigger per il controllo della tipologia dell’account che vuole inserire o modificare una notizia.
* Popolamento delle tabelle *Gruppo, Funzionalita* e *Gruppo\_Funzionalita*

Per la cancellazione del Database è stato generato un ulteriore script.

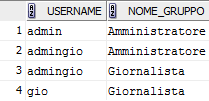
* Risultato del popolamento della tabella *Funzionalita*:



* Risultato del popolamento della tabella *Account*:

### popolamentoTabellaAccount.png

* Risultato del popolamento della tabella *Account\_Gruppo*:



* Risultato del popolamento della tabella *Gruppo\_Funzionalita*:

