**UIIP**

**Project Work Requirements**

**Sistema Editoriale**

|  |  |
| --- | --- |
| **Reply** |  |
| Redatto da: | Michele Pantaleone – Mario Stefanutti |
| Approvato da: | Saverio Ciotoli – Alessia Spaccatrosi |

**Storia del documento**

|  |  |  |  |
| --- | --- | --- | --- |
| Codice documento | Data | Autore | Storia |
| 1.0 | 30-07-2012 | Michele Pantaleone | Prima stesura |
| 1.1 | 05-04-2013 | Stefanutti Mario | Seconda stesura |

**Lista di distribuzione**

|  |  |
| --- | --- |
| **Società** | **Nominativo** |
| Sytel Reply Roma | Saverio Ciotoli |
| Sytel Reply Roma | Alberto Antonelli |
| Sytel Reply Roma | Fabrizio Grisoli |
| Sytel Reply Roma | Fabrizio Fiodo |
| Sytel Reply Roma | Stefanutti Mario |
| Sytel Reply Roma | Alessia Spaccatrosi |

**Documenti di riferimento**

|  |  |  |  |
| --- | --- | --- | --- |
| **RIF.** | **Codice** | **Data** | **Titolo** |
|  |  |  |  |
|  |  |  |  |

**Allegati al documento**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fornito da.** | **Codice documento** | **Data** | **Titolo** |
|  |  |  |  |

**Abbreviazioni e Acronimi**

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

**Indice**

1 Introduzione 4

1.1 Scopo del Documento 4

1.2 Scopo del Progetto 4

2 Descrizione generale 4

2.1 Specifiche da implementare 4

2.2 Architettura 6

2.3 Business Data Model 6

3 Organizzazione di Progetto 9

3.1 Project Life Approach 9

3.2 Modello temporale 10

3.3 Modalità di rilascio e verifica 11

3.4 Esempio di gestione operativa 12

4 Descrizione processi 21

4.1 Accesso al sistema 21

4.1.1 Lista Account 21

4.1.2 Crea Account 21

4.1.3 Modifica Account 21

4.1.4 Cancella Account 21

4.2 Gestione Notizia On-Line 22

4.2.1 Crea Notizia 22

4.2.2 Lista Notizie 22

4.2.3 Modifica Notizia 22

4.2.4 Annulla 22

4.2.5 Visualizza Notizia 23

4.2.6 Registra Notizia 23

4.2.7 Cancella Notizia 23

4.3 Gestione Notizia Batch 23

4.3.1 Trasmetti Notizia 23

4.3.2 Ricezione Notizie provenienti dai sistemi contributori (Fornitori) 23

4.3.3 Trasmissione Notizie dalla redazione verso i sistemi Clienti. 24

5 Requisiti non funzionali 25

5.1 Gestione dei messaggi e dei codici di errore. 25

5.2 Scalabilità del Sistema Editoriale. 25

5.3 Logging 25

6 Documentazione da produrre 26

6.1 Procedura di installazione e setup 26

# Introduzione

## Scopo del Documento

Scopo del documento è quello di fornire, nell’ambito di un progetto didattico più ampio, le linee guida per lo sviluppo di un sistema informatico (Project Work).

I capitoli successivi descrivono i requisiti funzionali e le specifiche tecniche da seguire per la realizzazione del sistema.

## Scopo del Progetto

Lo scopo del progetto è la realizzazione di un sistema editoriale in grado di gestire il ciclo di vita delle notizie per una redazione giornalistica di una agenzia di stampa il cui business primario è la fornitura di news verso clienti esterni (giornali, tv, portali, ecc.).

# Descrizione generale

Il sistema deve essere multiutente ed accessibile via web (http) mediante l’uso di un browser (Internet Explorer, Chrome, Firefox), quindi senza dover installare nessun software sui PC dei giornalisti.

Gli accessi devono essere controllati e vincolati all’inserimento di login e password per far accedere solo utenti autorizzati e appartenenti alla redazione.

Oltre a consentire una gestione completa delle notizie da parte della redazione (creazione, modifica, versionamento, visualizzazione, ricerca, cancellazione, archiviazione, ecc.), il sistema deve anche poter ricevere notizie in modalità automatica da contributori esterni e poterle successivamente inviare verso i sistemi dei clienti.

## Specifiche da implementare

Il sistema redazionale deve implementare i seguenti requisiti:

Dare ai giornalisti la possibilità di

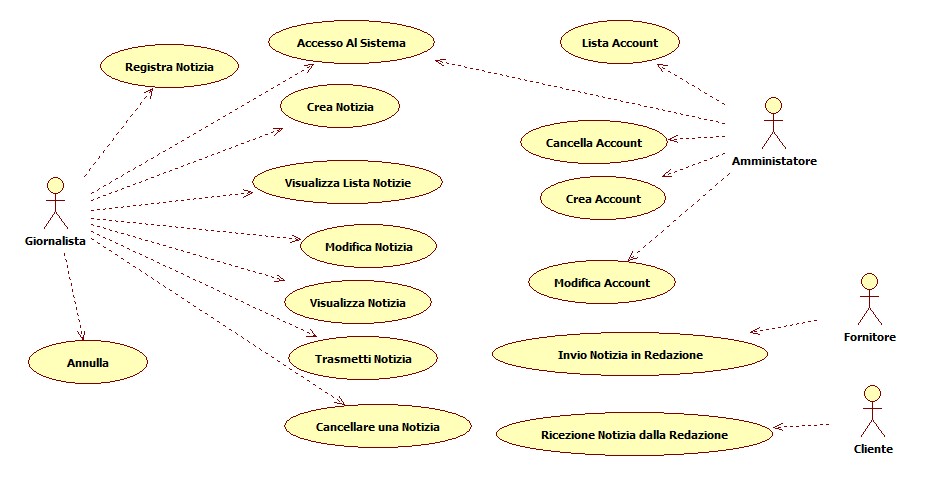
* accedere alle funzionalità del sistema via web, tramite browser, previo inserimento di username e password
* visualizzare la Lista delle Notizie presenti nella redazione
* creare una nuova Notizia
* modificare in modalità esclusiva una Notizia presente nella redazione
* registrare le modifiche effettuate sui campi della Notizia
* visualizzare una Notizia presente nella redazione
* cancellare una Notizia presente nella redazione
* trasmettere una Notizia verso un sistema esterno.
* annulla le modifiche apportate ad una Notizia prese in modifica

Dare ad un amministratore la possibilità di

* creare un account per un giornalista
* cancellare un account per un giornalista
* modificare un account per un giornalista
* visualizzare la lista degli account presenti.

Dare la possibilità ai sistemi Fornitori di inviare notizie al sistema redazionale su directory condivisa in formato xml.

Dare la possibilità sistemi Clienti di ricevere le notizie dal sistema editoriale su directory condivisa in formato xml.



*Figura 2.1.1 Use Case dei requisiti utente.*

## Architettura

Presentation Layer (WEB Application)

Business Layer (Web Services SOAP over http)

Receiver

Transmitter

Data Layer

(StoreProcedures PLSQL )

Database

*Figura 2.2.1 Architettura del sistema editoriale.*

L’architettura del sistema deve essere multilayer e rispettare la specifica JavaEE.

Si richiede di sviluppare il Presentation Layer utilizzando un qualsiasi framework opensource (Struts, GWT, SpringMVC, l’implementazione JSF di Oracle, ecc.).

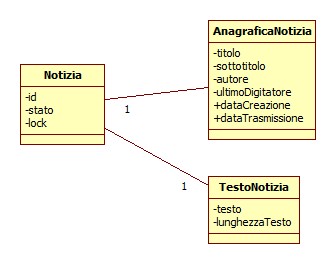
I servizi del Business Layer devono essere implementati come WebServices utilizzando la tecnologia Apache Axis 2 (<http://axis.apache.org/axis2/java/core/>). Sia il presentation layer che il Business Layer devono essere implementati sul WebServer Apache Tomcat (<http://tomcat.apache.org/>). L’accesso ai dati presenti nel Data Layer deve essere realizzato implementando delle Stored Procedure in Oracle PLSQL.

Il Database deve essere realizzato in Oracle DataBase 11g: http://www.oracle.com/it/products/database/index.html

Il Receiver e il Trasmitter devono essere dei processi Java che tramite l’utilizzo del framework Quartz (<http://quartz-scheduler.org/>) vengono schedulati allo startup.

## Business Data Model

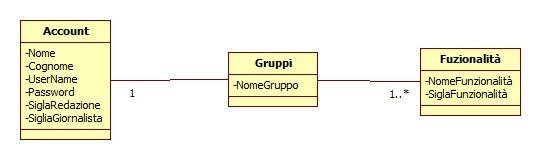
Di seguito viene rappresentato il modello dei dati della Notizia sui quali il sistema deve effettuare le elaborazioni.



Descrizione campi della Notizia:

* id 🡪 chiave della notizia
* stato 🡪 indica lo stato della notizia:
  + S (editabile), Q (In trasmissione), T (trasmessa), C (Cancellata)
* lock 🡪 indica se la notizia è stata già presa in carico da un giornalista:
  + Y (bloccata da un altro utente), N (disponibile)
* Titolo 🡪 titolo della notizia: qualsiasi valore
* sottotitolo 🡪 sottotitolo della notizia: qualsiasi valore
* autore 🡪 sigla del giornalista che ha creato la notizia
* utlimoDigitatore 🡪 sigla dell’ultimo giornalista che ha modificato la notizia
* testo 🡪 testo della notizia
* lunghezzaTesto 🡪 lunghezza del testo della Notizia espressa come un numero intero.
* DataCreazione 🡪 data di creazione della notizia
* DataTrasmissione 🡪 data in cui la notizia viene trasmessa
* UltimoDigitatore 🡪 sigla dell’ultimo giornalista che ha impostato il lock sulla Notizia o che l’ha trasmessa

Di seguito viene riportato il modello dei dati dell’account che deve essere utilizzato per la gestione degli accessi al sistema.



L’account deve essere associato ad uno o più gruppi ai quali vengono associate specifiche funzionalità.

I gruppi possibili (Pre-configurati sul DataBase) sono:

* Amministratore
* Giornalista

Le funzionalità possibili (Pre-configurate sul DataBase) sono:

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

L’associazione tra i gruppi e le funzionalità devono essere già pre-configurate sul DataBase come descritto nella tabella seguente.

|  |  |
| --- | --- |
| Gruppo | Funzionalità |
| Amministratore | CreaAccount |
| Amministratore | CancellaAccount |
| Amministratore | ModificaAccount |
| Amministratore | ListaAccount |
| Giornalista | CreazioneNotizia |
| Giornalista | ModificaNotizia |
| Giornalista | RegistraNotizia |
| Giornalista | CancellazioneNotizia |
| Giornalista | TrasmettiNotizia |
| Giornalista | VisualizzaNotizia |
| Giornalista | ListaNotizie |
| Giornalista | Annulla |

N.B.

Tutti gli accessi ai dati devono essere effettuati tramite l’utilizzo di StoreProcedure scritte in linguaggio Oracle PLSQL. Non utilizzare codice SQL direttamente dai WebService.

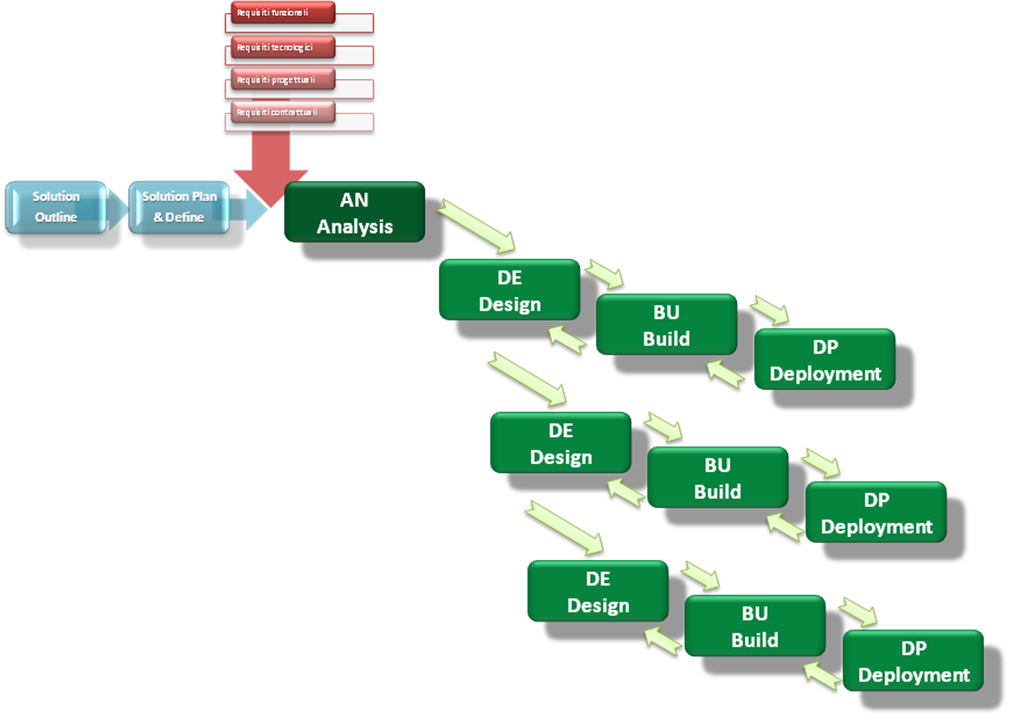
# Organizzazione di Progetto

Di seguito verrà descritto il ciclo di vita del progetto, il modello temporale con le relative release e la modalità di rilascio.

