Norme di Progetto

Gruppo JurassicSWE · Progetto IronWorks

JurassicSWE@gmail.com

Informazioni sul documento

Versione	2.0.0
Redazione	Gruppo JurassicSWE
Verifica	Marco Masiero
Approvazione	Leo Moz
Uso	Interno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo JurassicSWE

Sommario

Tale documento descrive le regole, gli strumenti e le convenzioni che il gruppo JurassicSWE dovrà rispettare durante tutta la realizzazione del progetto IronWorks.

Registro delle modifiche

Versione	Data	Ruolo	Nominativo	Descrizione
2.0.0	2018-04-27	Responsabile	Leo Moz	Approvazione del documento
1.1.0	2018-04-26	Verificatore	Marco Masiero	Verifica del documento
1.0.4	2018-04-26	Amministratore	Lidia Alecci	Stesura da § 3.2.3.4 a 3.2.3.6
1.0.3	2018-04-26	Amministratore	Francesco Minna	Stesura § 3.2.3.1
1.0.2	2018-04-26	Amministratore	Lidia Alecci	Stesura § 3.1.3.5
1.0.1	2018-04-24	Amministratore	Daniele Dal Maso	Modifica struttura
1.0.0	2018-03-14	Responsabile	Francesco Minna	Approvazione del documento
0.2.0	2018-04-14	Verificatore	Gianluca Travasci	Verifica del documento
0.1.0	2018-04-13	Verificatore	Lidia Alecci	Verifica del documento
0.0.16	2018-03-13	Amministratore	Leo Moz	Stesura § 2.2.6.1 e 2.2.6.3
0.0.15	2018-03-13	Amministratore	Lidia Alecci	Stesura § 4.1
0.0.14	2018-03-12	Amministratore	Leo Moz	Modifica della § 3.5.2
0.0.13	2018-03-12	Amministratore	Leo Moz	Modifica della § 1.2
0.0.12	2018-03-12	Amministratore	Leo Moz	Modifica delle § 1.1, 1.4, 2.1.3, 3.3, 3.5
0.0.11	2018-03-11	Amministratore	Leo Moz	Migliorie al template
0.0.10	2018-03-11	Responsabile	Francesco Minna	Migliorie al template
0.0.9	2018-03-10	Amministratore	Daniele Dal Maso	Modifica § 3
0.0.8	2018-03-9	Amministratore	Marco Masiero	Stesura § 2.1
0.0.7	2018-03-8	Amministratore	Daniele Dal Maso	Stesura § 3.4 e 3.5
0.0.6	2018-03-8	Amministratore	Daniele Dal Maso	Stesura § 3.1, 3.2 e 3.3
0.0.5	2018-03-7	Responsabile	Gianluca Travasci	Stesura § 5.1.1 e 5.1.2
0.0.4	2018-03-6	Verificatore	Gianluca Travasci	Stesura § 5.1.4
0.0.3	2018-03-6	Responsabile	Gianluca Travasci	Stesura § 5.1.3
0.0.2	2018-03-5	Responsabile	Gianluca Travasci	Stesura § Introduzione
0.0.1	2018-03-5	Responsabile	Francesco Minna	Creazione del template e stesura dell'indice.

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di definire le regole, gli strumenti e le convenzioni adottate dal gruppo JurassicSWE durante l'intero svolgimento del progetto. In quest'ottica, questo documento deve essere visionato da tutti i componenti del gruppo, i quali sono tenuti ad applicare quanto scritto al fine di mantenere omogeneità e coesione.
In caso di modifiche o aggiunte al presente documento è obbligatorio informare ogni membro del gruppo.
Si fa notare la natura incrementale del documento e come quindi esso non definisca una versione finale.

1.2 Scopo del prodotto

Lo scopo del *prodotto*$_g$ è quello di realizzare un software, in particolare un'*applicazione web*$_g$, per disegnare diagrammi *UML*$_g$ di robustezza, e di un generatore di codice che, a partire dalle definizioni contenute in un diagramma, produca il codice di classi *Java*$_g$ per ospitare i dati delle entità persistenti, ed i metodi per leggere e scrivere questi dati in un *database relazionale*$_g$.

1.3 Glossario

Al fine di evitare ambiguità, i termini che possono essere interpretati in modi diversi a seconda del contesto, o che necessitano di una descrizione approfondita, sono scritti in *corsivo* con una "g" pedice, solo alla loro prima occorrenza.
La definizione di tali termini è contenuta nel documento *Glossario v2.0.0* .

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Standard ISO/IEC 12207:1995**
  http://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf (consultato il 2018-03-15);
- **Capitolato:**
  http://www.math.unipd.it/~tullio/IS-1/2017/Progetto/C5.pdf (consultato il 2018-03-15).

1.4.2 Riferimenti informativi

- **Documentazione - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L12.pdf (consultato il 2018-03-15);
- **Processi Software - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L03.pdf (consultato il 2018-03-15);
- **Gestione di Progetto - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L06.pdf (consultato il 2018-03-15);
- **Informazioni su ShareLaTeX**
  https://www.sharelatex.com/ (consultato il 2018-03-15);
- **Documentazione su Git**
  http://www.piratpartiet.it/mediawiki/index.php?title=Mini_Guida_a_GIT#Comandi_di_base_a_riga_di_comando (consultato il 2018-03-15)
  https://support.gitkraken.com/release-notes/current (consultato il 2018-03-15);
- **Esiti RR:** *Esiti Revisione dei Requisiti del gruppo JurassicSWE*
  http://www.math.unipd.it/~tullio/IS-1/2017/Progetto/RR/JurassicSWE.pdf (consultato il 2018-04-27).

2 Processi primari

2.1 Fornitura

2.1.1 Scopo del processo

Il *processo*$_g$ di fornitura contiene tutte le attività e i compiti del fornitore. Esso può essere avviato sia dalla decisione di preparare una risposta ad una richiesta del *Proponente*$_g$, sia con la stipulazione di un contratto con l'acquirente per la consegna del prodotto.
Il processo continua con la determinazione delle procedure e delle risorse necessarie per l'organizzazione ed il completamento del progetto, incluso lo sviluppo di un *Piano di Progetto* e l'esecuzione di questo fino alla consegna del materiale prodotto all'acquirente.
Il processo di fornitura è composto dalle seguenti attività:

- avvio;
- preparazione della risposta;
- contrattazione;
- pianificazione;
- esecuzione e controllo;
- revisione e valutazione;
- consegna e completamento.

2.1.2 Descrizione

Questa sezione tratta le norme che i membri del gruppo JurassicSWE sono tenuti a rispettare nelle fasi di progettazione, sviluppo e consegna del prodotto IronWorks, al fine di proporsi e diventare fornitori nei confronti del Proponente Zucchetti S.p.A. e dei *Committenti*$_g$ Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.1.3 Attività

2.1.3.1 Studio di fattibilità

È compito del Responsabile di progetto organizzare riunioni preventive tra i membri del gruppo al fine di permettere lo scambio di opinioni sui capitolati proposti.
Il documento *Studio di Fattibilità v2.0.0* è redatto dall'*Analista*$_g$ e, per ogni *capitolato*$_g$, indica:

- **Descrizione generale**: breve presentazione del progetto, descrizione delle caratteristiche principali richieste nel prodotto ed il suo ambito di utilizzo;

- **Dominio tecnologico e applicativo**: si valuta il capitolato prendendo in considerazione l'ambito in cui il progetto si colloca e le tecnologie richieste;
- **Aspetti positivi e critici**: considerazioni del gruppo sugli aspetti positivi e i fattori di rischio nella scelta del capitolato corrente;
- **Valutazione finale**: riassunto degli aspetti principali che hanno portato il gruppo ad accettare o scartare il capitolato in questione.

2.1.3.2 Piano di progetto

Il Responsabile, aiutato dagli Amministratori, dovrà redigere un piano da seguire per tutta la durata del progetto.
Il documento conterrà:

- **Analisi dei rischi**: si analizzano in dettaglio i principali rischi che potrebbero insorgere durante il progetto e si fornisce una possibile soluzione ad essi. Verrà inoltre fornita la probabilità che essi occorrano e il livello di gravità ad essi associato;
- **Modello di sviluppo**: viene descritto il modello di sviluppo scelto dal gruppo JurassicSWE per realizzare il progetto IronWorks;
- **Pianificazione**: si pianificano le attività da svolgere nelle varie fasi del progetto e si forniscono le scadenze temporali ad esse associate;
- ***Preventivo***$_g$: in base a quanto stabilito nella pianificazione si stima la quantità di lavoro necessaria per ogni fase, proponendo così un preventivo per il costo totale del progetto;
- **Organigramma**

2.1.3.3 Piano di qualifica

I Verificatori dovranno redigere un piano contenente le strategie da adottare per la verifica e la validazione del materiale prodotto dal gruppo JurassicSWE .
Il documento conterrà:

- **Strategia di gestione della qualità**: si definiscono gli obiettivi di qualità di processo e di prodotto e vengono individuate le *metriche*$_g$ e le relative misure;
- **Gestione amministrativa della revisione**: si stabiliscono gli obiettivi da raggiungere per la qualità di processo e di prodotto, assegnando dei range alle metriche;
- **Standard di qualità**: vengono descritti nel dettaglio gli standard di qualità scelti dal gruppo;
- **Valutazioni per il miglioramento**: contiene tre tabelle in cui rispettivamente vengono riportati i problemi e le relative soluzioni nell'organizzazione, nel ricoprire un determinato ruolo e nell'uso degli strumenti scelti;

- **Resoconto delle attività di verifica**: per ogni attività si devono riportare le metriche calcolate e un resoconto sulla verifica di tale attività.

2.1.4 Rapporti di fornitura con il Proponente

Durante l'intero svolgimento del progetto il gruppo intende instaurare con l'azienda proponente Zucchetti S.p.A., nello specifico con il referente Dr. Gregorio Piccoli, un rapporto di collaborazione costante e costruttiva al fine di:

- determinare i bisogni del Proponente;
- specificare come saranno definiti ed eseguiti i processi;
- stimare i costi;
- accordarsi circa la qualifica di prodotto.

