**Documento di sviluppo**

**Sistema Editoriale**

**Gruppo Blu**

|  |  |
| --- | --- |
| **Reply** |  |
| Redatto da: | Gruppo Blu |
| Approvato da: |  |

**Indice**

1 Introduzione 3

1.1 Premessa 3

1.2 Contenuti del Documento 3

1.3 Ambito di riferimento 3

2 I servizi offerti dal Sistema Editoriale Gruppo Blu 4

2.1 LoginWS 4

2.2 AccountWS 4

2.3 NewsWS 4

2.4 NewsBatchWS 5

3 Gestione degli errori 6

4 Test JUnit 7

4.1 Test sul login 7

4.2 Test Account 7

4.3 Test Notizia 9

5 Loggin del sistema 11

6 Transmitter e Receiver 12

# Introduzione

## Premessa

Il Sistema Editoriale Gruppo Blu è un sistema informatico che permette di gestire il ciclo di vita delle notizie per una redazione giornalistica di una agenzia di stampa il cui business è la fornitura di news verso clienti esterni (giornali, TV, portali, ecc.).

Il sistema consente alla redazione giornalistica di:

* Gestire in maniera completa le news
* Gestire gli Account dei Giornalisti
* Inviare le news verso i sistemi dei clienti
* Ricevere le news in modo automatico da contributori esterni

## Contenuti del Documento

Obiettivo del seguente documento è quello di descrivere più nel dettaglio i servizi implementati nel sistema e i possibili messaggi di errore. Inoltre, vengono presentati i vari test effettuati sul sistema per verificarne il perfetto funzionamento.

## Ambito di riferimento

Il seguente documento è stato redatto per illustrare ulteriormente le funzionalità del sistema in riferimento ai vari servizi che sono stati pubblicati.

# I servizi offerti dal Sistema Editoriale Gruppo Blu

Nel Sistema Editoriale Gruppo Blu, sono state implementati i seguenti quattro web service:

* un web service per la gestione dell’accesso al sistema;
* un web service per la gestione degli account;
* un web service per la gestione delle notizie online;
* un web service per la gestione delle notizie di tipo batch.

## LoginWS

Il web service LoginWS è stato implementato per gestire l’accesso al sistema e per effettuare tutti i controlli di autenticazione dell’utente loggato sul sistema in modo da verificare a quali funzionalità l’utente può accedere.

E’ composto dal seguente metodo:

* controllaLogin: usato per verificare che l’utente che sta accedendo al sistema abbia inserito le giuste credenziali. Il metodo ritorna l’account dell’utente loggato in modo da poterlo memorizzare in sessione durante tutto il ciclo di vita dell’applicazione.

## AccountWS

Il web service AccountWS è stato implementato per gestire gli account dei giornalisti e amministratori presenti nel sistema.

E’ composto dai seguenti metodi:

* inserisciAccount: usato per inserire l’account di un giornalista nel sistema
* modificaAccount: usato per modificare l’account di un utente che può essere sia giornalista e/o amministratore.
* listaAccountOffset: usato per visualizzare la lista degli utenti presenti nel sistema. Il metodo presenta in ingresso dei parametri che permettono di effettuare la paginazione dell’elenco, in modo da visualizzare non più di 10 utenti per pagina al fine da garantire tempi di risposta più veloci da parte del database.
* cancellaAccount: usato per effettuare la cancellazione logica di un utente giornalista dal sistema.
* ritornaAccount: usato per prelevare dal database l’account dell’utente selezionato per renderlo disponibile alle altre funzionalità del sistema.
* numeroTotUtenti: usato per contare il numero totale di utenti presenti nel database.

## NewsWS

Il web service NewsWS è stato implementato per gestire le notizie online caricate nel sistema.

E’ composto dai seguenti metodi:

* inserisciNotizia: usato per inserire una nuova notizia nel sistema.
* notizia\_in\_modifica: usato per prendere in carico una notizia e bloccarne la sua modifica da parte di altri utenti presenti nel sistema.
* annullaModifica: usato per annullare le modifiche relative a una notizia, rendendola nuovamente disponibile agli altri utenti del sistema.
* modificaNotizia: usato per modificare una notizia presente nel sistema. La modifica viene effettuata solo se la notizia è editabile e non è già presa in carico da un altro utente del sistema.
* rilasciaNotizie: usato per rilasciare le notizie prese in carico in modo da renderle nuovamente disponibili agli altri utenti presenti nel sistema.
* listaNotizieOffset: usato per visualizzare la lista delle notizie presenti nel sistema. Il metodo presenta in ingresso dei parametri che permettono di effettuare la paginazione dell’elenco, in modo da visualizzare non più di 10 notizie per pagina al fine da garantire tempi di risposta più veloci da parte del database.
* visualizza\_per\_parametro\_offset: usato per visualizzare l’elenco delle notizie presenti nel sistema in base a una ricerca effettuata o sul titolo, o sull’autore o sullo stato della notizia. Il metodo presenta in ingresso dei parametri che permettono di effettuare la paginazione dell’elenco, in modo da visualizzare non più di 10 notizie risultanti per pagina al fine da garantire tempi di risposta più veloci da parte del database. Il parametro passato per effettuare la ricerca può essere o l’intero titolo o autore o una o più parole presenti in essi.
* cancellaNotizia: usato per effettuare la cancellazione logica di una notizia presente nel sistema.
* ritornaNotizia: usato per prelevare dal database la notizia selezionata in modo da renderla disponibile alle altre funzionalità del sistema.
* numeroTotNotizie: usato per contare il numero totale di notizie presenti nel database.
* numeroTotNotizieFiltrate: usato per contare il numero totale di notizie restituite dalla ricerca effettuata.