## Project Life Approach

L’approccio che dovrà essere adottato è di tipo Incrementale.

L’approccio incrementale prevede un rilascio multiplo del sistema, attraverso più cicli di Design (DE), Build (BU) e Deployment (DP); pertanto le funzionalità e le capacità previste sono rilasciate in modo progressivo (incrementale).

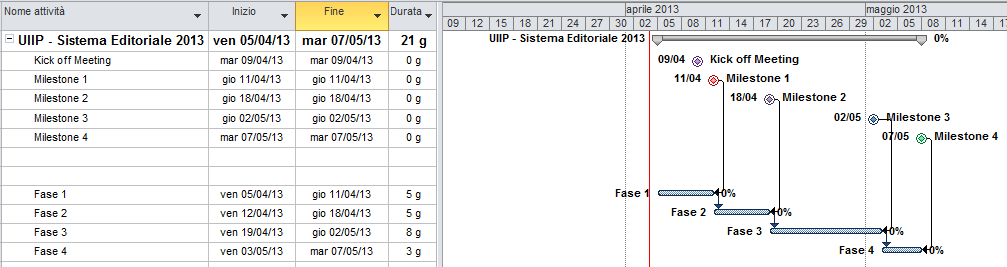


Le fasi di Design (DE), Build (BU) e Deployment (DP), sono suddivise in unità di lavoro più piccole incluse all’interno di più cicli di lavorazione, che iniziano quando la fase di Analysis (AN) è completamente definita.

A fronte di un piano dei rilasci prestabilito, l’approccio incrementale garantisce che tutti i contenuti del Project Life Model (fasi, stream, deliverable) siano completati per ogni rilascio.

## Modello temporale

Il progetto inizia il 09/04/2013 e termina il 07/05/2013 e prevede 4 Milestone principali come mostrato dal seguente digramma di Gantt:



Da questo verranno istituite 4 Fase progettuali che sfoceranno in una Baseline di rilascio software/documentale autoconsistente come di seguito descritto:

* **Fase 1** – data rilascio 11/04/2013
  + Documento di architettura comprensivo di modello dati e tecnologie scelte
  + Script creazione DB
  + Manuale di installazione (per quanto verrà rilasciato in questa fase)
* **Fase 2** – data rilascio 18/04/2013
  + Interfacce tra i sistemi (WSDL dei WS) per tutti i processi previsti
  + Implementazione dei processi dell’area ‘[Accesso al sistema](#_Accesso_al_sistema)’
  + Manuale di installazione (aggiungere direttive a quello esistente per quanto verrà rilasciato in questa fase)
  + Manuale Utente (per quanto verrà rilasciato in questa fase)
* **Fase 3** – data rilascio 02/05/2013
  + Implementazione Processi ‘[Gestione Notizia On-Line](#_Gestione_Notizia_On-Line)’ e ‘[Gestione Notizia Batch](#_Gestione_Notizia_Batch)’
  + Manuale di installazione (aggiungere direttive a quello esistente per quanto verrà rilasciato in questa fase)
  + Manuale Utente (aggiungere direttive a quello esistente per quanto verrà rilasciato in questa fase)
* **Fase 4** – data rilascio 07/05/2013
  + Miglioramento interfaccia grafica
  + Revisione documentazione
  + Consolidamento rilascio finale

## Modalità di rilascio e verifica

Come repository software verrà utilizzato githup (https://github.com/):

**account** github

**name**: sytelUiip

**user**:uiip2013.sytel@gmail.com

**pass**: syteluiip2013

**repository**: <https://github.com/sytelUiip/projectWork.git>

In questo documento è riportata la struttura Github da utilizzare per la gestione dei progetti Uiip e un esempio di quella che potrebbe essere una corretta gestione operativa dei vari repository.

Su Github è stato creato l’utente **sytelUiip** sul quale sono stati configurati i seguenti repository:

* projectWork (progetto globale contenente documentazione e item generici)
* UIIP\_2013\_Azzurro (Repository di riferimento per il gruppo Azzurro)
* UIIP\_2013\_Giallo (Repository di riferimento per il gruppo Giallo)
* UIIP\_2013\_Verde (Repository di riferimento per il gruppo Verde)
* UIIP\_2013\_Rosso (Repository di riferimento per il gruppo Rosso)
* UIIP\_2013\_Blu (Repository di riferimento per il gruppo Blu)

Ogni gruppo di progetto **Uiip** preleva le risorse condivise dal repository **projectWork** e utilizza per gli sviluppi il repository associato al proprio gruppo.

Per la gestione del repository di gruppo GitHub ogni utente di ogni gruppo deve:

* Creare un utenza su Github
* Con l’utenza appena creata effettuare la **fork** del repository associato al proprio gruppo
* Effettuare il **clone** locale della propria copia locale del repository
* Lavorare sul proprio repository locale effettuando i commit necessari
* Allineare la propria copia remota del repository.
* Effettuare delle richieste di allineamento tra la propria copia remota e il repository base dell’utente sytelUiip.

L’utente **sytelUiip** esamina le richieste di allineamento ricevute per ciascuno dei repository base gestiti (ovvero per tutti i repository associati ai gruppi) ed eventualmente accetta le modifiche apportate.

L’utenza sytelUiip può essere data ad una sola persona responsabile dell’allineamento di tutti i repository di gruppo o ad una persona per ciascun gruppo che dovrebbe gestire solamente il repository base delle proprio gruppo.

I rilasci e gli item presi in considerazione dagli utenti esterni ai gruppi **sono esclusivamente quelli presenti sui repository base**.

## Esempio di gestione operativa

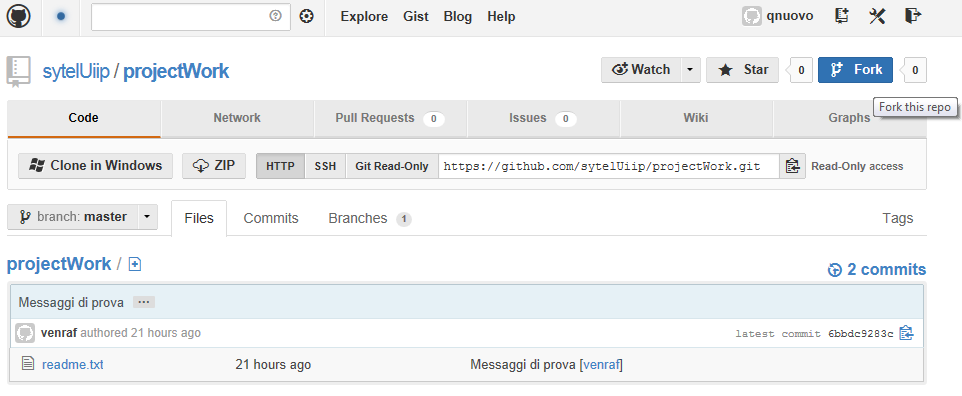
L’esempio di procedura operativa riportata di seguito è basato sul gestore grafico di GitHub installato localmente su ambiente windows.