2.1.5 Documentazione fornita

Vengono qui elencati i documenti forniti a Zucchetti S.p.A. e ai Committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin, al fine di assicurare la massima trasparenza circa le attività di:

- **Pianificazione, consegna e completamento**: descritte all'interno del *Piano di Progetto v2.0.0* ;
- **Analisi**: l'analisi dei requisiti e i *casi d'uso*$_g$ sono specificati in *Analisi dei Requisiti v2.0.0* ;
- **Verifica e validazione**: tutte le procedure atte a garantire la qualità dei processi e di prodotto sono descritte e definite nel documento *Piano di Qualifica v2.0.0* ;
- **Glossario**: i termini di tutti i documenti redatti dal gruppo che possono generare ambiguità si trovano nel *Glossario v2.0.0* con la relativa descrizione.

2.1.6 Collaudo e consegna del Prodotto

Al fine di consegnare il prodotto terminato, il gruppo deve effettuare un *collaudo*$_g$ in presenza del Proponente e dei Committenti.
Precedentemente a questo test, il gruppo deve assicurare correttezza, completezza e affidabilità di ogni parte del materiale realizzato, affinché si possa dimostrare che:

- tutti i requisiti obbligatori descritti nel documento *Analisi dei Requisiti v2.0.0* siano completamente soddisfatti;
- l'esecuzione di tutti i test descritti nel documento *Piano di Qualifica v2.0.0* abbiano esito positivo.

In seguito al collaudo finale, il Responsabile di progetto presenta il consuntivo al *Committente*$_g$ e consegna il prodotto su supporto fisico.

2.2 Sviluppo

2.2.1 Scopo del processo

Questo processo contiene l'insieme di attività e compiti svolti dal team, al fine di produrre il software finale richiesto dal proponente, conforme ad ogni Sua richiesta.

2.2.2 Descrizione

Il processo di sviluppo segue le norme definite dallo standard *ISO/IEC 12207*$_g$, pertanto si compone delle seguenti fasi:

- **Analisi dei requisiti**;
- **Progettazione**;
- **Codifica**.

2.2.3 Attività

2.2.3.1 Analisi dei requisiti

È compito degli Analisti redigere il documento *Analisi dei Requisiti*, che comprenderà la lista di tutti i requisiti individuati dall'analisi del capitolato e da altre fonti, come gli incontri con Proponente e Committente.
Ogni requisito dovrà essere meno ambiguo possibile e rispettare le norme sotto riportate.

2.2.3.1.1 Requisiti

Rappresentiamo ogni requisito con il seguente formalismo:

R[Tipo][Utilità][Identificatore_principale].[Identificatore_di_livello]

dove:

- **Tipo**: indica se il requisito è di tipo
  - **F**: funzionale;
  - **P**: prestazionale;
  - **Q**: qualitativo;
  - **P**: progettuale.
- **Utilità**: può assumere i seguenti valori
  - **O**: obbligatorio (irrinunciabile per qualcuno degli *Stakeholder*$_g$);

  - **F**: opzionale (relativamente utile oppure contrattabile più avanti nel progetto);
  - **D**: desiderabile (non strettamente necessario ma ha valore aggiunto riconoscibile).
- **Identificatore_principale**: numero per disambiguare più requisiti dello stesso *Tipo* ed *Utilità*;
- **Identificatore_di_livello**: un numero (preceduto da un punto ".") per disambiguare i sotto-requisiti; può essere seguito da un altro Identificatore_di_livello.

Esempio: RFD3.2.1 identifica il primo figlio del secondo sotto-caso del terzo Requisito Funzionale Desiderabile.

2.2.3.1.2 Casi d'uso

La tecnica di ricerca e analisi dei requisiti viene fatta per mezzo dei *casi d'uso*$_g$.
Ogni caso d'uso sarà rappresentato dal seguente formalismo:

UC[Codice]

dove:
Codice: è un numero che identifica in modo univoco i casi d'uso.

Ogni caso d'uso, inoltre, deve essere descritto secondo la seguente specifica:

- **ID**: è il codice identificativo del caso d'uso, di cui il formalismo sopra;
- **Nome**: titolo sintetico del caso d'uso;
- **Attori**: gli attori coinvolti, sia principali sia secondari;
- **Descrizione**: breve descrizione del caso d'uso;
- **Precondizioni**: lo stato in cui si deve trovare il sistema affinché si verifichino le post-condizioni;
- **Postcondizioni**: le istruzioni che risultano vere dopo il verificarsi degli eventi del caso d'uso;
- **Scenario principale**: rappresenta il flusso principale degli eventi come lista numerata di azioni tra l'utente e il sistema;
- **Scenari alternativi**: casi d'uso esterni al flusso principale degli eventi che servono a gestire eventuali eccezioni o errori.

Gli oggetti principali del *diagramma di robustezza*$_g$, ossia *Actor*$_g$, *Control*$_g$, *Boundary*$_g$, *Entity*$_g$, Commenti e *Relazioni*$_g$, potranno essere inclusi nel termine generico "oggetto" e avranno l'iniziale del nome in maiuscolo.

2.2.3.2 Progettazione

Questa fase segue quella di Analisi e ha come scopo la ricerca di una soluzione al problema che soddisfi le richieste e i vincoli di tutti gli Stakeholder.
Gli obiettivi della progettazione sono:

- soddisfare i requisiti con efficacia, usando un sistema di qualità;
- identificare un buon schema architetturale utile al caso e con parti riusabili;
- dominare la complessità del sistema, suddividendolo in parti più piccole (e.g. *Divide et Impera*$_g$).

A tal proposito verranno svolte le seguenti attività:

- ***Technology Baseline***$_g$: selezione delle tecnologie, *framework*$_g$ e librerie tramite *Proof of Concept*$_g$ (l'abbozzo di un dimostratore funzionante, che evidenzi come la tecnologia selezionata possa servire efficacemente allo sviluppo del prodotto atteso);
- ***Product Baseline***$_g$: baseline architetturale, coerente con quanto mostrato in Technology Baseline, viene presentata tramite diagrammi delle classi e di sequenza. L'analisi è comprensiva di *design pattern*$_g$ contestualizzati all'*architettura*$_g$.

Sarà compito del *Progettista*$_g$ scrivere tale documentazione.

2.2.3.3 Codifica

Questa fase consiste nell'implementazione delle soluzioni al problema emerso dalle fasi precedenti. Attraverso la programmazione si ottiene il prodotto software richiesto che, oltre ad essere conforme ai requisiti, deve avere le qualità di modificabilità e leggibilità.
Per conseguire tali qualità sono state fissate alcune norme e convenzioni che ciascun membro del gruppo è tenuto a rispettare.

2.2.3.3.1 Intestazione

Ogni *file*$_g$ contenente codice deve avere la seguente intestazione:

```
/*
* File: nome del file
* Versione: versione del file
* Tipo: tipo di file
* Data: data di creazione
* Autore: nome autore/i
* E-mail: email autore/i
*
* Licenza: tipo di licenza
*
* Avvertenze: lista delle avvertenze e limitazioni
```

```
*
* Registro modifiche:
* Autore | Descrizione | Data ultima modifica
*
*/
```

2.2.3.3.2 Nomenclatura

Le convenzioni sui nomi rendono il codice più omogeneo e comprensibile facilitandone la lettura.

Nome	Regole	Esempi
Classi	I nomi della classi devono essere sostantivi, con la prima lettera di ogni parola maiuscola (notazione *CamelCase*$_g$); cercare di usare nomi semplici e descrittivi.	class Raster; class ImageSprite;
Interfacce	I nomi delle interfacce vanno scritti con la stessa convenzione usata per i nomi delle classi.	interface RasterDelegate; interface Storing;

Metodi	I nomi dei metodi dovrebbero essere dei verbi, con la prima lettera in minuscolo, e la prima lettera di ogni parola successiva in maiuscolo (notazione camel case, con la prima lettera minuscola).	draw();
		drawLine();
		getBackground();

Variabili	I nomi delle variabili, dovrebbero essere brevi, significativi e "mnemonici", cioè atti a veicolare ad un osservatore esterno l'intento del loro uso. Nomi che abbiano un solo carattere andrebbero evitati, tranne per le variabili temporanee "usa e getta". La notazione adottata è la stessa dei metodi: prima lettera in minuscolo, e prima lettera di ogni parola successiva in maiuscolo (notazione CamelCase, con la prima lettera minuscola).	result = f(); heightValue = max(a,b);

Costanti	Le costanti vanno scritte in maiuscolo (all caps), separando ogni parola con un trattino basso (underscore).	int `IN_WIDTH` = 4;
		int `MAX_WIDTH` = 999;
		int `GET_THE_CPU` = 1;

2.2.3.3.3 Formattazione

- **Indentazione**: è richiesto l'utilizzo di esattamente una tabulazione. Le tabulazioni devono essere impostate esattamente a 4 spazi;
- **Parentesi**: generalmente è una buona norma usare le parentesi nelle espressioni che coinvolgono più operatori, per evitare problemi di precedenza sugli operatori stessi;
- **Classi e interfacce**: alcune regole di formattazione:
  - nessuno spazio tra il nome di un metodo e la parentesi "(" di inizio elenco parametri;
  - la parentesi "{" va messa alla fine della stessa riga dell'istruzione di dichiarazione;
  - la parentesi "}" va messa in una nuova riga alla fine del blocco di istruzioni corrispondente. Nel caso in cui il blocco sia vuoto, va posta subito dopo l'apertura "{";
  - i metodi sono separati con una riga vuota.

2.2.3.3.4 Commenti

Il codice va opportunamente commentato, per far si che sia leggibile anche a distanza di tempo e da altre persone diverse dall'autore.
Molti linguaggi (come Java e JavaScript) offrono due tipi di commenti: commenti di implementazione e commenti di documentazione:

- i **commenti di implementazione** sono quelli che troviamo anche in *C++*$_g$, delimitati da /*...*/ e //. Sono utili per commentare il codice o per descrivere una particolare implementazione;
- i **commenti di documentazione** sono delimitati da /**...*/, possono essere estratti in file *HTML*$_g$ usando opportuni strumenti (come *Javadoc*$_g$ e *JSDoc*$_g$). Hanno lo scopo di descrivere le specifiche del codice, per essere letti da sviluppatori che potrebbero non necessariamente avere il *codice sorgente*$_g$ a portata di mano.

