



# Norme di Progetto

*Gruppo DigitalCookies — Progetto SWEDesigner*

[digitalcookies.group@gmail.com](mailto:digitalcookies.group@gmail.com)

## Informazioni sul documento

<b>Versione</b>	1.0.0
<b>Redazione</b>	Carlo Sindico, Alessia Bragagnolo, Alberto Giudice
<b>Verifica</b>	Alberto Giudice, Christian Cabrera
<b>Approvazione</b>	Davide Albertini
<b>Uso</b>	Interno
<b>Distribuzione</b>	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo DigitalCookies

## Descrizione

Questo documento descrive le regole, gli strumenti e le convenzioni adottate dal gruppo DigitalCookies durante la realizzazione del progetto SWEDesigner.

## Registro delle modifiche

Versione	Data	Collaboratori	Ruolo	Descrizione
1.0.0	02-03-2017	Davide Albertini	Responsabile	Approvazione
0.2.0	02-03-2017	Christian Cabrera	Verificatore	Verifica sezione processi di supporto e organizzativi
0.1.0	01-03-2017	Alberto Giudice	Verificatore	Verifica sezione introduzione e sezione processi primari
0.0.6	28-02-2017	Alessia Bragagnolo	Amministratore	Fine sezione processi organizzativi
0.0.5	28-02-2017	Alberto Giudice	Amministratore	Inizio stesura sezione processi organizzativi
0.0.4	27-02-2017	Alberto Giudice	Amministratore	Stesura sezione processi di supporto
0.0.3	27-02-2017	Carlo Sindico	Amministratore	Stesura sezione processi primari
0.0.2	24-02-2017	Alessia Bragagnolo	Amministratore	Stesura sezione introduzione
0.0.1	24-02-2017	Carlo Sindico	Amministratore	Creazione del template

---

## Indice

<b>1</b>	<b>Introduzione</b>	<b>6</b>
1.1	Scopo del documento . . . . .	6
1.2	Scopo del prodotto . . . . .	6
1.3	Ambiguità . . . . .	6
1.4	Riferimenti . . . . .	7
1.4.1	Normativi . . . . .	7
1.4.2	Informativi . . . . .	7
<b>2</b>	<b>Processi primari</b>	<b>8</b>
2.1	Fornitura . . . . .	8
2.1.1	Scopo . . . . .	8
2.1.2	Aspettative . . . . .	8
2.1.3	Descrizione . . . . .	8
2.1.4	Attività . . . . .	8
2.1.4.1	Studio di fattibilità . . . . .	8
2.1.4.2	Piano di Progetto . . . . .	9
2.1.4.3	Piano di Qualifica . . . . .	9
2.2	Sviluppo . . . . .	10
2.2.1	Scopo . . . . .	10
2.2.2	Aspettative . . . . .	10
2.2.3	Descrizione . . . . .	10
2.2.4	Attività . . . . .	10
2.2.4.1	Analisi dei requisiti . . . . .	10
2.2.4.1.1	Scopo . . . . .	10
2.2.4.1.2	Aspettative . . . . .	11
2.2.4.1.3	Descrizione . . . . .	11
2.2.4.1.4	Casi d'uso . . . . .	11
2.2.4.1.5	Codice identificativo . . . . .	12
2.2.4.1.6	Requisiti . . . . .	12
2.2.4.1.7	Codice identificativo . . . . .	12
2.2.4.1.8	UML . . . . .	13
2.2.4.2	Progettazione . . . . .	13
2.2.4.2.1	Scopo . . . . .	13
2.2.4.2.2	Aspettative . . . . .	13
2.2.4.2.3	Descrizione . . . . .	13
2.2.4.2.4	Specifica Tecnica . . . . .	13
2.2.4.2.5	Definizione di Prodotto . . . . .	14
2.2.4.3	Codifica . . . . .	14
2.2.4.3.1	Scopo . . . . .	15
2.2.4.3.2	Aspettative . . . . .	15
2.2.4.3.3	Descrizione . . . . .	15