## NewsBatchWS

Il web service NewsBatchWS è stato implementato per gestire le notizie da trasmettere verso altri sistemi editoriali.

E’ composto dal seguente metodo:

* trasmettiNotizia: usato per mettere in coda di uscita una notizia che deve essere trasmessa a un sistema editoriale esterno.

# Gestione degli errori

Il sistema è in grado di gestire le seguenti eccezioni:

|  |  |  |
| --- | --- | --- |
| Codice di Errore | Messaggio | Descrizione |
| COD\_INS | Account già presente | Si verifica quando si cerca di inserire nel sistema un utente già presente. |
| COD\_NOTEXIST | Utente non autorizzato a effettuare questa operazione | Si verifica quando non si è abilitati nell’esecuzione dell’operazione richiesta. |
| COD\_ERRLOG | Accesso non consentito | Si verifica quando si cerca di accedere al sistema con un account non presente in esso. |
| COD\_NON\_EDIT | Notizia non editabile | Si verifica quando si cerca di accedere a una notizia non più editabile. |
| COD\_CON | Errore nella connessione con il db | Si verifica quando si hanno dei problemi di connessione con il database. |

I seguenti codici di errore vengono generati a livello DAO, cioè nel layer più vicino al database, e rilanciati fino ai web service dove vengono catturati e convertiti in AxisFault.

Se un web service lancia quindi uno di questi AxisFault, verranno gestiti a livello front end dove è così possibile risalire dai codici di errore ai messaggi ad essi corrispondenti. Ciò viene effettuato in maniera automatica mediante il framework Struts2. Con tale framework è stato inoltre gestita l’internazionalizzazione dei messaggi nelle seguenti lingue:

* italiano;
* inglese;
* francese;
* spagnolo.

I messaggi verranno quindi visualizzati in base alla lingua scelta nel proprio browser.

# Test JUnit

In questa paragrafo vengono descritti i test effettuati sul livello DAO dell’applicazione per verificare il corretto funzionamento dei servizi implementati.

Per effettuare tali test è stato utilizzato il framework di unit testing JUnit, specifico per il linguaggio di programmazione Java.

I test sono stati suddivisi nel seguente modo:

* test sul login;
* test sull’account;
* test sulle notizie.

## Test sul login

I seguenti test sono stati effettuati per verificare le funzionalità relative alla gestione dell’accesso al sistema.

| Nome Test | Descrizione | Annotazioni/Assert Statement |
| --- | --- | --- |
| TestGetAccount | Viene verificato se i campi Username e Password di un determinato Account sono corretti | AssertEquals |
| TestGetAccountInesistente | Verifica che ad un Username e Password errati non corrisponda alcun Account | @Test(expected=NullPointerException.class) |
| TestGetAccountSenzaParametri | Verifica che ad un Username e Password vuoti non corrisponda alcun Account | @Test(expected=NullPointerException.class) |
| TestControllaAccount | Verifica l’esistenza di un Account nel sistema | AssertTrue |
| TestControllaAccountInesistente | Verifica che nel sistema non sono presenti un determinato username e password | AssertFalse |

## Test Account

I seguenti test sono stati effettuati per verificare le funzionalità relative alla gestione dell’inserimento, modifica, visualizzazione e cancellazione degli account.

| Nome Test | Descrizione | Annotazioni/Assert Statement |
| --- | --- | --- |
| TestInserisciAccount | Viene verificato l’inserimento di un Account nel sistema | *AssertEquals* |
| TestInserisciAccountGiaPresente | Si verifica che non venga inserito nel sistema un utente giornalista già presente | *@Test(expected=Exception.class)* |
| TestInserisciAccountUsernameDiverso | Si verifica la possibilità di inserire nel sistema un utente che differisce da un altro solo per Username | *AssertEquals* |
| TestRitornaAccount | Si verifica l’esistenza di un Account nel sistema | *AssertEquals* |
| TestRitornaAccountSoloGiornalista | Si verifica che dato un determinato Account venga restituito il gruppo di appartenenza | *AssertEquals* |
| TestRitornaAccountInesistenza | Verifica l’impossibilità di un utente non registrato di accedere al sistema | *@Test(expected=NullPointerException.class)* |
| TestAggiungiFunzionalitàAccount | Verifica l’aggiunta della funzionalità giornalista ad un amministratore | *AssertEquals* |
| TestModificaAccount | Verifica la modifica dei parametri di un Account | *AssertEquals* |
| TestCancellaAccountSoloGiornalista | Verifica l’effettiva cancellazione di un Account giornalista | *AssertEquals* |
| TestCancellaAccountAmministratore | Verifica la cancellazione della funzionalità giornalista ad un amministratore | *AssertEquals* |