I commenti dovrebbero essere utilizzati per mostrare una panoramica del codice e fornire ulteriori informazioni non facilmente disponibili nel codice stesso; inoltre dovrebbero contenere solo le informazioni che sono rilevanti per la lettura e la comprensione del programma.
Ad esempio, informazioni su come un dato pacchetto è compilato o in quale directory risiede non dovrebbero essere incluse come commento.

2.2.3.3.5 Ricorsione

Preferiamo evitare quanto più possibile codice ricorsivo a favore di quello iterativo, in quanto potrebbe portare a una maggiore occupazione di memoria e risultare poco efficiente.

2.2.4 Strumenti utili

Di seguito sono elencati gli strumenti che saranno adoperati da ciascun membro del gruppo durante il processo di sviluppo.

2.2.4.1 Astah

Come editor di diagrammi UML si è scelto di usare *Astah Professional*$_g$, un software per la modellazione UML pensato per team di sviluppo *Agile*$_g$. Oltre a supportare i diagrammi di robustezza, Astah offre potenti strumenti per la codifica automatica dal linguaggio UML a codice di programmazione come per esempio Java o C++.
Attraverso apposite funzionalità è possibile poi collegare più file e consentire a più utenti di lavorare contemporaneamente sullo stesso diagramma. Quando viene apportata una modifica in un diagramma, la modifica diventa visibile negli altri diagrammi di tutti gli altri utenti. Ciò permette di rendere la modellazione collaborativa più semplice ed efficace.

http://www.astah.net/ (consultato il 2018-03-20)

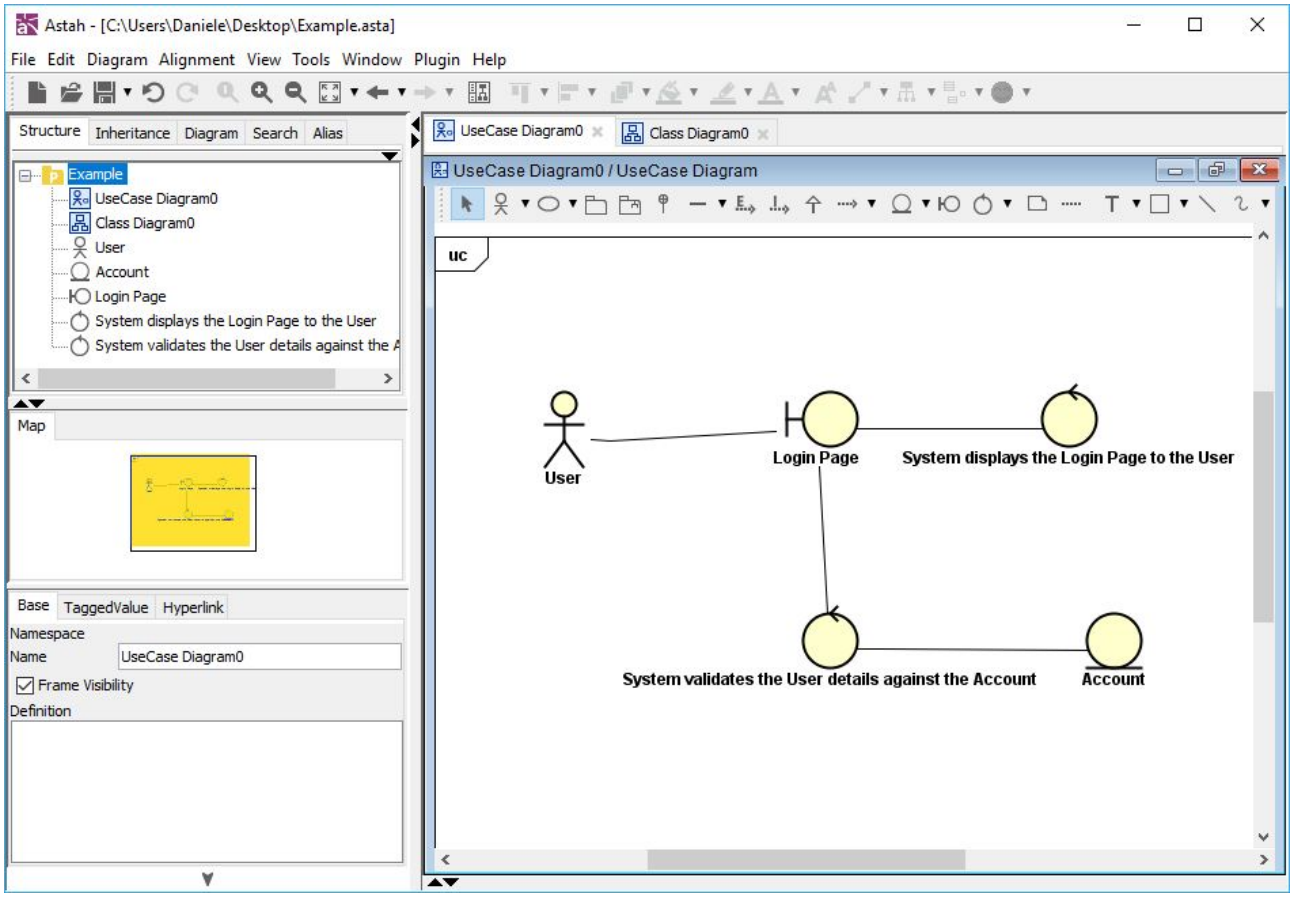

Figura 1: Astah Professional per Windows

2.2.4.2 PragmaDB

Per gestire i risultati dell'*Analisi dei Requisiti* si è scelto di usare una versione modificata ed estesa di *PragmaDB*$_g$, un applicativo web che permette, tra le altre cose, l'inserimento e il tracciamento di Fonti, Requisiti, Casi d'Uso, Attori, Classi e Test.
Inoltre offre altre utili funzionalità, tra cui l'esportazione direttamente in LaTeX dei dati inseriti e di tabelle per il loro tracciamento, la generazione dei Diagrammi UML dei Casi d'Uso e delle Classi e il calcolo di alcune metriche. Tutti i membri possono accedervi effettuando il login con le proprie credenziali.

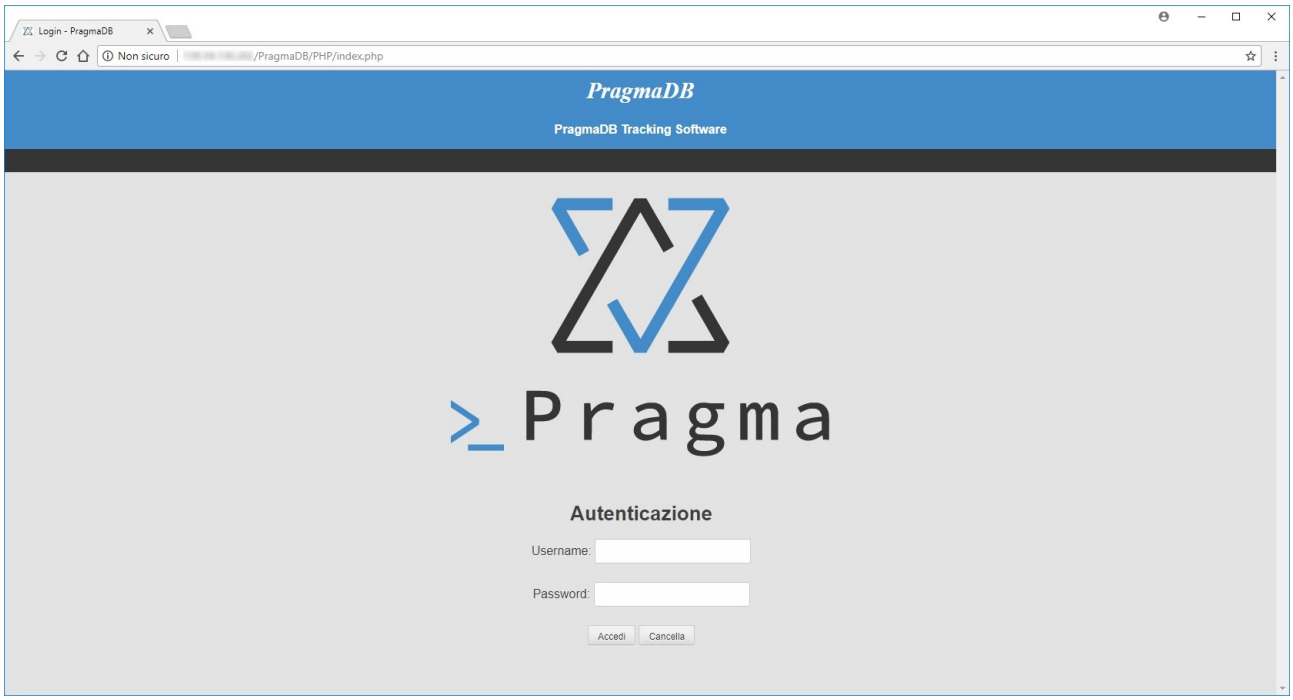

Figura 2: Homepage di PragmaDB

2.2.4.3 IntelliJ IDEA

Per la produzione del codice *JavaScript*$_g$ viene utilizzato *Intellij IDEA*$_g$ nella versione per studenti. Questo è un potente *IDE*$_g$ multi-piattaforma con molte funzionalità integrate e plug-in per ogni esigenza.

https://www.jetbrains.com/idea/ (consultato il 2018-03-20)