---

2.2.4.3.4	Stile di codifica . . . . .	15
2.2.4.3.5	Intestazione . . . . .	15
2.2.4.3.6	Versionamento . . . . .	16
2.2.4.3.7	Ricorsione . . . . .	16
2.2.5	Strumenti . . . . .	17
2.2.5.1	Trender . . . . .	17
2.2.5.2	Astah . . . . .	18
2.2.5.3	IntelliJ IDEA . . . . .	19
<b>3</b>	<b>Processi di supporto</b>	<b>21</b>
3.1	Documentazione . . . . .	21
3.1.1	Scopo . . . . .	21
3.1.2	Aspettative . . . . .	21
3.1.3	Descrizione . . . . .	21
3.1.4	Procedure . . . . .	21
3.1.4.1	Approvazione dei documenti . . . . .	21
3.1.5	Template . . . . .	22
3.1.6	Struttura dei documenti . . . . .	22
3.1.6.1	Prima pagina . . . . .	22
3.1.6.2	Registro delle modifiche . . . . .	22
3.1.6.3	Indice . . . . .	23
3.1.6.4	Contenuto principale . . . . .	23
3.1.6.5	Note a piè di pagina . . . . .	23
3.1.7	Versionamento . . . . .	23
3.1.8	Norme tipografiche . . . . .	24
3.1.8.1	Stile del testo . . . . .	24
3.1.8.2	Elenchi puntati . . . . .	25
3.1.8.3	Formati comuni . . . . .	25
3.1.8.4	Sigle . . . . .	26
3.1.9	Elementi grafici . . . . .	26
3.1.9.1	Tabelle . . . . .	26
3.1.9.2	Immagini . . . . .	27
3.1.10	Classificazione dei documenti . . . . .	27
3.1.10.1	Documenti informali . . . . .	27
3.1.10.2	Documenti formali . . . . .	27
3.1.10.3	Verbali . . . . .	27
3.1.11	Strumenti . . . . .	28
3.1.11.1	L <sup>A</sup> T <sub>E</sub> X . . . . .	28
3.1.11.2	TexStudio . . . . .	29
3.1.11.3	Lucidchart . . . . .	29
3.2	Verifica . . . . .	30
3.2.1	Scopo . . . . .	30
3.2.2	Aspettative . . . . .	30

---

3.2.3	Descrizione . . . . .	30
3.2.4	Attività . . . . .	31
3.2.4.1	Analisi . . . . .	31
3.2.4.1.1	Analisi statica . . . . .	31
3.2.4.1.2	Analisi dinamica . . . . .	31
3.2.4.2	Test . . . . .	32
3.2.4.2.1	Test di unità . . . . .	32
3.2.4.2.2	Test di integrazione . . . . .	32
3.2.4.2.3	Test di sistema . . . . .	32
3.2.4.2.4	Test di regressione . . . . .	32
3.2.4.2.5	Test di accettazione . . . . .	32
3.2.5	Strumenti . . . . .	33
3.2.5.1	Verifica ortografica . . . . .	33
3.2.5.2	Validazione W3C . . . . .	33
3.2.5.3	Analisi statica . . . . .	33
3.2.5.4	Analisi dinamica . . . . .	33
3.2.5.5	Metriche . . . . .	34
<b>4</b>	<b>Processi organizzativi</b>	<b>35</b>
4.1	Gestione . . . . .	35
4.1.1	Scopo . . . . .	35
4.1.2	Aspettative . . . . .	35
4.1.3	Descrizione . . . . .	35
4.1.4	Ruoli di progetto . . . . .	35
4.1.4.1	Amministratore di Progetto . . . . .	36
4.1.4.2	Responsabile di Progetto . . . . .	36
4.1.4.3	Analista . . . . .	36
4.1.4.4	Progettista . . . . .	36
4.1.4.5	Verificatore . . . . .	37
4.1.4.6	Programmatore . . . . .	37
4.1.5	Procedure . . . . .	37
4.1.5.1	Gestione delle comunicazioni . . . . .	37
4.1.5.1.1	Comunicazioni interne . . . . .	37
4.1.5.1.2	Comunicazioni esterne . . . . .	37
4.1.5.2	Gestione degli incontri . . . . .	38
4.1.5.2.1	Incontri Interni . . . . .	38
4.1.5.2.2	Incontri Esterni . . . . .	39
4.1.5.3	Gestione degli strumenti di coordinamento . . . . .	40
4.1.5.3.1	Ticketing . . . . .	40
4.1.5.4	Gestione degli strumenti di versionamento . . . . .	41
4.1.5.4.1	Repository . . . . .	41
4.1.5.4.2	Struttura del repository . . . . .	42
4.1.5.4.3	Tipi di file e .gitignore . . . . .	42

---

4.1.5.4.4	Norme sui commit . . . . .	42
4.1.5.5	Gestione dei rischi . . . . .	43
4.1.6	Strumenti . . . . .	43
4.1.6.1	Sistema operativo . . . . .	43
4.1.6.2	Slack . . . . .	43
4.1.6.3	Wrike . . . . .	44
4.1.6.4	Git . . . . .	45
4.1.6.5	GitHub . . . . .	45

## Elenco delle figure

1	Trender . . . . .	18
2	Astah Desktop per Windows . . . . .	19
3	IntelliJ IDEA Desktop per Windows . . . . .	20
4	TexStudio Desktop per Windows . . . . .	29
5	Lucidchart . . . . .	30
6	Organizzazione di un incontro interno . . . . .	39
7	Organizzazione di un incontro esterno . . . . .	40
8	Assegnazione di un ticket . . . . .	41
9	Slack Desktop per Windows . . . . .	44
10	Wrike Web application per Windows . . . . .	45

# 1 Introduzione

## 1.1 Scopo del documento

Questo documento ha lo scopo di definire le regole, gli strumenti e le convenzioni adottate dal gruppo DigitalCookies durante l'intero svolgimento del progetto. In quest'ottica, questo documento deve essere visionato da tutti i componenti del gruppo, che sono obbligati ad applicare quanto scritto al fine di mantenere omogeneità e coesione in ogni aspetto del progetto.