Creare un nuovo utente (es: **qnuovo**) su: <https://github.com/>

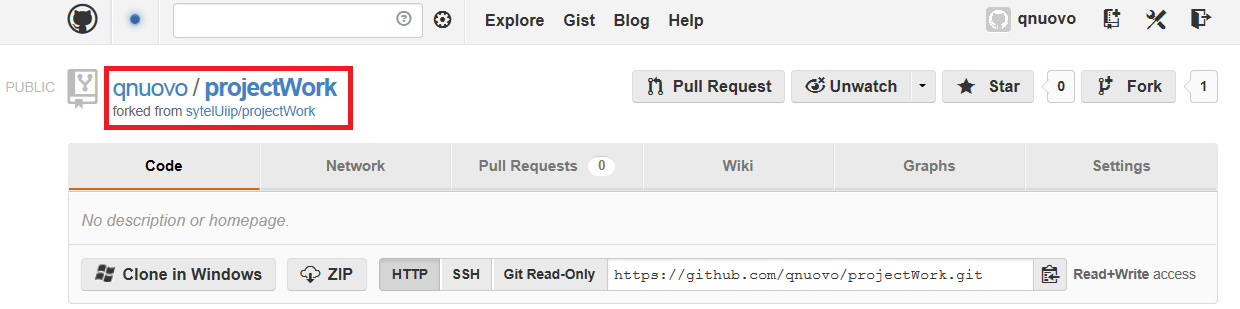
Effettuare il **sign-in**. Ricercare il repository master sul quale si intende operare (es: **projectWork**):



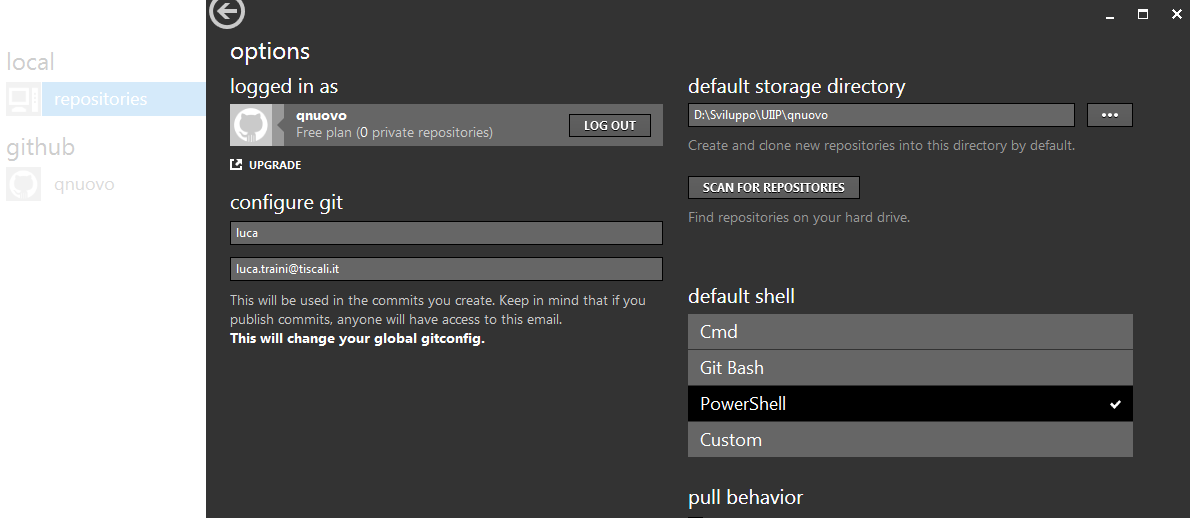
Selezionare quindi il repository individuato ed effettuare il fork:



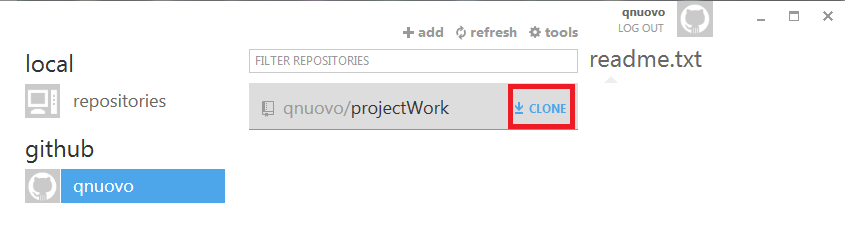
Il repository per il quale di è effettuato il fork risulta ora tra i repository del nuovo utente:



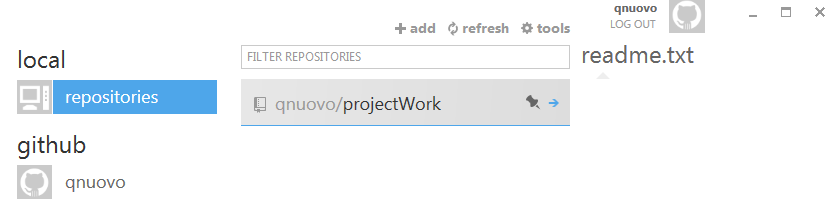
Aprire l’applicazione grafica GitHub, autenticarsi con l’utente creato su <https://github.com/> e definire opportunamente i parametri di configurazione (es: Storage Directory dove verranno craeti i cloni locali dei repository).