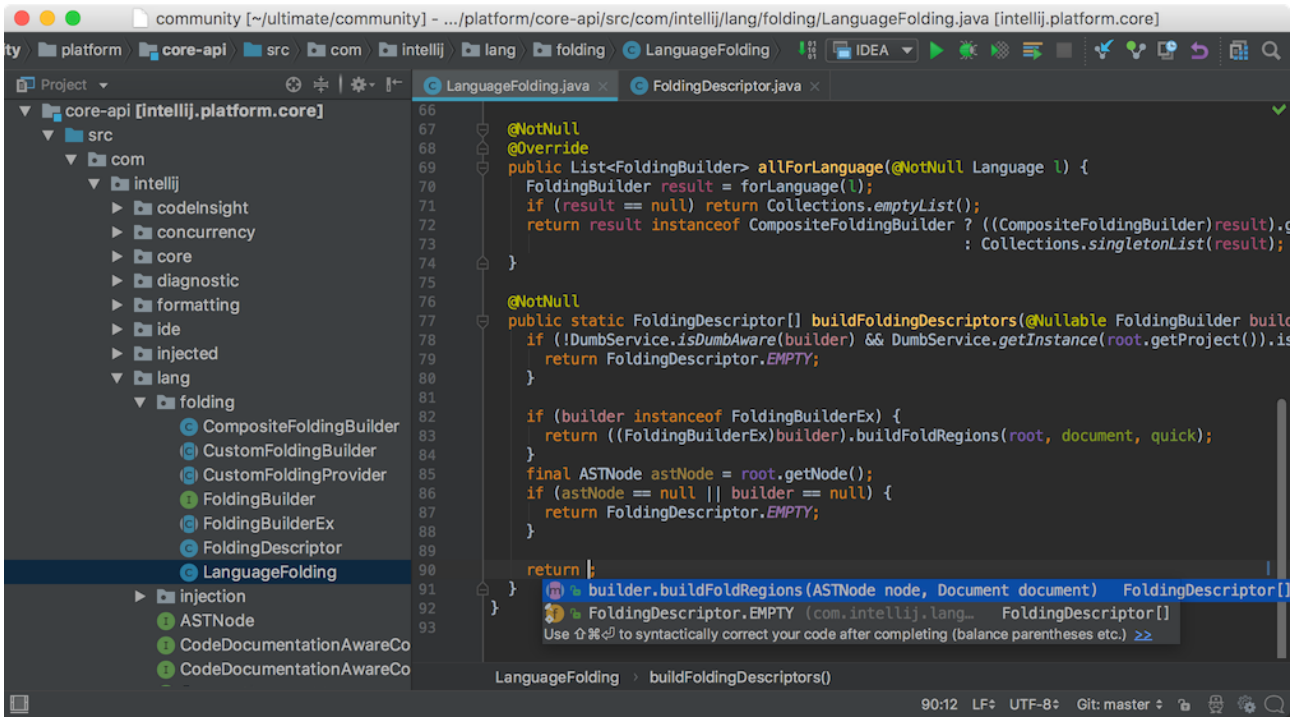

Figura 3: IntelliJ IDEA

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo del processo

Il processo di documentazione ha lo scopo di registrare le informazioni prodotte dai processi durante il *ciclo di sviluppo*$_g$ del software.
La documentazione serve per gestire lo svolgimento dei processi produttivi e la loro qualità, e per garantire la giusta applicazione delle regole dettate dal modello di sviluppo adottato.

3.1.2 Descrizione

In questa sezione sono indicate tutte le norme e le convenzioni scelte ed usate dal gruppo, per permettere una stesura di documenti validi e coerenti.

3.1.3 Attività

3.1.3.1 Struttura dei documenti

In questa sezione sono contenute le regole per la stesure della documentazione.

3.1.3.1.1 Nome documento

Il nome del documento, se non è un verbale, è scritto seguendo queste regole:

[Nome Documento]_[Versione]

dove:

- **Nome Documento**: ogni iniziale di parola è scritto in maiuscolo e non ci sono spazi nel nome;
- **Versione**: la versione è scritta come descritto nella sezione 3.1.3.2.

3.1.3.1.2 Versionamento

Ogni documento, fatta eccezione dei verbali, è soggetto a versionamento per il tracciamento delle modifiche effettuate durante lo sviluppo del progetto.
Ad ogni modifica corrisponde un avanzamento di versione, rispettando le seguenti regole:

v[X].[Y].[Z]

dove:

- **X**:
  - inizia da 0;
  - viene incrementato dal Responsabile di progetto dopo l'approvazione del documento;
  - quando viene incrementato, Y ritorna al valore iniziale 0.
- **Y**:
  - inizia da 0;
  - viene incrementato dal *Verificatore*$_g$ ad ogni verifica;
  - quando viene incrementato, Z viene riportato al valore iniziale 0.
- **Z**:
  - inizia da 0;
  - viene incrementato dal *Redattore*$_g$ del documento ad ogni modifica;

3.1.3.1.3 Prima pagina

La prima pagina di ogni documento è strutturata nel seguente ordine:

- **Logo del gruppo**: situato in alto e centrato orizzontalmente;
- **Nome del documento**: inserito sotto il logo, centrato orizzontalmente e scritto in grassetto;
- **Gruppo e progetto**: sotto al titolo si trova il nome del gruppo e il nome del progetto separati da un punto;
- **Recapito**: indirizzo di posta elettronica del gruppo centrato orizzontalmente appena sotto il nome del gruppo e del progetto;
- **Tabella delle informazioni**: sotto il recapito si trova invece una tabella contenente le seguenti informazioni:
  - versione del documento, se il documento non è un verbale;
  - nome e cognome dei membri del gruppo incaricati della stesura del documento;
  - nome e cognome dei membri del gruppo incaricati della verifica del documento;
  - nome e cognome dei membri del gruppo incaricati dell'approvazione del documento;
  - uso a cui è destinato il documento (interno o esterno);
  - destinatari del documento.
- **Sommario**: sintetica descrizione del documento, centrata orizzontalmente e inserita verso la fine della pagina.

3.1.3.1.4 Registro delle modifiche

La seconda pagina di ogni documento, ad eccezione dei verbali, è destinata al registro delle modifiche, scritto in forma tabellare.
Il registro è composto dalle seguenti voci, nell'ordine (da sinistra verso destra):

- **Versione**: riporta il numero di versione sul quale è stata effettuata la modifica; ogni modifica corrisponde ad un aumento di versione, seguendo le regole descritte nella sezione 3.1.3.2;
- **Data**: data in cui è avvenuta la modifica, scritta nel formato descritto nella sezione 3.1.5.2;
- **Ruolo**: ruolo ricoperto in quel momento dalla persona che ha effettuato la modifica;
- **Nominativo**: nome e cognome della persona che ha effettuato la modifica;
- **Descrizione**: breve e concisa descrizione delle modifiche apportate. Tali descrizioni saranno caratterizzati dalle seguenti voci:
  - Creazione template e stesura dell'indice: è la prima operazione da fare per creare il template LaTeXdel documento e l'indice dello stesso;
  - Migliorie al template: modifiche strutturali al documento;
  - Stesura sezione/Appendice/caso d'uso: si userà qualora venga scritta una sezione/Appendice/caso d'uso del documento in esame;
  - Aggiunta termini: solo per il *Glossario*, quando un membro del gruppo aggiunge un nuovo o più termini, si userà tale descrizione;
  - Modifica della sezione: si userà qualora venga modificata una sezione del documento in esame precedentemente scritta;
  - Eliminazione sezione: si userà qualora venga eliminata una sezione del documento in esame;
  - Verifica del documento: si userà quando il documento verrà verificato e incrementato nel suo versionamento;
  - Approvazione del documento: si userà quando il documento verrà approvato e incrementato nel suo versionamento.

3.1.3.1.5 Indice, elenco delle figure e elenco delle tabelle

Ogni documento, tranne i verbali, deve avere un indice, per permettere una veloce e facile consultazione tramite una lettura ipertestuale e non necessariamente sequenziale. Il numero delle varie sezioni segue le regole descritte nella sezione 3.1.3.6.
Se sono presenti immagini o tabelle, dovranno essere inserite in un indice specifico.

3.1.3.1.6 Numerazione paragrafi

Ogni paragrafo deve appartenere a una sezione e rispetta le seguenti regole:

- tutte le sezioni sono indicate da un numero incrementale che parte da 1;
- in base al grado di annidamento si aggiungerà sempre un punto seguito dal numero del nuovo sotto paragrafo in maniera iterativa e seguendo le regole descritte sopra.

3.1.3.1.7 Intestazione e piè di pagina

Ogni documento sarà caratterizzato da intestazione e piè di pagina.
L'intestazione sarà così suddivisa:

- logo del gruppo posizionato a sinistra;
- nome del documento e versione corrente a destra.

Il piè di pagina sarà così suddiviso:

- indirizzo email del gruppo a sinistra;
- numerazione progressiva della pagina rispetto al totale a destra.

3.1.3.2 Struttura verbali

Ogni verbale, sia interno che esterno, è formato da due sezioni principali.

- **Informazioni**:
  - **Informazioni generali**:
    - data;
    - luogo;
    - ora inizio;
    - ora fine.
  - **Partecipanti**: elenco che descrive i partecipanti;
  - **Argomenti**: breve riassunto degli argomenti trattati.
- **Riepilogo delle decisioni**: tabella con due colonne: la prima contenente il codice univoco, e l'altra contenente una breve descrizione della decisione presa.

I verbali esterni contengono altre due sezioni, rispettivamente prima della sezione "Riepilogo delle decisioni":

- **Domande e risposte**: elenco numerato di domande fatte al Proponente, con le rispettive risposte;

- **Consigli e suggerimenti del Proponente**: può non essere presente; contiene un elenco numerato degli eventuali consigli che il Proponente dà al gruppo.

3.1.3.3 Norme tipografiche

3.1.3.3.1 Acronimi

In tutti i documenti redatti dal gruppo, le uniche parole che è consentito scrivere interamente in maiuscolo sono gli acronimi.

3.1.3.3.2 Elenchi puntati

Gli elenchi puntati servono ad esprimere un concetto in modo sintetico e strutturato, evitando di utilizzare uno stile troppo narrativo, che mal si adatta in documenti di tipo tecnico-informativo.
Ogni voce nell'elenco puntato inizia con la lettera minuscola (tranne dove è necessario la lettera maiuscola, per esempio con nomi di attività) e finisce con un punto e virgola, ad eccezione dell'ultima, che si conclude con un punto.
Se l'elenco ha l'obiettivo di descrivere punti salienti, allora il nome del termine va scritto in grassetto, con la prima lettera maiuscola e se è presente la relativa descrizione va inserita dopo i due punti.
Distinguiamo due tipi di elenchi puntati:
Gli **elenchi puntati non numerati** sono rappresentati graficamente da:

- un pallino nel primo livello;
  - un trattino nel secondo livello;
    - un asterisco nel terzo.

Si eviterà di scendere oltre il quarto livello per mantenere una buona leggibilità.
Mentre gli **elenchi puntati numerati** saranno graficamente rappresentati da:

1. un numero arabo nel primo livello
   (a) una lettera nel secondo livello
      i. un numero romano nel terzo livello

Come per le liste non numerate si eviterà di scendere oltre il quarto livello per mantenere una buona leggibilità.
Inoltre è permesso anche l'annidamento di elenchi puntati non numerati in elenchi puntati numerati e viceversa, ove il redattore del documento lo ritiene necessario e utile.