In caso di modifiche o aggiunte al presente documento è obbligatorio informare ogni membro del gruppo.

## 1.2 Scopo del prodotto

Lo scopo del *prodotto<sub>G</sub>* è creare un software di costruzione di diagrammi *UML<sub>G</sub>* con relativa generazione di codice *Java<sub>G</sub>*. Il codice potrà essere generato dall'utente a partire dai diagrammi UML delle *classi<sub>G</sub>* e da una versione modificata del diagramma delle *attività<sub>G</sub>*.

L'utente, interagendo con il sistema, sarà in grado di:

- delineare la struttura delle classi utilizzando lo standard UML;
- definire il corpo dei metodi delle classi sfruttando una versione modificata del diagramma delle attività;
- generare un applicativo scritto in codice Java a partire dai diagrammi sopracitati.

L'utente potrà inoltre sfruttare la *libreria<sub>G</sub>* fornita con il prodotto per generare con facilità diagrammi relativi al dominio dei giochi di carte.

L'*editor<sub>G</sub>* sarà fruibile dall'utente attraverso un *browser<sub>G</sub>* desktop idoneo all'utilizzo delle tecnologie *HTML5<sub>G</sub>*, *CSS3<sub>G</sub>* e *JavaScript<sub>G</sub>*.

## 1.3 Ambiguità

Al fine di evitare ogni ambiguità relativa al linguaggio impiegato nei documenti viene fornito il *Glossario v1.0.0*, contenente la definizione dei termini in corsivo marcati con una G pedice.

## 1.4 Riferimenti

### 1.4.1 Normativi

- **ISO/IEC 12207**  
[https://en.wikipedia.org/wiki/ISO/IEC\\_12207](https://en.wikipedia.org/wiki/ISO/IEC_12207);
- **Capitolato:**  
<http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6p.pdf>;
- **Verbale di incontro interno** con i componenti del gruppo del 25-02-2017;
- **Verbal di incontro esterno** con il *proponente<sub>G</sub>* Zucchetti S.p.A. del 23-02-2017 e del 23-03-2017.

### 1.4.2 Informativi

- Per una dettagliata guida  $\text{\LaTeX}$  fare riferimento a *Guida ai comandi  $\text{\LaTeX}$  v1.0.0*;
- Per una dettagliata guida *Git<sub>G</sub>* fare riferimento a *Guida ai comandi Git v1.0.0*.



## 2 Processi primari

### 2.1 Fornitura

#### 2.1.1 Scopo

Questo *processo<sub>G</sub>* ha lo scopo di trattare le norme e i termini che i membri del gruppo DigitalCookies sono tenuti a rispettare per diventare fornitori della proponente Zucchetti S.p.A. e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin per quanto concerne il prodotto SWEDesigner.

#### 2.1.2 Aspettative

Nel corso dell'intero progetto il gruppo intende instaurare con Zucchetti S.p.A., in particolare nella figura del referente Dr. Gregorio Piccoli, un rapporto di costante collaborazione al fine di:

- Determinare aspetti chiave per soddisfare i bisogni del proponente;
- Determinare vincoli sui processi e sui requisiti;
- Stimare i costi;
- Concordare la qualifica del prodotto.

#### 2.1.3 Descrizione

Il gruppo intende mantenere un costante dialogo con il proponente per avere un riscontro efficace sul lavoro svolto.

#### 2.1.4 Attività

##### 2.1.4.1 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 è redatto dall'*Analista<sub>G</sub>* sulla base dei seguenti punti:

- **Dominio tecnologico e applicativo:** si valuta il capitolato prendendo in considerazione la conoscenza attuale delle tecnologie richieste da parte dei membri del gruppo, valutando anche eventuali esperienze passate con problematiche simili;

- **Rapporto costi/benefici:** si analizzano la quantità di requisiti obbligatori, il costo in rapporto ai risultati previsti e l'interesse del gruppo rispetto alle tematiche del *capitolato<sub>G</sub>*;
- **Individuazione dei rischi:** si analizzano i punti critici della realizzazione, quali ad esempio mancanza di conoscenze adeguate o difficoltà nell'individuazione di requisiti. Si analizzano inoltre eventuali problematiche che possono sorgere in corso d'opera.