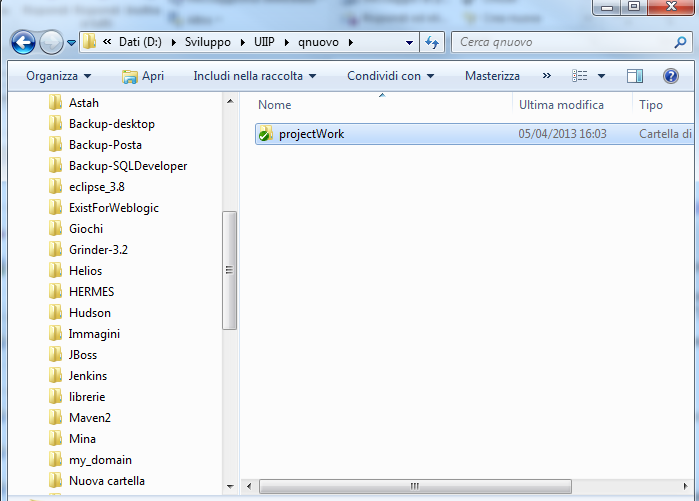
Effettuare il clone locale del fork remoto:



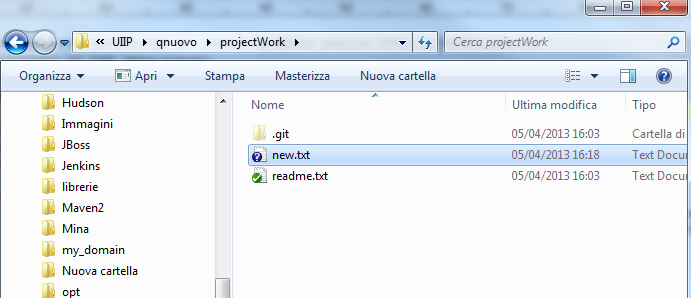
Terminato l’operazione il clone locale viene aggiunto tra i **local repository**:



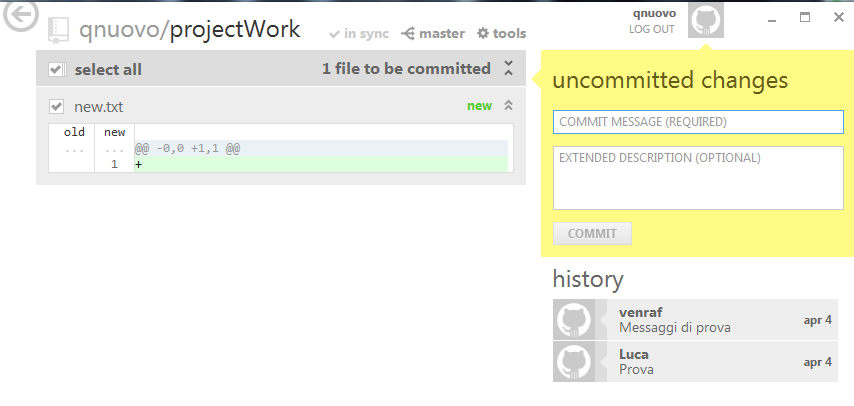
La copia locale del repository è stata creata a partire dalla Storage Directory:



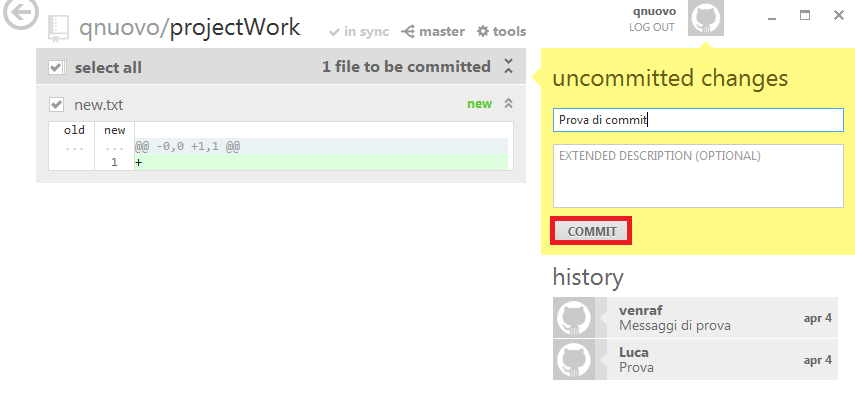
A questo punto è possibile apportare modifiche sulla propria copia locale del repository. Ad esempio creare il file **new.txt**.

****

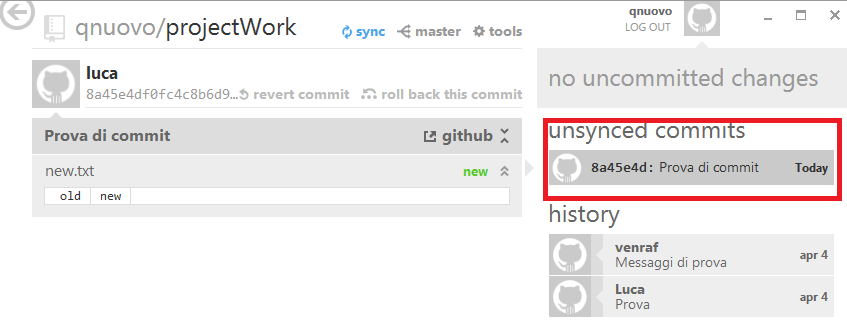
Analizzando il repository locale dall’interfaccia grafica vengono mostrati gli item per i quali è possibile effettuare il commit sul repository locale:



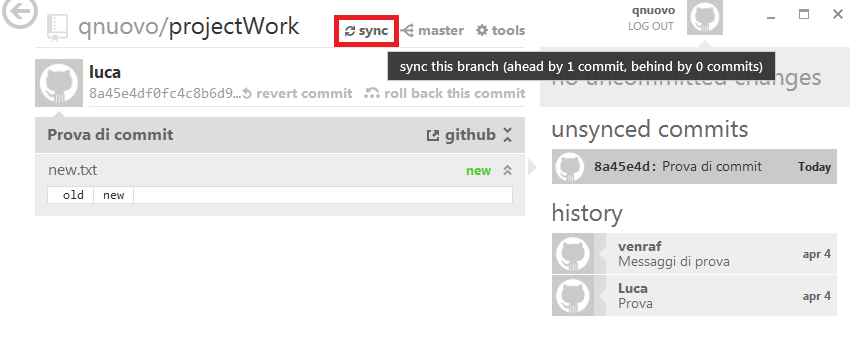
Per effettuare il commit è necessario inserire un messaggio:



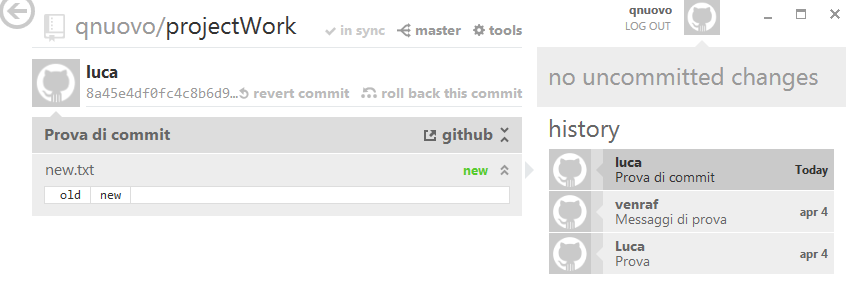
Il commit appena effettuato ha effetto esclusivamente sul clone locale del fork:

****

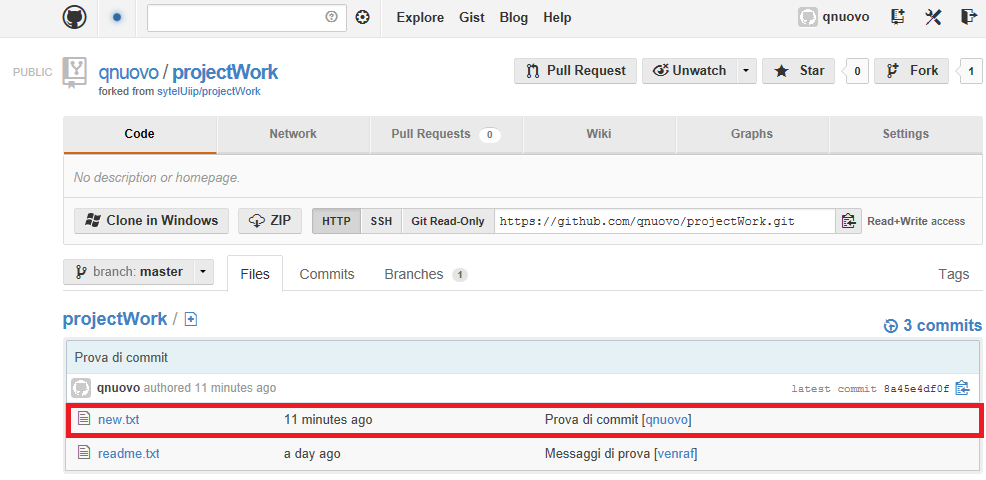
Per sincronizzare il clone locale con il fork remoto è sufficiente eseguire l’operazione di **sync**:

****

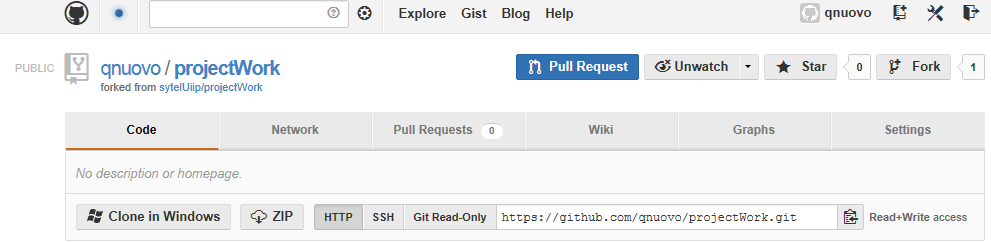
Al termine della sincronizzazione il fork e il clone risultano allineati:

****

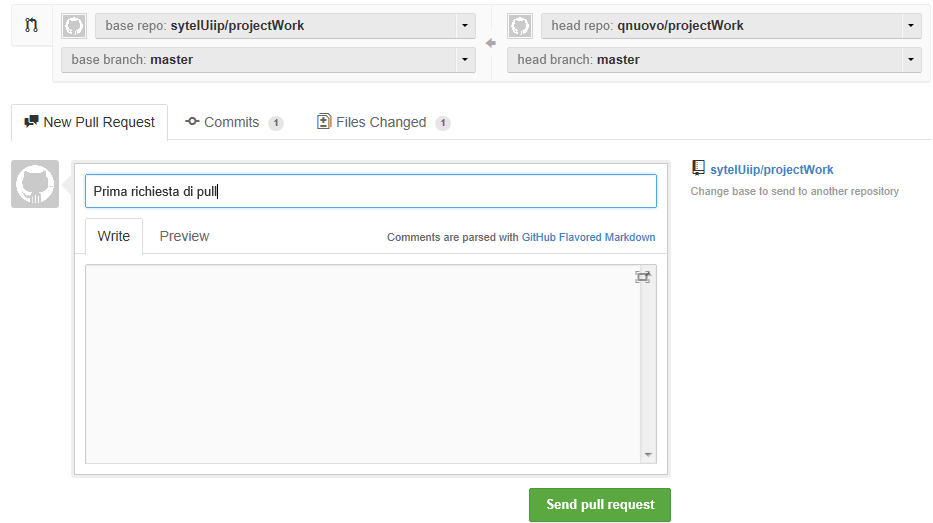
Analizzando il fork remoto direttamente dal sito di GitHub è possibile constare l’avvenuto allineamento:



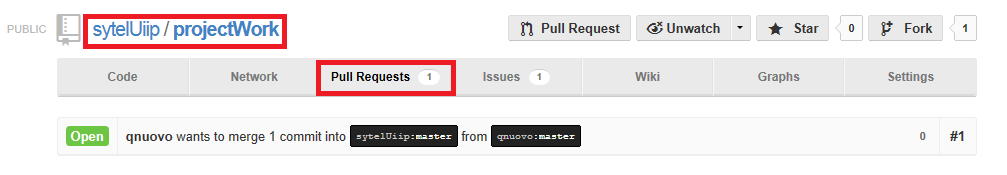
A questo punto è possibile riportare le modifiche apportate sul fork sul repository originale dell’utente **sytelUiip.** Le modifiche non possono essere riportate autonomamente dal fork al repository base, ma è invece possibile inviare una *richiesta di allineamento*, una **Pull Request**:



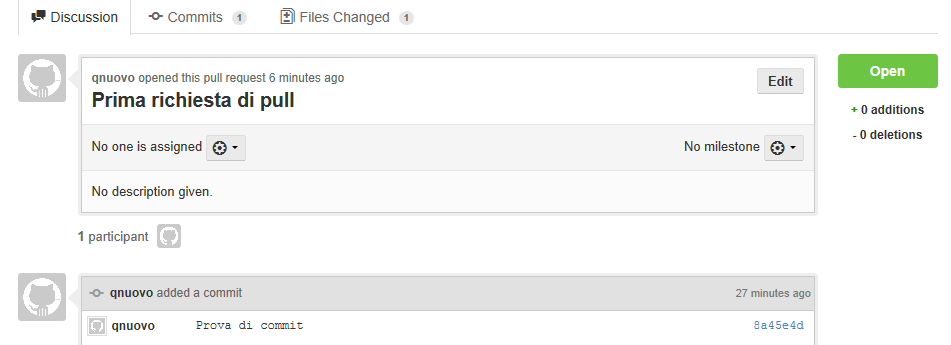
Scelti i branch del fork e del repository originale sui quali si intende operare ed inseriti titolo e descrizione, è possibile inviare la richiesta col comando **Send Pull Request**:



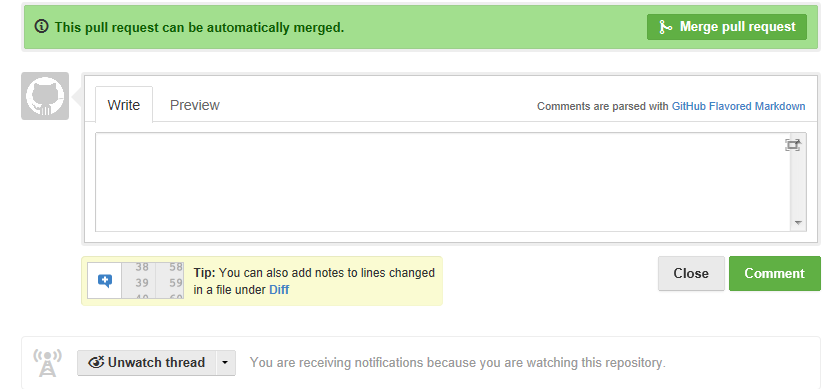
La richiesta arriva al repository base dell’utente **sytelUiip** che essendo l’owner del repository è responsabile del trattamento delle **pullRequest** inviate dai fork:



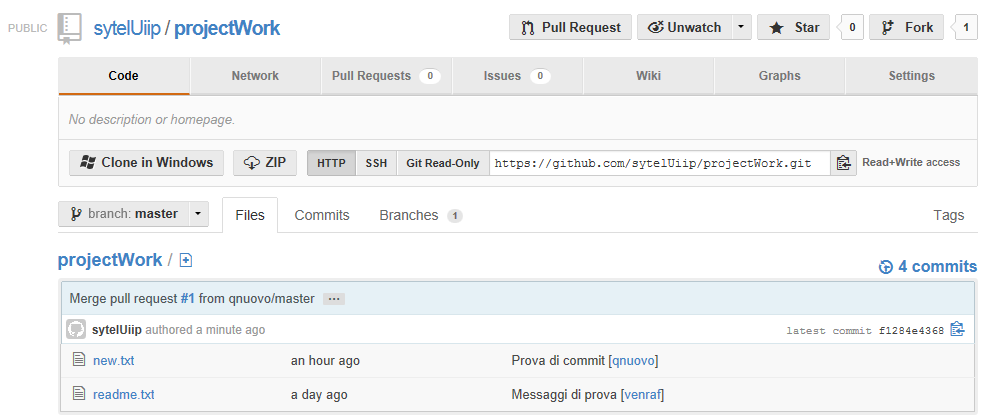
L’utente sytelUiip deve esaminare le richieste di pull aperte:



E decidere di accogliere le proposte di allineamento effettuando la **Merge pull request** o respingere la richieste effettuando la **close**:



Una volta effettuata la *merge* il repository base e il fork risultano allineati:



La realizzazione fisica delle Baseline di rilascio dovrà avviene sul Repository githup, dello specifico gruppo, in una cartella dedicata con la seguente struttura:

* **Rilasci**
* Baseline 1
* Baseline 2
* Baseline 3
* Baseline 4

Alla data di rilascio, indicata per ogni Baseline, verrà ‘congelato’ il contenuto per le opportune verifiche. Nei giorni successivi al rilascio verrà fornito un feedback che potrà essere affrontata anche negli incontri via Skype previsti una volta a settimana (da definire di volta in volta).

# Descrizione processi

I paragrafi successivi descrivono nel dettaglio le modalità operative e le soluzioni tecnologiche per soddisfare i requisiti utente.

## Accesso al sistema

Lato WebApplication si deve implementare una form di login che preveda due textbox per inserire username e password. Quando il giornalista effettua il login la WebApplication deve memorizzare l’account restituito dal web service nella propria session e per ogni funzionalità (ad esempio CreaNotizia, ModificaNotizia ecc.) deve controllare se il giornalista appartiene ad un gruppo al quale è stata associata la funzionalità. Nel caso in cui il giornalista non abbia il permesso l’applicazione non deve fargli eseguire la funzionalità e deve restituire un messaggio di errore. Se il giornalista non ha nessun account associato la WebApplication deve visualizzare il relativo messaggio di errore e ritornare alla pagina di login.

Si deve implementare un WebService che restituisca l’account associato ad una username e password con i gruppi e le funzionalità associate. Nel caso in cui il giornalista non abbia nessun account associato il WebService deve restituire un codice di errore alla WebApplication.

Per tutte le funzionalità di deve sempre controllare se il giornalista è abilitato.

### Lista Account

Se il giornalista ha un account associato al gruppo Amministratore deve avere la possibilità di visualizzare un link che porta alla pagina con la lista degli account presenti sul Sistema Editoriale.

Al riguardo si deve implementare un WebService che restituisca la lista degli account presenti sul Sistema Editoriale.

### Crea Account

Nella pagina della WebApplication dove viene presentata la lista degli account ci deve essere un button tramite il quale poter accedere alla pagina di creazione degli account.

Si deve implementare un WebService che permetta la creazione di un account da associare ad un giornalista.

La chiave dell’account deve essere formata da Nome e Cognome del giornalista, nel caso sia già presente il WebService dovrà restituire un codice di errore alla WebApplication.

### Modifica Account

Lato WebApplicatiuon partendo dalla lista degli account deve essere possibile poter clickare su ciascuno di essi per accedere ad una pagina dove è possibile modificare l’account selezionato.

Al riguardo si deve implementare un WebService che permetta la modifica di un account associato ad un giornalista. Deve essere possibile modificare tutti i dati di Account eccetto Nome e Cognome. Inoltre deve essere possibile cambiare il gruppo associato.

### Cancella Account

Lato WebApplication partendo dalla lista deve essere possibile cancellare l’account.

Si deve implementare un WebService che permetta la cancellazione di un account eliminando a cascata anche l’eventuale associazione ad un gruppo.

## Gestione Notizia On-Line

### Crea Notizia

Lato WebApplication si deve implementare una pagina con una form che consenta di inserire tutti i campi di una notizia.

La Notizia deve avere, una volta creata, obbligatoriamente i campi Id (generato univocamente in modo automatico dall’applicazione), Stato, Titolo, Sottotitolo, Autore, DataCreazione, Testo, LughezzaTesto.

Per il campo Stato di default si deve inserire il valore S (editabile) e per il campo Lock il valore N (disponibile). Nel campo Autore si deve inserire la sigla presa dall’account del giornalista. Nel campo DataCreazione si deve mettere la data in cui viene inserita la notizia nel sistema editoriale.

### Lista Notizie

Lato WebApplication si deve implementare una pagina che verrà restituita a valle del login e che presenterà tutte le notizie presenti nella sistema editoriale. Si deve implementare un WebService che restituisca la lista delle notizie presenti nella redazione.

Deve essere possibile poter applicare dei filtri sulla lista da restituire, sui campi Stato, Autore, Titolo. Nella form della WebApplication deve essere presente un menu a tendina nel quale poter selezionare uno dei tre campi sui quali filtrare e un campo di testo nel quale poter inserire il valore da ricercare per il campo selezionato. La ricerca deve essere fatta solo su uno dei tre campi.

La lista delle notizie deve essere ordinata in base alla DataCreazione in modalità decrescente.