3.1.3.3.3 Formato data

La data deve essere scritta nel seguente formato:

[AAAA]-[MM]-[DD]

dove:

- **AAAA**: rappresenta l'anno usando quattro cifre;
- **MM**: rappresenta il mese usando due cifre: da 01 a 12;
- **DD**: rappresenta il giorno usando due cifre: da 01 a 31.

La rappresentazione appena illustrata segue lo standard internet date come descritto nell'*ISO 8601*$_g$. Inoltre, ciò permette un corretto ordinamento alfanumerico.

3.1.3.3.4 Formato orario

Ogni orario deve essere così rappresentato:

[HH]:[MM]

dove:

- **HH**: rappresenta l'ora usando due cifre, con un numero che va da 00 a 23;
- **MM**: rappresenta i minuti usando due cifre, con un numero che va da 00 a 59.

La rappresentazione appena illustrata segue lo standard internazionale descritto nell'ISO 8601. Inoltre, ciò permette un corretto ordinamento alfanumerico.
Tutti gli orari sono intesi secondo il fuso orario CET (UTC+1) o CEST (UTC+2) in base alla data.

3.1.3.3.5 Come riferirsi ad altri documenti

Per le varie tipologie di riferimenti vengono applicati i seguenti formalismi:

- **Decisione verbale**: ogni decisione fatta durante un verbale è inserita all'interno della tabella "Riepilogo delle decisioni", in cui ogni scelta è così tracciata:

  [VV]_[AAAA]-[MM]-[DD].[N]

  dove:

  - **VV**: può assumere solo due valori, VE se il verbale a cui si fa riferimento è un verbale esterno, o VI se il verbale è interno;
  - **AAAA-MM-DD**: segue le norme descritte nella sezione 3.1.5.2;
  - **N**: è il numero che identifica la scelta fatta.

  Se ci si riferisce all'intero verbale allora il numero finale non apparirà.

- **Documenti di progetto**: se ci si riferisce ad un documento che non sia un verbale, per evitare ambiguità, esso sarà riferito in corsivo e secondo il seguente formalismo:

[Nome Documento] [Versione]_[Numero della sezione]

dove:

- **Nome Documento**: nome del documento, con ogni lettera iniziale in maiuscolo ad eccezione di eventuali di congiunzioni e preposizioni;
- **Versione**: si riferisce alla versione del documento ed è rappresentato come descritto nella sezione 3.1.3.2;
- **Numero della sezione**: può essere non presente, se ci si riferisce ad un intero documento, altrimenti è rappresentato come descritto nella sezione 3.1.3.6.

- **Risorse esterne**: tutti i riferimenti a risorse esterne (URI) sono scritti in caratteri di colore blu, seguiti da parentesi tonde contenenti la data di consultazione.

3.1.3.3.6 Convenzioni di termini

- **Ruoli**: ogni ruolo è scritto con la prima lettera in maiuscolo;
- **Acronimi**: gli acronimi devono essere scritti con tutte le lettere in maiuscolo;
- **Termini del glossario**: i termini a glossario rispettano quanto indicato nella sezione 1.3.

3.1.3.4 Ciclo di vita dei documenti

Ogni documento si può trovare in una delle seguenti tre fasi:

- **Sviluppo**: lo sviluppo è la prima fase in cui si trova un documento e ci rimane fino a quando il Responsabile di progetto non ritiene che il documento sia pronto per la verifica;
- **Verifica**: avviene da parte di un Verificatore secondo le modalità scritte nel *Piano di Qualifica v2.0.0* . Al termine della verifica, in base al risultato ottenuto, il Responsabile di progetto deciderà se approvare il documento oppure segnalare gli errori a chi si occupa della stesura di quel documento;
- **Approvazione**: il documento è stato approvato dal Responsabile di progetto e il numero della versione è aggiornato nelle modalità descritte nella sezione 3.1.3.2.

3.1.3.5 Gestione di configurazione

3.1.3.5.1 Controllo della configurazione

Per identificare e tracciare le richieste di cambiamenti, analizzare e valutare le modifiche effettuate e approvare o meno quelle proposte, viene utilizzato il sistema di issue di Github. Le issue riportano nella loro descrizione la ragione della modifica effettuata, nonché la o le $commit_g$ ad esse associate ed eventuali ulteriori issue correlate.

3.1.3.5.2 Stato di configurazione

Successivamente ad ogni revisione, l'ultima commit effettuata viene marcata con un'etichetta di versione, essa costituirà la nuova baseline di riferimento.

3.1.3.5.3 Rilasci e consegne

I rilasci e le consegne dei prodotti software e della documentazione correlata sono sottoposti a verifica e validazione. Prima di ogni revisione viene consegnato al Committente, secondo tempistiche stabilite dal Piano di Progetto e mezzi concordati, un archivio contenente tutta la documentazione richiesta in formato PDF. La consegna è accompagnata da una Lettera di Presentazione, anch'essa inclusa nell'archivio.

3.1.4 Strumenti utili

3.1.4.1 Latex

Per la stesura dei documenti viene usato LaTeX che garantisce uniformità tra i documenti grazie ad alcuni template personalizzati e permette di trattare i documenti come codice che può essere versionato.
L'*editor*$_g$ consigliato è *Texmaker*$_g$ multi-piattaforma ed open source, si ricorre anche all'uso del servizio online *ShareLatex*$_g$ per poter lavorare simultaneamente e con aggiornamenti in tempo reale allo stesso documento.
Entrambi gli editor citati integrano un visualizzatore pdf e la funzionalità di controllo ortografico.

http://www.xm1math.net/texmaker/ (consultato il 2018-03-20)

https://www.sharelatex.com/ (consultato il 2018-03-20)

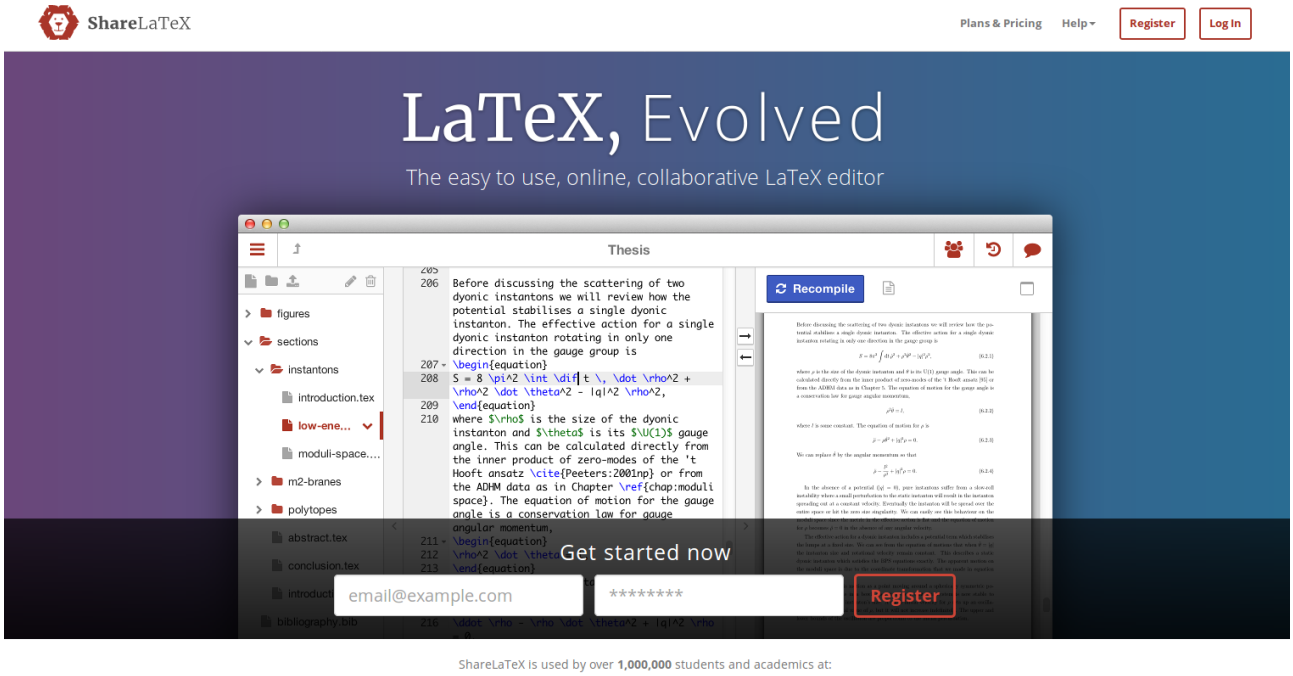

Figura 4: Homepage di ShareLatex

3.2 Verifica

3.2.1 Scopo del processo

Il processo di verifica accerta che durante l'esecuzione delle attività dei processi non siano presenti errori.

3.2.2 Descrizione

La verifica va eseguita sia a progetto finito che durante tutto lo sviluppo per accertarsi che sia mantenuta la consistenza, la correttezza e la completezza del software.
Per effettuare la verifica si effettuano due analisi: analisi statica e analisi dinamica.

3.2.3 Attività

3.2.3.1 Metriche e obbiettivi di qualità

Sia gli obbiettivi che le metriche devono essere classificati secondo la seguente notazione:

[Classe][Tipo][Oggetto][Codice identificativo] - [Nome]

- **Classe**: indica se si tratta di un obbiettivo o di una metrica. Si utilizzerà:
  - **O**: per un obbiettivo;
  - **M**: per una metrica.
- **Tipo**: stabilisce se riguarda un prodotto o un processo e può assumere i seguenti valori:
  - **PD**: per gli obbiettivi di prodotto;
  - **PC**: per gli obbiettivi di processo.
- **Oggetto**: nel caso di obbiettivi o metriche di prodotto, indica se si riferisce alla parte software oppure ad un documento. Si utilizzerà:
  - **D**: per i documenti;
  - **S**: per il software.
- **Codice identificativo**: codice numerico univoco necessario per l'identificazione;
- **Nome**: nome per descrivere l'obiettivo o la metrica.