#### 2.1.4.2 Piano di Progetto

Il *Responsabile*, aiutato dagli *Amministratori*, dovrà redigere un piano da seguire nella realizzazione del progetto. Il documento dovrà contenere:

- **Analisi dei rischi:** si analizzano nel dettaglio i rischi che potrebbero insorgere nel corso del progetto e i modi per affrontarli, capendo la probabilità che essi accadano e il livello di gravità ad essi associato;
- **Pianificazione:** si pianificano le attività da svolgere nel corso del progetto, fornendo delle scadenze temporali precise;
- **Preventivo e Consuntivo:** sulla base della pianificazione si stima la quantità di lavoro necessaria per ogni fase, proponendo così un preventivo per il costo totale del progetto. Alla fine di ogni attività si redige inoltre un consuntivo di periodo per tracciare l'andamento rispetto a quanto preventivato.

#### 2.1.4.3 Piano di Qualifica

I *Verificatori* dovranno scegliere una strategia da adottare per la *verifica<sub>G</sub>* e la *validazione<sub>G</sub>* del materiale prodotto dal gruppo. Il documento dovrà contenere:

- **Visione generale della strategia di verifica:** si stabiliscono le procedure di controllo sulla qualità di processo e di prodotto, tenendo in considerazione le risorse a disposizione;
- **Misure e metriche:** si devono stabilire delle metriche oggettive per i documenti, i processi e il software;
- **Gestione della revisione:** si devono stabilire le modalità di comunicazione delle anomalie e le procedure di controllo per la qualità di processo;
- **Pianificazione del collaudo:** si devono definire nel dettaglio le metodologie di collaudo del prodotto realizzato;
- **Resoconto delle attività di verifica:** alla fine di ogni attività si devono riportare le metriche calcolate e un resoconto sulla verifica di tale attività.

## 2.2 Sviluppo

### 2.2.1 Scopo

Questo processo contiene tutte quelle attività e quei compiti svolti dal gruppo nel produrre il software finale richiesto dal proponente.

### 2.2.2 Aspettative

Per una corretta implementazione di tale processo le aspettative sono le seguenti:

- realizzare un prodotto finale conforme alle richieste del proponente;
- realizzare un prodotto finale soddisfacente i test di verifica;
- realizzare un prodotto finale soddisfacente i test di validazione;
- fissare gli obiettivi di *sviluppo<sub>G</sub>*;
- fissare i vincoli tecnologici;
- fissare i vincoli di design.

### 2.2.3 Descrizione

Il processo di sviluppo si svolge in accordo con lo standard ISO/IEC 12207.

Pertanto si compone delle seguenti attività:

- Analisi dei requisiti
- Progettazione
- Codifica

### 2.2.4 Attività

#### 2.2.4.1 Analisi dei requisiti

##### 2.2.4.1.1 Scopo

Individuare ed elencare, evitando ambiguità, tutti i requisiti del capitolato. I requisiti possono essere estrapolati da più fonti:

- capitolato d'appalto;
- verbali di riunioni interne o esterne;

- casi d'uso.

Il risultato dell'attività è il documento chiamato *Analisi dei Requisiti v1.0.0*; esso è redatto dagli *Analisti* e contiene una lista dei requisiti e dei casi d'uso. I requisiti individuati permetteranno la definizione dei test di superamento dei requisiti del software in sviluppo.

#### **2.2.4.1.2 Aspettative**

Obiettivo dell'attività è la creazione della documentazione formale contenente tutti i requisiti richiesti dal proponente.

#### **2.2.4.1.3 Descrizione**

Nel documento *Analisi dei Requisiti v1.0.0* sono specificati tutti i requisiti analizzati con i metodi precedentemente riportati. Il tracciamento dei requisiti avviene tramite il software *Trender<sub>G</sub>*. La tecnica utilizzata per l'analisi e la ricerca dei requisiti è quella dei casi d'uso.

#### **2.2.4.1.4 Casi d'uso**

Ogni *caso d'uso<sub>G</sub>* è descritto dalla seguente struttura:

- Codice identificativo
- Titolo
- Diagramma UML
- Attori primari
- Attori secondari
- Scopo e descrizione
- Precondizione
- Postcondizione
- Flusso base degli eventi
- Inclusioni (se presenti)
- Estensioni (se presenti)

#### 2.2.4.1.5 Codice identificativo

Ogni caso d'uso è identificato da un codice, che segue il seguente formalismo:

$$UC\{\text{codice\_padre}\}.\{\text{codice\_livello}\}$$

Dove:

- **codice\_padre**: numero che identifica univocamente i casi d'uso;
- **codice\_livello**: numero progressivo che identifica i sottocasi. Può a sua volta includere altri livelli.