## Test Notizia

I seguenti test sono stati effettuati per verificare le funzionalità relative alla gestione delle notizie da parte dei giornalisti.

| Nome Test | Descrizione | Annotazioni/Assert Statement |
| --- | --- | --- |
| TestInserisciNotizia | Viene verificato se la Notizia viene inserita nel DataBase | *AssertTrue* |
| TestInsertCampoMancante | Si verifica che la Notizia non venga inserita nel DB, nel caso in cui l’utente non riempie tutti i campi previsti dalla form | *@Test(expected=SQLException.class)* |
| TestCancellaNews | Si verifica che la notizia venga cancellata logicamente settando il campo Stato a “C” | *AssertTrue* |
| TestCancellaNewsGiaCancellata | Si verifica che non è possibile cancellare una notizia già cancellata | *@Test(expected=Exception.class)* |
| testListaNews | Si verifica che vengano restituite un numero di notizie pari al più a quelle previste dalla paginazione | *AssertEquals* |
| testVisualizzaNotizia | Si verifica che venga visualizzata la Notizia | *AssertEquals* |
| testVisualizzaNewsPerTitolo | Si verifica che venga restituita la lista delle notizie con un determinato titolo | *AssertTrue* |
| testVisualizzaNewsPerAutore | Si verifica che venga restituita la lista delle notizie di un determinato autore | *AssertTrue* |
| testVisualizzaNewsPerStato | Si verifica che venga restituita la lista delle notizie con un determinato stato | *AssertTrue* |
| testPermessoUpdate | Si verifica il permesso di modificare una notizia quando non è bloccata e lo stato è editabile | *AssertTrue* |
| testUpdateNews | Si verifica che la notizia sia effettivamente modificata | *AssertEquals* |
| testNews\_in\_update | Si verifica che il lock della notizia sia posto ad ‘Y’ | *AssertTrue* |
| testPermessoLockuno | Si verifica che una notizia bloccata non sia modificabile da parte di altri giornalisti | *AssertFalse* |
| testPermessoLockdue | Si verifica che una notizia bloccata sia modificabile da parte del giornalista che l’ha bloccata | *AssertTrue* |
| testPermessoUpdateStato | Si verifica che non sia permesso modificare una notizia non editabile | *AssertFalse* |
| testCancellaUpdateNews | Si verifica che venga annullata la modifica ad una determinata notizia rendendola disponibile | *AssertTrue* |
| testCancellaNewsLockata | Si verifica che una notizia bloccata non possa essere cancellata | *AssertFalse* |
| testCancellaNewsLockataUltimoAcc | Una notizia bloccata è cancellata solo dal giornalista che l’ha bloccata | *AssertTrue* |
| Testrilascia­\_News | Si verifica che vengano sbloccate tutte le notizie prese in carico da un giornalista | *AssertTrue* |

**NB: Per eseguire alcuni dei test relativi alla Notizia, è necessario inserire l’ID di una notizia presente nel database.**

# Loggin del sistema

Come ulteriore verifica del corretto funzionamento del nostro sistema, si è utilizzata la libreria Java Log4j per tracciare le chiamate ai metodi invocati durante l’esecuzione dell’applicativo.

Log4j è stato configurato con il seguente file di properties:

log4j.logger.logApp=INFO, MYLOG\_CONS, MYLOG\_FILE

log4j.appender.MYLOG\_CONS=org.apache.log4j.ConsoleAppender

log4j.appender.MYLOG\_FILE.append=true

log4j.appender.MYLOG\_FILE=org.apache.log4j.DailyRollingFileAppender

log4j.appender.MYLOG\_FILE.File=C:\\infoLog.log

log4j.appender.MYLOG\_FILE.DatePattern= '.'yyyy-MM-dd '.log'

log4j.appender.MYLOG\_CONS.layout=org.apache.log4j.PatternLayout

log4j.appender.MYLOG\_CONS.layout.ConversionPattern=%d [%t] %C**{1}** - %m%n

log4j.appender.MYLOG\_FILE.layout=org.apache.log4j.PatternLayout

log4j.appender.MYLOG\_FILE.layout.ConversionPattern=%d [%t] - %m%n

I messaggi verranno quindi in tal modo tracciati sia nella console dell’application server, che memorizzati in un file creato nel server all’interno del drive C, chiamato infoLog.log.

Sono stati inseriti esclusivamente messaggi di tipo Info.

# Transmitter e Receiver

Per la schedulazione dei processi che implementano le funzionalità di ricezione e trasmissione di notizie da e verso l’esterno del sistema, è stato utilizzata la libreria Quartz.

Si è deciso di implementare i due processi come due applicazioni web che verranno avviate nel server e resteranno in esso attive fino alla chiusura del server stesso. Le due applicazioni contengono al loro interno due processi (job) che verranno attivati da Quartz ogni 5 secondi per verificare se hanno notizie da elaborare.