I campi delle notizie da visualizzare nella lista sono Titolo, Autore, DataCreazione, DataTrasmissione, Stato, Lock, UltimoDigitatore.

### Modifica Notizia

Lato WebApplication a partire dalla lista delle notizie si deve fornire la possibilità di selezionarne una in stato editabile e senza lock da parte di nessun altro giornalista.

Nel caso in cui il giornalista cerchi di modificare una notizia che ha un lock da parte di un altro giornalista, l’invocazione verso il WebService non deve essere fatta. Nel caso in cui la sigla del giornalista sia uguale a quella di chi ha inserito il lock (ultimoDigitatore) si deve permettere la modifica.

Si deve implementare un web service che in base alla chiave della notizia ne restituisca tutti i campi e inserisca nel campo lock della Notizia il valore Y e nel campo ultimoDigitatore la sigla del giornalista.

Se nel campo lock è già presente il valore Y e il valore dell’ultimoDigitatore è diverso dalla sigla del giornalista, il WebService deve restituire un codice di errore alla WebApplication. Il controllo sul lock e sulla sigla dell’ultimoDigitatore deve essere effettuato sia lato WebApplication che lato WebService.

### Annulla

La funzionalità permette di annullare le modifiche che il giornalista ha apportato alla Notizia, ma che non ha ancora reso persistente nella base dati tramite la funzionalità Registra. Quando viene richiamata questa funzionalità si devono solo valorizzare i campi STATUS e LOCK della Notizia rispettivamente a S (Editabile) e a N (disponibile), lasciando il campo UltimoDigitatore valorizzato con la sigla dell’utente che ha preso in modifica la Notizia. Naturalmente va introdotto il controllo del Lock sulla notizia già menzionato nel paragrafo relativo alla funzionalità Modifica Notizia. La funzionalità Annulla deve essere presente sia nella form relativa alla Modifica della notizia che nella form relativa alla lista delle Notizie.

### Visualizza Notizia

La funzionalità è identica a quella della modifica, con la differenza che non viene gestito il lock sulla Notizia. È una funzionalità utile a visualizzare la Notizia in sola lettura.

### Registra Notizia

Lato WebApplication una volta modificati i campi della notizia, si deve fornire la possibilità di aggiornare i campi sul DataBase tramite la funzionalità Registra.

Si deve implementare un WebService che aggiorna tutti i campi della notizia e che inserisce nel campo lock il valore N, per rendere la notizia nuovamente a disposizione di tutti gli altri giornalisti. Nel campo ultimoDigitatore si deve lasciare la sigla del giornalista che l’aveva precedentemente lockata.

### Cancella Notizia

Lato WebApplication a partire dalla lista delle notizie deve essere possibile effettuare la cancellazione di notizie che non sono in modifica da parte di altri giornalisti.

Si deve implementare un WebService che effettui la cancellazione nel Sistema Editoriale della notizia. Si deve effettuare il controllo del lock e della sigla dell’ultimoDigitatore sia lato WebApplication che lato WebService.

La cancellazione della notizia è di tipo logico, inserendo nel campo stato il valore C che non rende più la notizia editabile, ma solo visualizzabile con la funzionalità Visualizza.

## Gestione Notizia Batch

### Trasmetti Notizia

Lato WebApplication a partire dalla lista delle notizie deve essere possibile effettuare la trasmissione di notizie che non sono in modifica da parte di altri giornalisti e che abbiano lo stato editabile.

Si deve implementare un WebService che effettui la trasmissione della notizia. Precedentemente si deve effettuare il controllo del lock e dello stato sia lato WebApplication che lato WebService.

La trasmissione della notizia è di tipo logico, va effettuata inserendo nel campo stato il valore Q (in trasmissione) che non rende più la notizia editabile, ma solo visualizzabile con la funzionalità Visualizza. Inoltre deve essere inserito nel campo dataTramissione il valore della data-ora corrente.

### Ricezione Notizie provenienti dai sistemi contributori (Fornitori)

Nel sistema editoriale si deve schedulare un processo (Receiver) che ogni 5 secondi controlla se nella directory ci sono file di testo XML che hanno una struttura che rispetta il formato in allegato. Una volta rilevato un file, crea la notizia nel DataBase con i campi che trova nel file.

Nella sigla dell’autore deve inserire la sigla del Receiver (al quale non è associato nessun account) RCV.



### Trasmissione Notizie dalla redazione verso i sistemi Clienti.

Nel sistema editoriale si deve schedulare un processo (Trasmitter) che ogni 5 secondi controlla sul DataBase se ci sono notizie da trasmettere (stato con valore Q). In caso positivo prende tutti valori della notizia e costruisce un XML che rispetta il formato dello schema allegato.

Una volta costruito l’XML, il task lo deve scrivere in un file di testo con la seguente nomenclatura: DataCreazione-Id-DataTrasmissione.xml. Il file va inserito in un a directory dove si presume ci possa essere in ascolto un task del cliente finale.

Una volta conclusa l’operazione di scrittura del file deve modificare lo stato della notizia appena trasmessa al valore T.



La DataCreazione e la DataTrasmissione nel nome del file devono rispettare il seguente formato:

* YYYYMMDDhh24mmss (AnnoMeseGiornoOraMinutiSecondi).

# Requisiti non funzionali

## Gestione dei messaggi e dei codici di errore.

Quando si verificano errori di tipo applicativo, come quando si tenta di prendere in modifica una notizia già presa in carico da un altro giornalista o non si ha il permesso di eseguire determinate funzionalità non appartenendo al gruppo che le detiene, i WebService devono restituire dei codici di errore (numeri interi). Questa soluzione permette l’internazionalizzazione del sistema editoriale, cioè la lingua dei messaggi che devono essere mostrati deve essere configurabile.

## Scalabilità del Sistema Editoriale.

Il sistema editoriale verrà in prima istanza utilizzato da un redazione di 10 giornalisti, successivamente verrà rilasciato a tutte le redazioni di un’agenzia giornalistica. Quindi il sistema deve essere scalabile e prevedere dei tempi di risposta verso giornalisti non superiori di 2 secondi

## Logging

Tutti i layer applicativi devono tracciare le proprie attività su dei files di log giornalieri. Si consiglia di utilizzare la libreria Apache Log4J.

# Documentazione da produrre

## Procedura di installazione e setup

Deve essere realizzato un documento di installazione dettagliato, in modo da poter essere utilizzato da personale non tecnico per l’installazione e la configurazione iniziale del sistema editoriale.

Il documento deve descrivere in dettaglio:

* Caratteristiche HW necessarie (HW, SO, memoria, etc.)
* Software terze parti da installare (Oracle, etc.)
* Procedura di installazione
* Procedura di setup iniziale
* Start e stop del sistema