3.2.3.1.1 Metriche per i processi

- **MPC1 - Schedule Variance**: è un indice di efficienza che ha come oggetto la durata temporale di un processo o di un'attività. Questo indice aiuta il team nell'analisi dell'u-

tilizzo di risorse temporali.
La formula per calcolarlo è:

$$SV = data\ conclusione\ reale - data\ conclusione\ preventivata$$

- **MPC2 - Cost Variance**: è una metrica che analizza il costo e le risorse devolute ad un processo o ad una attività.
  La formula per calcolarla è:

$$CV = costo\ delle\ risorse\ reale - costo\ delle\ risorse\ preventivato$$

- **MPC3 - SPICE**: tale standard viene illustrato nel dettaglio nell'Appendice A.1; in generale viene utilizzato alla fine di ogni periodo per monitorare e valutare la qualità dei processi impiegati.

3.2.3.1.2 Metriche per la documentazione

- **MPDD1 - Indice Gulpease**: l'indice di Gulpease è un indice di leggibilità di un testo, tarato sulla lingua italiana; è stato scelto questo indice poiché tutti i documenti scritti dal gruppo JurassicSWE sono in lingua italiana.
  L'indice di Gulpease considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero di lettere.

$$89 + \frac{300*(numero\ delle\ frasi) - 10*(numero\ delle\ lettere)}{(numero\ delle\ parole)}$$

  Il risultato della precedente formula è un valore compreso tra 0 e 100, dove 100 indica la leggibilità più alta mentre 0 la leggibilità più bassa.
  In generale risulta che i testi con un indice:

  - inferiore a 80 sono difficili da leggere per chi ha la licenza elementare;
  - inferiore a 60 sono difficili da leggere per chi ha la licenza media;
  - inferiore a 40 sono difficili da leggere per chi ha un diploma superiore.

  Naturalmente, oltre al valore dell'indice di un documento, bisogna tenere anche presente la notorietà dei singoli termini utilizzati nel documento stesso.

- **MPDD2 - Correzione errori ortografici**: indica il numero di errori ortografici rilevati manualmente da un Verificatore o automaticamente dal controllo ortografico dell'$editor_g$ in cui vengono redatti i documenti.
  Poiché i documenti non devono contenere errori ortografici, questi dovranno essere immediatamente corretti dal Verificatore stesso.

3.2.3.1.3 Metriche per il software

- **MPDS1 - Requisiti obbligatori soddisfatti**: possibilità di monitorare in ogni istante la percentuale di requisiti obbligatori soddisfatti, con la seguente formula:

$$\frac{numero\ requisiti\ obbligatori\ soddisfatti}{numero\ totale\ requisiti\ obbligatori}$$

- **MPDS2 - Requisiti accettati soddisfatti**: possibilità di monitorare in ogni istante la percentuale di requisiti accettati soddisfatti, con la seguente formula:

$$\frac{numero\ requisiti\ accettati\ soddisfatti}{numero\ totale\ requisiti\ accettati}$$

- **MPDS3 - percentuale superamento test**: questa metrica indica quanti dei test implementati hanno esito positivo e può essere ottenuta così:

$$\frac{numero\ requisiti\ obbligatori\ individuati}{numero\ requisiti\ obbligatori\ soddisfatti} * 100$$

- **MPDS4 - Numero parametri per metodo**: un numero troppo elevato di parametri in un metodo potrebbe indicare un grado di complessità troppo elevato del metodo.
- **MPDS5 - Numero attributi per metodo**: un valore elevato di attributi può indicare che una classe si fa carico di una quantità eccessiva di responsabilità. Una soluzione potrebbe quindi essere quella di scomporre una parte di tale classe in una seconda classe collegata tramite incapsulamento, ridefinendone le interfacce.
- **MPDS6 - Grado di accoppiamento**: il grado di accoppiamento viene calcolato per capire quanto le classi siano o meno dipendenti rispetto al resto del sistema. I valori risultanti sono utilizzati per il calcolo dell'instabilità.
- **MPDS7 - Complessità ciclomatica**: l'indice di complessità di un programma aiuta ad identificare il numero di test necessari al raggiungimento di un $coverage_g$ completo. Questa metrica software può essere applicata anche a $packages_g$, moduli, metodi o classi. Il calcolo avviene sfruttando il grafo di controllo di flusso e l'indice non è altro che il numero di cammini indipendenti attraverso il $codice\ sorgente_g$.
  La formula utilizzata è:

$$e - n + 2p$$

  Dove:

  - **e**: è il numero di nodi del grafo, nonché il numero di tutti i gruppi indivisibili di istruzioni;
  - **n**: rappresenta il numero di archi del grafo, cioè il numero di collegamenti tra due nodi tali che il nodo seguente possa essere eseguito immediatamente dopo il nodo preso di riferimento;

  - **p**: è il numero di componenti connesse.

- **MPDS8 - Numero metodi per classe**: qualora una classe dovesse avere un numero elevato di metodi, probabilmente essa viola i principi della programmazione $SOLID_g$ soprattutto quello della $Single\ Responsibility_g$, per il quale ogni classe deve assolvere ad un solo compito. In caso la classe presenti un elevato numero di metodi, sarà preferibile scomporla in più classi, se possibile.
- **MPDS9 - Numero di classi per $package_g$**: un numero troppo elevato di classi all'interno di un unico package incrementerebbe le responsabilità dello stesso, andando così contro i principi della programmazione SOLID.
- **MPDS10 - Test automatici**: questa metrica dà un idea della percentuale di test automatici implementati dal gruppo JurassicSWE. La volontà è quella di aumentare sempre più il valore di questa metrica.

  Tale metrica viene calcolata nel seguente modo:

$$\frac{numero\ test\ automatici}{numero\ test\ manuali} * 100$$

- **MPDS11 - Rapporto linee di commento/linee di codice**: avere un buon rapporto linee di codice su linee di commento all'interno dello stesso aumenta la manutenibiltà.

  Tale rapporto si calcola nel seguente modo:

$$\frac{numero\ di\ linee\ di\ codice\ totali}{numero\ di\ linee\ di\ commento} * 100$$

- **MPDS12 - Failure Avoidance**: questa metrica viene applicata per monitorare l'affidabilità del prodotto nel far fronte a situazioni erronee e/o impreviste.

  La formula per calcolarla è:

$$\frac{numero\ situazioni\ anomale\ evitate}{numero\ totale\ situazioni\ anomale\ occorse}$$

3.2.3.2 Analisi

3.2.3.2.1 Analisi statica

L'analisi statica è una tecnica che permette la rivelazione di errori all'interno della documentazione e del codice senza l'esecuzione del prodotto software, permettendo di effettuare verifiche anche se il prodotto non è finito; quindi è un'analisi che può essere effettuata durante tutto il ciclo di vita.
Si effettua tramite due tecniche:

- **Walkthrough**: si esegue una lettura a largo spettro con lo scopo di rivelare errori. Per essere efficace, la lettura deve essere effettuata da più persone, tra le quali non dovrebbe esserci la persona che l'ha redatta. Lo scopo è stilare una lista di controllo contenente i difetti rilevati, documentando come si è individuato l'errore e discutendo possibili soluzioni.
  Viene effettuata tramite le seguenti fasi:
  - pianificazione;
  - lettura;
  - discussione;
  - correzione dei difetti.
- **Inspection**: è una lettura mirata a rilevare la presenza di difetti che si basa su presupposti ed *error guessing*$_g$, per esempio su una lista di controllo.
  Le fasi sono:
  - pianificazione;
  - definizione lista di controllo;
  - lettura;
  - correzione dei difetti.

Le due metodologie sono di tipo *desk check*$_g$; tuttavia il walkthrough richiede maggior attenzione e tempo, mentre l'*inspection*$_g$ è più rapido, ma si basa su errori presupposti.
La strategia migliore è effettuare all'inizio il walkthrough e quando si ha individuato la maggior parte degli errori, focalizzarsi su punti critici utilizzando l'inspection.

3.2.3.2.2 Analisi dinamica

L'analisi dinamica è una tecnica di analisi del prodotto che richiede l'esecuzione del programma, ed è usata sia nella verifica sia nella validazione. Viene effettuata tramite test su cui si effettua l'analisi dei risultati; per essere considerata efficace deve essere ripetibile e quindi, per ogni caso di prova, si deve essere a conoscenza di:

- **Precondizione**: stato iniziale del sistema e i valori di ingresso (input);

- **Postcondizione**: stato finale del sistema, valori in uscita (output).

Per effettuare l'analisi dinamica si utilizzano implementazioni dei seguenti concetti:

- **Driver**: è una componente fittizia, che sostituisce l'unità chiamante per testare l'unità chiamata;
- **Stub**: componente utilizzata in sostituzione delle unità chiamate per testare un'unità chiamante, simulando il comportamento di codice non ancora esistente;
- **Logger**: componente non intrusivo di registrazione dei dati per l'analisi dei risultati.

3.2.3.3 Test

Un test è un esperimento effettuato con lo scopo di cercare malfunzionamenti, cioè un comportamento inatteso del sistema.
I test sono onerosi in quanto richiedono molte risorse umane e infrastrutturali, e molte attività di ricerca, analisi e correzione. Lo sforzo non deve comunque essere eccessivo poiché, per la legge del rendimento decrescente, superata una certa soglia di risorse impiegate, il loro rapporto con i benefici ottenuti non è vantaggioso.
Elenco dei test:

- **Test di unità**: Servono per verificare la correttezza del codice della più piccola parte di software testabile nell'applicazione, chiamata unità. Ogni unità deve essere prima sottoposta a test e poi potrà essere integrata insieme alle unità già testate. Per velocizzare i tempi richiesti dal test di unità una buona strategia è svolgerli in parallelo e automatizzarli dove è possibile.
  Questo test permette di individuare circa i 2/3 dei difetti trovati in tutta l'analisi dinamica. Viene effettuato dallo stesso *Programmatore*$_g$ per le unità più semplici o da un Verificatore;
- **Test di integrazione**: Si usa per testare il funzionamento integrato di tutti i moduli che compongono un intero processo. Rileva difetti di progettazione, problemi derivati da integrazione con applicazioni non ben conosciute o difetti causati da componenti che presentano un comportamento inaspettato.
  L'integrazione delle componenti avviene:
  - **Assemblando le parti in modo incrementale**: cioè aggiungendo solo insiemi ben verificati, così appena si trova un errore è molto probabile che sia causato dall'ultima parte inserita;
  - **Assemblando i produttori prima dei consumatori**: garantendo che i produttori forniscano ai secondi un flusso di chiamate e dati corretti;
  - **Assemblando in modo che ogni passo di integrazione sia sempre reversibile**: consentendo di retrocedere verso uno stato noto e sicuro.