#### 2.2.4.1.6 Requisiti

Ogni requisito è strutturato come segue:

- Codice identificativo
- Descrizione
- Fonti

#### 2.2.4.1.7 Codice identificativo

Ogni requisito ha il seguente formalismo:

$$R\{X\}\{Y\}\{\text{codice\_padre}\}.\{\text{codice\_livello}\}$$

Dove:

- **X**: identifica uno dei seguenti tipi di requisito:
  - 1: requisito funzionale;
  - 2: requisito prestazionale;
  - 3: requisito qualitativo;
  - 4: vincolo progettuale.
- **Y**: identifica uno dei seguenti gradi di necessità:
  - O: requisito obbligatorio;
  - F: requisito facoltativo;
  - D: requisito desiderabile.
- **codice\_padre**: numero che identifica univocamente i requisiti;

- **codice\_livello:** numero progressivo che identifica i sottorequisiti. Può a sua volta includere altri livelli.

#### 2.2.4.1.8 UML

I diagrammi UML devono essere realizzati usando la versione del linguaggio *v2.0*.

#### 2.2.4.2 Progettazione

##### 2.2.4.2.1 Scopo

Questa attività definisce, in funzione dei requisiti specificati nell'*Analisi dei Requisiti*, tutte le caratteristiche essenziali del prodotto software richiesto. Essa ha come obiettivo la stesura dei seguenti documenti:

- *Specifica Tecnica*
- *Definizione di Prodotto*

##### 2.2.4.2.2 Aspettative

Il processo ha come risultato la redazione dei documenti sopra citati. Ciò permetterà coerenza ed affidabilità in funzione del prodotto finale.

##### 2.2.4.2.3 Descrizione

L'attività di progettazione deve rispettare i requisiti ed i vincoli stabiliti tra il gruppo e il proponente. Pertanto, vengono redatti i seguenti documenti:

- **Specifica Tecnica:** contiene tutte le specifiche riguardanti la progettazione ad alto livello del prodotto e delle sue componenti. Descrive inoltre i diagrammi UML utilizzati per la realizzazione dell'architettura e i test di verifica;
- **Definizione di Prodotto:** descrive in dettaglio la progettazione, integrando quanto riportato nella *Specifica Tecnica*. Specifica inoltre le definizioni delle classi e i diagrammi UML relativi. Definisce infine i test necessari alla verifica.

##### 2.2.4.2.4 Specifica Tecnica

Questo documento sarà scritto dal *Progettista<sub>G</sub>* e dovrà includere:

- **Diagrammi UML:**

- Diagrammi delle classi
- Diagrammi dei  $package_G$
- Diagrammi di attività
- Diagrammi di sequenza
- **Design pattern:** Devono essere descritti i  $design\ pattern_G$  utilizzati per realizzare l'architettura. Ogni design pattern deve essere accompagnato da una descrizione ed un diagramma, che ne esponga il significato e la struttura;
- **Tracciamento delle componenti:** ogni requisito deve riferirsi al componente che lo soddisfa. Nella sezione 2.2.5.1 viene descritto l'applicativo web *Trender*, utile a generare automaticamente le tabelle di tracciamento. Tramite questa operazione è possibile garantire che ogni requisito venga soddisfatto;
- **Test di integrazione:** devono essere definite delle classi di verifica volte a verificare che ogni componente del sistema funzioni nella maniera voluta.

#### 2.2.4.2.5 Definizione di Prodotto

Questo documento sarà scritto dal *Progettista* e dovrà includere:

- **Diagrammi UML:**
  - Diagrammi delle classi
  - Diagrammi di attività
  - Diagrammi di sequenza
- **Definizioni delle classi:** ogni  $classe_G$  deve essere descritta in modo da spiegarne in maniera esaustiva lo scopo e le funzionalità, evitando ridondanze;
- **Tracciamento delle classi:** ogni requisito deve essere tracciato in modo da garantire che ogni classe ne soddisfi almeno uno e poter risalire alle classi a esso associate. Nella sezione 2.2.5.1 viene descritto l'applicativo web *Trender*, utile anche a generare automaticamente le tabelle di tracciamento. Tramite questa operazione è possibile garantire che ogni classe soddisfi almeno un requisito.
- **Test di unità:** devono essere definiti dei test di  $unit\ _G$  in modo da verificare che le componenti del sistema funzionino nel modo stabilito.

#### 2.2.4.3 Codifica

#### 2.2.4.3.1 Scopo

Questa attività ha come scopo l'effettiva realizzazione del prodotto software richiesto. In questa fase si concretizza la soluzione attraverso la programmazione, in modo da ottenere il prodotto software finale.

#### 2.2.4.3.2 Aspettative

Obiettivo dell'attività è la creazione di un prodotto software conforme alle richieste prefissate con il proponente.