  L'integrazione incrementale si può applicare in due diverse modalità:

  - **Bottom-up**: si integrano per prime le parti con minore dipendenza funzionale, si riduce il numero di stub necessari ma si ritarda la possibilità di testare funzionalità di alto livello;
  - **Top-down**: in questo caso si sviluppano prima le unità più esterne poste sulle foglie dell'albero delle dipendenze; necessita l'uso di molti stub ma permette di integrare prima le funzioni di alto livello.

- **Test di sistema**: Il test di sistema è usato dal fornitore per accertare che il prodotto rispetti i requisiti richiesti, controllando il comportamento dinamico del sistema completo;
- **Test di regressione**: Consiste in una ripetizione selettiva dei test di unità, d'integrazione e di sistema, integrando solo le parti che hanno passato i test di unità.
  Ha lo scopo di accertare che modifiche effettuate durante la correzione o estensione non abbiano compromesso l'efficacia del sistema, e quindi deve essere eseguito dopo ogni modifica o aggiunta;
- **Test di collaudo**: Attività supervisionata dal committente atta a dimostrare la conformità del prodotto tramite il superamento dei casi di prova specificati all'interno del contratto.
  Se il collaudo va a buon fine il prodotto può essere rilasciato.

3.2.3.4 Codice identificativo

Ogni test è strutturato come segue:

- codice identificativo;
- descrizione;
- stato.

Per ciascun test è assegnato un codice identificativo la cui sintassi segue il seguente formalismo:

T{tipo}{codice_identificativo}

Dove:

- **tipo**: identifica uno dei seguenti tipi di test:
  - **U**: test di unità;
  - **I**: test di integrazione;
  - **S**: test di sistema;
  - **V**: test di validazione.
- **codice_identificativo**: assume i seguenti valori in base al tipo di test:
  - **codice_numerico**: associato ai test di unità e di integrazione, è un codice progressivo che parte da 1;

  - **codice_requisito**: associato ai test di sistema e di validazione. Identifica il codice univoco associato ad ogni requisito descritto nel documento *Analisi dei Requisiti v2.0.0.*

Inoltre lo stato del test può assumere i seguenti valori:

- implementato;
- non implementato;
- non eseguito;
- superato;
- non supportato.

3.2.3.5 Gestione anomalie

Il Verificatore deve procedere alla verifica dell'operato e alla risoluzione di ogni ticket. Ciascuna anomalia individuata viene inserita all'interno di un elenco. L'elenco delle anomalie è comunicato al Responsabile che si occupa di assegnare un ticket per la correzione di esse.

3.2.3.6 Gestione modifiche

Qualora durante l'attività di verifica si presenti un problema relativo alla documentazione o al codice prodotto, che necessita di una correzione, verrà inoltrata una richiesta al Responsabile in modo tale che possa essere controllata l'effettiva esistenza del problema o meno. Nel caso si verifichi il problema, il Responsabile deve assegnare la modifica a chi ritiene più opportuno. Il Verificatore constata che la modifica sia avvenuta con successo.
La richiesta da presentare al Responsabile deve avere la seguente struttura:

- **autore**: nome e cognome di colui che invia la richiesta di modifica;
- **documento o file**: nome del documento/file che necessita di una modifica;
- **motivazione**: specifica del perché è richiesta la modifica.

Per tenere traccia dell'intero processo di modifica alla richiesta viene aggiunto se la stessa è stata approvata oppure rifiutata.

3.2.4 Strumenti utili

- **Verifica ortografica**: per verificare la correttezza ortografica dei documenti redatti dal gruppo viene utilizzata la verifica automatica offerta dall'editor con cui vengono scritti i documenti; per garantire un ulteriore correttezza ogni sezione viene successivamente riletta a mano dai Verificatori.
- **Analisi statica**: per l'analisi statica del codice JavaScript vengono utilizzati i seguenti strumenti:

  - **JSHint**: strumento *Open source*$_g$ che permette di rilevare errori e potenziali problemi nel codice: http://www.jshint.com/ (consultato il 2018-03-21)
  - **Closure Compiler**: strumento per compilare e analizzare codice JavaScript in supporto a JSHint: https://closure-compiler.appspot.com/home (consultato il 2018-03-21)
- **Analisi dinamica**: per l'analisi dinamica vengono utilizzati i seguenti strumenti:
  - **Mocha**: permette l'esecuzione di test asincroni e in serie, consentendo segnalazioni flessibili e accurate: http://mochajs.org/ (consultato il 2018-03-21)
  - **Karma**: un ambiente di testing, utile ad effettuare test su browser e dispositivi: http://karma-runner.github.io/0.13/index.html (consultato il 2018-03-21)

3.3 Validazione

3.3.1 Scopo del processo

La validazione è un processo usato per confermare che le caratteristiche del software siano conformi alle attese, e quindi alle richieste dell'utente.

3.3.2 Descrizione

Di questa fase se ne occupano i Verificatori i quali verificano che il software rispetti a pieno i requisiti imposti dal Committente e dal Proponente. Una volta che si ottiene l'approvazione dei Verificatori, il Responsabile di progetto controlla eventuali errori o mancanze.

3.3.3 Attività

3.3.3.1 Analisi dinamica

Il Verificatore ha il compito di rieseguire tutti i test ponendo attenzione ai risultati, in particolare a quelli ottenuti dai test di validazione. Il Responsabile dovrà poi analizzare i risultati ottenuti per decidere se rieseguire i test ulteriormente.

3.3.3.2 Test

- **Test di validazione**: il test di validazione rappresenta il collaudo del prodotto in presenza del proponente. Al superamento di tale collaudo segue il rilascio ufficiale dl prodotto sviluppato

3.3.3.3 Codice identificativo

Ogni test è strutturato come segue:

- codice identificativo;
- descrizione;
- stato.

Per ciascun test è assegnato un codice identificativo la cui sintassi segue il seguente formalismo:

TV{codice_requisito}

Dove:

- **codice_requisito**: identifica il codice univoco associato ad ogni requisito descritto nel documento *Analisi dei Requisiti v2.0.0.*

Inoltre lo Stato del test può assumere i seguenti valori:

- implementato;
- non implementato;
- non eseguito;
- superato;
- non supportato.

3.3.4 Strumenti utili

3.3.4.0.1 Validatore HTML

Per la validazione delle pagine HTML viene utilizzato lo strumento offerto da W3C: https://validator.w3.org/ (consultato il 2018-03-21)

3.3.4.0.2 Validatore CSS

Per la validazione dei fogli di stile CSS_g viene utilizzato lo strumento offerto dal W3C: https://jigsaw.w3.org/css-validator/ (consultato il 2018-03-21)

4 Processi organizzativi

4.1 Gestione organizzativa

4.1.1 Scopo

Il processo di gestione contiene le attività e i compiti generici che possono essere utilizzati da qualunque parte che deve gestire i rispettivi processi.
Il Responsabile ha la responsabilità della gestione di prodotto, di progetto e dei compiti dei processi applicabili, come i processi di acquisizione, fornitura, sviluppo, funzionamento, manutenzione e supporto.

4.1.2 Descrizione

Questo processo consiste nelle seguenti attività:

- iniziazione e definizione dello scopo;
- pianificazione;
- esecuzione e controllo;
- verifica e valutazione;
- chiusura.

4.1.3 Ruoli di progetto

Ciascun membro del gruppo JurassicSWE, a rotazione, si impegnerà a ricoprire i ruoli che corrispondono alle omonime figure aziendali. Nel documento *Piano di Progetto v2.0.0* vengono organizzate e pianificate le attività assegnate ai specifici ruoli previsti nell'attività di progetto. I ruoli che ogni componente del gruppo sarà tenuto a rispettare, in conformità a quanto stabilito da ciascuno di essi, sono descritti nei sotto paragrafi successivi.

4.1.3.1 Responsabile di progetto

Il Responsabile del progetto è una figura molto importante all'interno del team in quanto su di lui ricadono le responsabilità di pianificazione, gestione, controllo e coordinamento. Un altro ruolo del Responsabile è quello di rappresentare il gruppo all'esterno di esso, in quanto saranno a suo carico le comunicazioni dirette con Committente e Proponente.
In sunto, il Responsabile di progetto deve:

- gestire, controllare e coordinare le risorse e le attività del gruppo;

- gestire, controllare e coordinare gli altri componenti del gruppo;
- analizzare e gestire le criticità;
- approvare i documenti.

4.1.3.2 Amministratore

L'*Amministratore*$_g$ ha il compito fondamentale di supporto e controllo dell'ambiente di lavoro.
Le sue responsabilità sono:

- amministrazione delle infrastrutture di supporto;
- risoluzione di problemi legati alla gestione dei processi;
- gestione della documentazione di progetto;
- controllo di versioni e configurazioni.

4.1.3.3 Analista

L'Analista si occupa dell'analisi dei problemi e del dominio applicativo; è una figura che non sarà sempre presente durante il progetto, ma di estrema importanza per i compiti svolti, i quali sono:

- studio del dominio del problema e del problema stesso, definendone complessità e requisiti;
- redazione dei documenti quali *Analisi dei Requisiti* e *Studio di Fattibilità*.

4.1.3.4 Progettista

Il Progettista gestisce gli aspetti tecnologici e tecnici del progetto.
Le sue responsabilità sono:

- effettuare scelte efficienti ed ottimizzate su aspetti tecnici del progetto;
- sviluppare un'architettura che sfrutti tecnologie note ed ottimizzate su cui basare un prodotto stabile e mantenibile.

4.1.3.5 Programmatore

Il Programmatore è il responsabile della codifica del progetto e delle componenti di supporto, che serviranno per effettuare le prove di verifica e validazione sul prodotto.
Le sue responsabilità sono:

- implementare le decisioni del progettista;
- creare o gestire componenti di supporto per la verifica e la validazione del codice.

4.1.3.6 Verificatore

Il Verificatore ha solide conoscenze delle *Norme di Progetto* e, in quanto tale, egli garantirà che i processi svolti dal gruppo siano conformi alle norme e alle attese.
Le sue responsabilità sono:

- controllo delle attività di progetto secondo le normative stabilite.

4.1.4 Attività

4.1.4.1 Gestione delle comunicazioni

Seguono le modalità di comunicazione che il gruppo JurassicSWE adotterà per tutta la durata del progetto.
Le comunicazione avvengono principalmente verso due canali: interno, che coinvolge i partecipanti del gruppo, ed esterno, che comprende Proponente, Committente ed eventuali altri Stakeholder.

4.1.4.1.1 Comunicazioni interne

Le comunicazioni interne del gruppo avvengono utilizzando lo strumento di collaborazione aziendale Slack, perché offre un ambiente creato per gestire al meglio il lavoro di gruppo.
Questo strumento è stato preferito ad altri software di messaggistica per la possibilità di creare canali tematici e per l'integrabilità che offre con altri servizi di utilità usati dal team.
Ogni canale è specifico per una determinata attività ed è inoltre possibile avviare delle discussioni private tra due o più membri del gruppo.

4.1.4.1.2 Comunicazioni esterne

Le comunicazioni con soggetti esterni al gruppo sono di competenza del Responsabile di progetto che utilizzerà la casella di posta elettronica creata in fase di istituzione del team JurassiSWE: JurassicSWE@gmail.com.
Il Responsabile di progetto, se necessario, dovrà tenere informati tutti i componenti del gruppo sulle discussioni con le componenti esterne tramite i canali di comunicazione interna.

4.1.4.2 Incontri interni del team

Le riunioni interne del team verranno sempre organizzate dal Responsabile in accordo con tutti i membri del gruppo, nel rispetto delle modalità di comunicazione illustrate nella sezione 4.1.1.1.
La data e l'orario degli incontri vengono stabiliti attraverso *Doodle*$_g$, uno strumento che permette ai membri di inserire le proprie disponibilità nel calendario; verrà scelto il giorno in cui

almeno 3 membri potranno essere presenti.

https://doodle.com/it/ (consultato il 2018-03-15)

4.1.4.2.1 Verbali interni di riunione

Ad ogni riunione interna corrisponderà un verbale, il cui compito di redazione sarà onere del *Segretario*$_g$, nominato a turno dal Responsabile di progetto, il quale dovrà illustrare l'ordine del giorno e tenere nota delle discussioni e decisioni avvenute.
La struttura dei verbali viene descritta nella sezione 3.1.4.

4.1.4.3 Incontri esterni del team

Il Responsabile di Progetto ha il compito di comunicare, ed organizzare gli incontri esterni con il Proponente o Committente.
Come per gli incontri interni, se il Responsabile o un membro del gruppo ritiene necessario effettuare un incontro esterno, e il Responsabile stesso lo ritiene necessario, verrà comunicato a tutti i membri attraverso i mezzi di comunicazione interna adottati dal gruppo, ed in base alle disponibilità di tre membri del gruppo verrà fissato un incontro utilizzando l'email ufficiale del gruppo.

4.1.4.3.1 Verbali esterni di riunione

Come per le riunioni interne, per ogni incontro esterno del gruppo verrà redatto un verbale, la cui struttura viene descritta nella sezione 3.1.4.

4.1.4.4 Gestione di progetto

Per gestire meglio il lavoro di gruppo ed accertarsi che tutti i lavori vengano svolti correttamente e che niente venga lasciato per strada, si utilizza *Wrike*$_g$, un software di project management online che offre un ottimo sistema di ticketing per la gestione di progetto, usato per l'assegnazione di attività all'interno del gruppo.

4.1.4.4.1 Ticketing

E' compito del Responsabile suddividere le attività tra i diversi componenti del gruppo. La scaletta utilizzata in questa fase è la seguente:

- Creare una nuova attività, assegnandole un titolo;
- Inserire una descrizione che contiene un breve riassunto dell'attività da svolgere;

- Aggiungere un assegnatario all'attività;
- Aggiungere una data di inizio e fine dell'attività;
- Notificare alle persone interessate attraverso canale Telegram o Slack le attività da svolgere.

4.1.5 Strumenti utili

In questa sezione verranno riportate tutte le tecnologie che il team utilizzerà per tutta la durata del progetto.

4.1.5.1 Strumenti di comunicazione

Per quanto riguarda le comunicazioni interne, come già segnalato nel verbale VI_20180309, il gruppo intende utilizzare *Telegram*$_g$, servizio di messaggistica istantanea basato su cloud, e *Slack*$_g$, strumento di collaborazione aziendale.
Per quanto riguarda le comunicazioni esterne al gruppo è invece stato deciso di utilizzare Gmail.
Per le discussioni di technology baseline e product baseline, con il Prof. Riccardo Cardin, il gruppo utilizzerà *Hangout*$_g$, come concordato con il professore.
Hangout è un'applicazione che offre un'ampia gamma di servizi visto che, oltre alla classica messaggistica, offre l'opportunità di effettuare videochiamate, chiamate VoIP e videoconferenze.

4.1.5.2 Strumenti di condivisione

Per la condivisione di materiale utile il gruppo farà utilizzo di Google Drive per una rapida e facile condivisione di documenti.

4.1.5.3 Strumenti di coordinamento

Il sistema adottato per la coordinazione del team, attraverso l'assegnazione di *task*$_g$, è Wrike. Alcune delle funzionalità che questo strumento offre sono:

- possibilità di creare task, inserirli in una cartella o sotto un altro task;
- indicare una persona a cui assegnare il task;
- indicare una data di scadenza del task;
- inserire una descrizione del task.

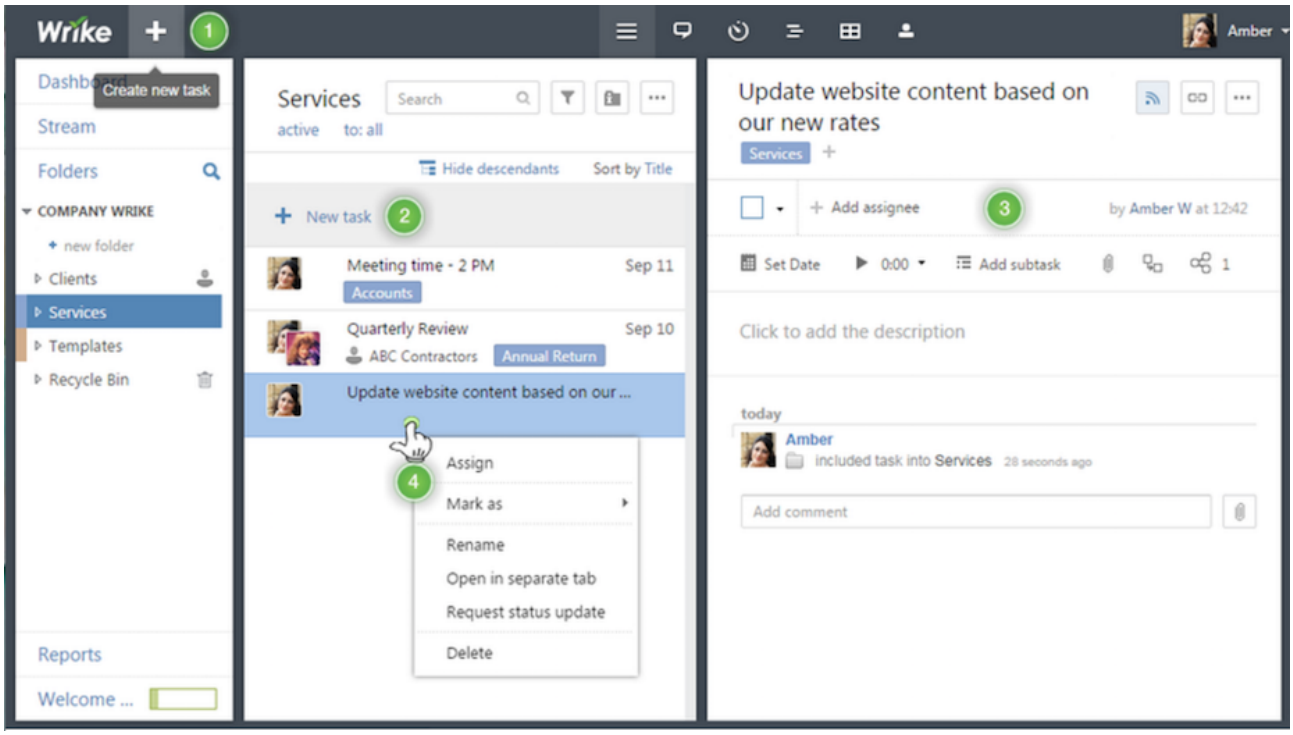

Figura 5: Wrike

4.1.5.4 Strumenti di versionamento

Per il salvataggio e il versionamento il gruppo intende utilizzare *GitHub*$_g$, il quale si basa sul sistema di versionamento *Git*$_g$. Per accedere al servizio GitHub il team utilizzerà *GitKraken*$_g$. Questo servizio è inoltre stato scelto per la possibilità segnalare anomalie attraverso issue.

4.1.5.5 Strumenti di grafica

Per la creazione dei grafici sono stati utilizzati i seguenti strumenti:

- **Diagrammi di Gantt**: è stato scelto come strumento *Gantt Project*$_g$, software per la creazione di diagrammi con la possibilità di inserire task e *milestone*$_g$;
- **Istogrammi e diagrammi a torta**: è stato scelto lo strumento Microsoft Office Excel 2016, il quale permette di creare diagrammi velocemente e intuitivamente;
- **Diagrammi UML**: è stato scelto il software Astah Professional, il quale permette la produzione di diagrammi UML, tra cui quello di robustezza, in modo semplice e chiaro.

4.1.5.6 Sistemi operativi

Data la mancata presenza di requisiti che impongano restrizioni sul sistema operativo da utilizzare, i membri del team potranno indistintamente usare sistemi *Windows*$_g$, *MacOSX*$_g$ o *Linux*$_g$.

4.2 Formazione del team

Per quanto riguarda l'apprendimento basilare delle tecnologie richieste per il progetto, l'intero team ha optato per l'auto-formazione con condivisione di informazioni utili da parte dei membri con più conoscenze in materia.