#### 2.2.4.3.3 Descrizione

L'attività deve rispettare quanto stabilito nel documento *Definizione di Prodotto*, rispettando al contempo le metriche definite nel *Piano di Qualifica v1.0.0*.

#### 2.2.4.3.4 Stile di codifica

Al fine di garantire uniformità nel codice del progetto ciascun membro del gruppo è tenuto a rispettare le seguenti norme:

- **Indentazione:** è richiesto l'utilizzo di esattamente una tabulazione;
- **Parentesi dei costrutti:** è richiesto di inserire le parentesi di delimitazione dei costrutti in linea e non al di sotto di essi;
- **Nomi:** i nomi di variabili, metodi e funzioni devono avere la prima lettera minuscola. I nomi delle classi devono avere la prima lettera maiuscola.

#### 2.2.4.3.5 Intestazione

Ogni *file<sub>G</sub>* contenente codice deve avere la seguente intestazione:



```
/*
* File: nome file
* Version: versione file
* Type: tipo file
* Date: data di creazione
* Author: nome autore/i
* E-mail: email autore/i
*
* License: tipo licenza
*
* Avvertenze: lista avvertenze e limitazioni
*
* Registro modifiche:
* Autore || Data || breve descrizione modifiche
*
*/
```

#### 2.2.4.3.6 Versionamento

La versione del codice viene inserita all'interno dell'intestazione del file e rispetta il seguente formalismo:

**X.Y**

- **X**: è l'indice di versione principale, un incremento di tale indice rappresenta un avanzamento della versione stabile, che porta il valore dell'indice Y ad essere azzerato;
- **Y**: è l'indice di modifica parziale, un incremento di tale indice rappresenta una verifica o una modifica rilevante, come per esempio la rimozione o l'aggiunta di una istruzione.

La versione *1.0* deve rappresentare la prima versione del file completo e stabile, cioè quando le sue funzionalità obbligatorie sono state definite e si considerano funzionanti. Solo dalla versione *1.0* è possibile testare il file, con degli appositi test definiti, per verificarne l'effettivo funzionamento.

#### 2.2.4.3.7 Ricorsione

La ricorsione deve essere evitata ove possibile. Di ogni funzione ricorsiva è richiesta la prova di terminazione e l'analisi del costo in termini di memoria. Nel caso in cui la memoria utilizzata risulti eccessiva la ricorsione deve essere rimossa.

## 2.2.5 Strumenti

Di seguito sono elencati gli strumenti utilizzati dal gruppo durante il progetto.

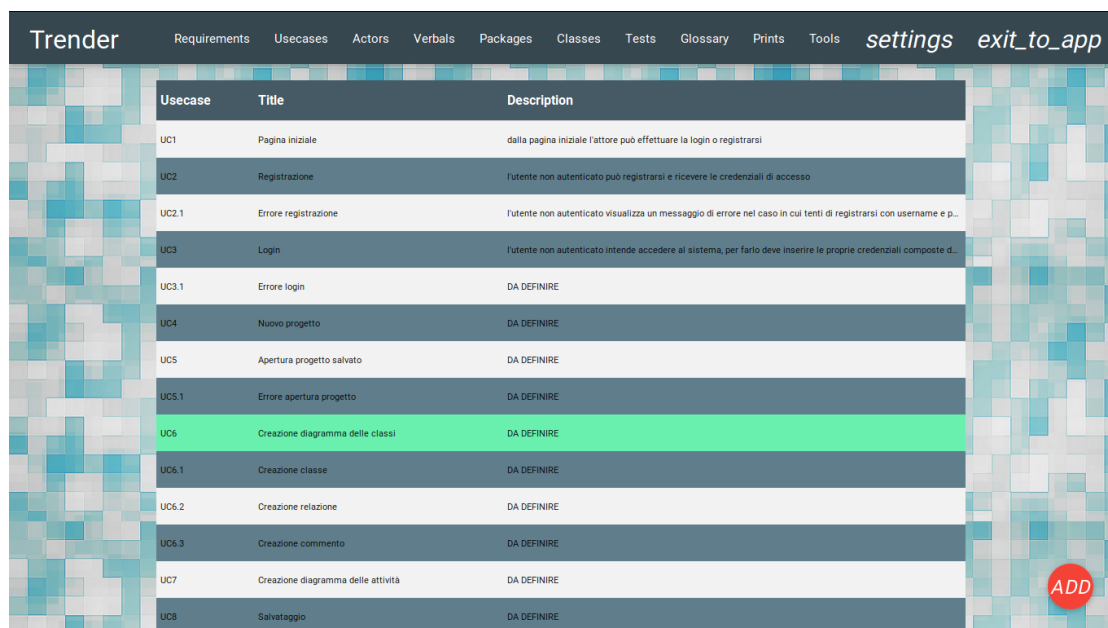
### 2.2.5.1 Trender

Il gruppo utilizza l'applicativo web Trender per gestire in maniera veloce e automatizzata tutti i dati ricavati dall'analisi dei requisiti.

Ogni componente del gruppo può accedervi, utilizzando l'account comune predisposto dall'*Amministratore<sub>G</sub>* di Progetto. Il *database<sub>G</sub>* viene gestito tramite *MySQL<sub>G</sub>*, mentre il resto del sito utilizza JavaScript e *PHP<sub>G</sub>*.

Le funzioni offerte da tale applicativo sono:

- Tracciamento dei requisiti
- Tracciamento dei casi d'uso
- Tracciamento dei verbali
- Tracciamento degli attori presenti nel sistema
- Tracciamento dei packages
- Tracciamento delle classi
- Tracciamento dei test
- Possibilità di stampare direttamente in codice  $\text{\LaTeX}$  quanto archiviato



The screenshot shows the Trender application interface. At the top, there is a navigation bar with the following tabs: Requirements, Usecases, Actors, Verbals, Packages, Classes, Tests, Glossary, Prints, Tools, settings, and exit\_to\_app. Below the navigation bar is a table with three columns: Usecase, Title, and Description. The table contains 15 rows of use cases. The row for UC6, 'Creazione diagramma delle classi', is highlighted in green. A red 'ADD' button is visible in the bottom right corner of the table area.

Usecase	Title	Description
UC1	Pagina iniziale	dalla pagina iniziale l'attore può effettuare la login o registrarsi
UC2	Registrazione	l'utente non autenticato può registrarsi e ricevere le credenziali di accesso
UC2.1	Errore registrazione	l'utente non autenticato visualizza un messaggio di errore nel caso in cui tenti di registrarsi con username e p...
UC3	Login	l'utente non autenticato intende accedere al sistema, per farlo deve inserire le proprie credenziali composte d...
UC3.1	Errore login	DA DEFINIRE
UC4	Nuovo progetto	DA DEFINIRE
UC5	Apertura progetto salvato	DA DEFINIRE
UC5.1	Errore apertura progetto	DA DEFINIRE
UC6	Creazione diagramma delle classi	DA DEFINIRE
UC6.1	Creazione classe	DA DEFINIRE
UC6.2	Creazione relazione	DA DEFINIRE
UC6.3	Creazione commento	DA DEFINIRE
UC7	Creazione diagramma delle attività	DA DEFINIRE
UC8	Salvataggio	DA DEFINIRE

Figura 1: Trender

### 2.2.5.2 Astah

Per la produzione dei diagrammi UML viene utilizzato *Astah Professional Edition<sub>G</sub>* versione 7.2, in quanto offre molte agevolazioni per la produzione veloce dei diagrammi e risulta semplice da usare.

<http://astah.net/>

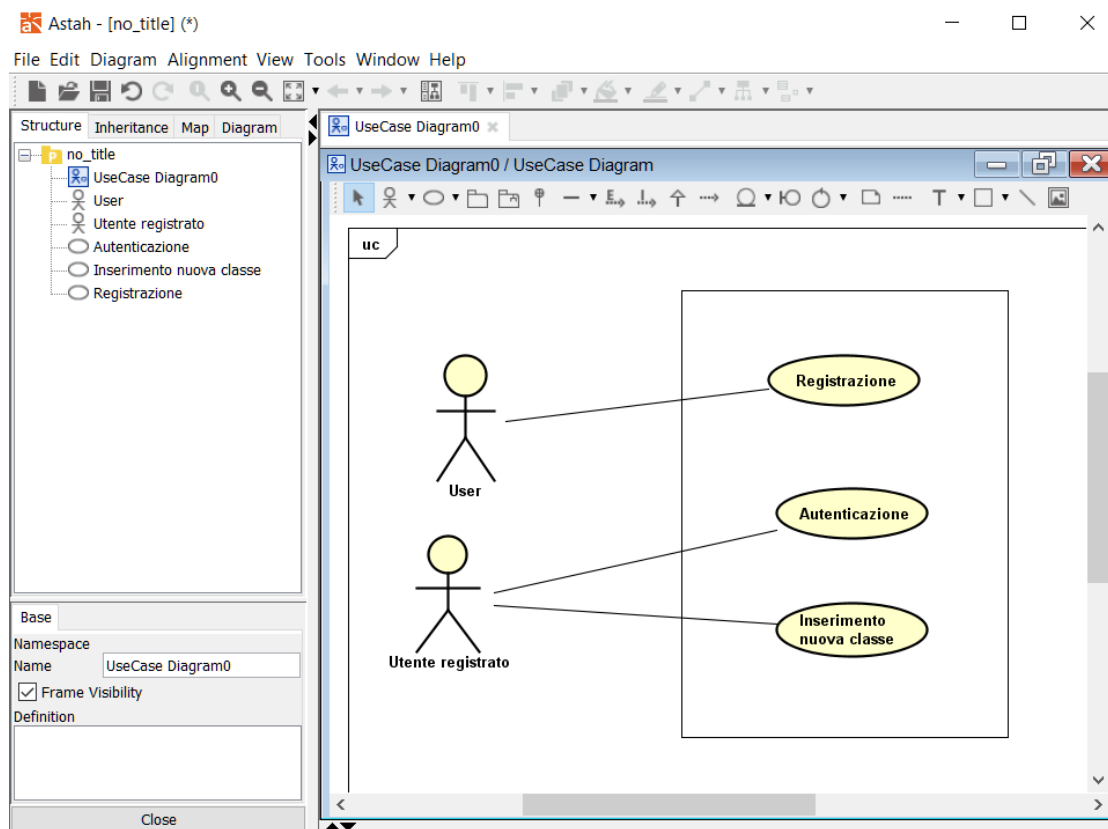


Figura 2: Astah Desktop per Windows

### 2.2.5.3 IntelliJ IDEA

IntelliJ IDEA viene utilizzato per la codifica in Java e JavaScript. Questo  $IDE_G$  offre piena compatibilità con  $Linux_G$ ,  $Windows_G$  e  $macOS_G$ , oltre ad essere un potente editor con molte funzionalità integrate.

<https://www.jetbrains.com/idea/>

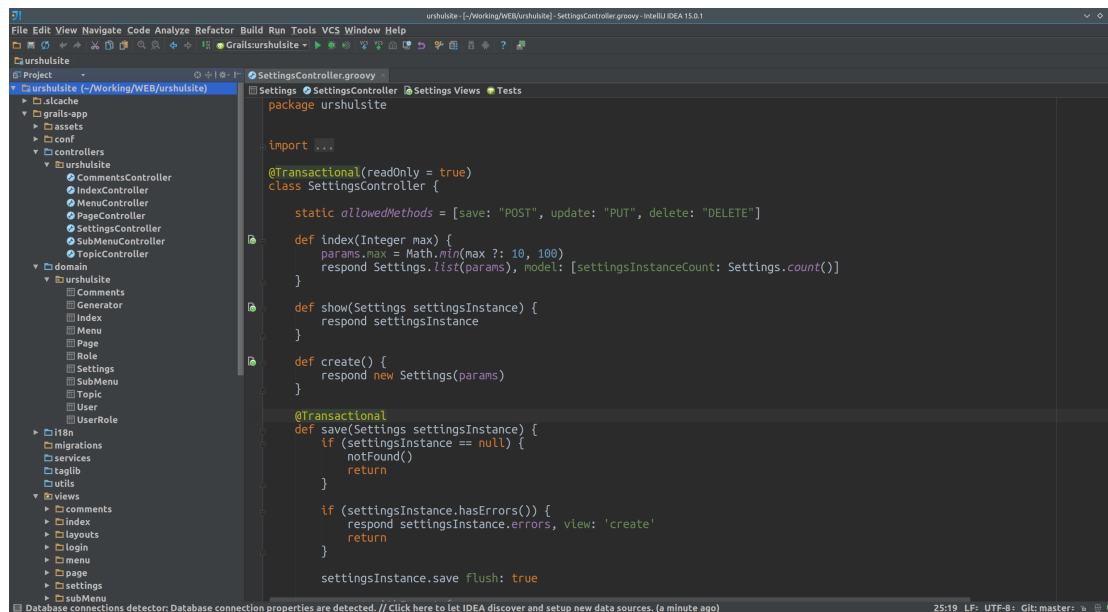


Figura 3: IntelliJ IDEA Desktop per Windows

## 3 Processi di supporto

### 3.1 Documentazione

#### 3.1.1 Scopo

Questo processo include i dettagli su come deve essere redatta e mantenuta la documentazione durante il ciclo di vita del software.

#### 3.1.2 Aspettative

Le aspettative della corretta implementazione di tale processo sono:

- una visione precisa della documentazione che va prodotta durante il ciclo di vita del software;
- l'individuazione di una serie di norme per la stesura di documenti coerenti e validi;
- la stesura di una documentazione formale e coerente.

#### 3.1.3 Descrizione

In questa sezione devono essere indicate tutte le norme e le convenzioni adottate dal gruppo, per consentire la stesura di una documentazione valida e coerente.

#### 3.1.4 Procedure

Per la stesura della documentazione il gruppo ha utilizzato il linguaggio  $\text{\LaTeX}$ .

##### 3.1.4.1 Approvazione dei documenti

Ogni documento non formale di cui sia stata completata la stesura dovrà essere sottoposto al *Responsabile* di Progetto, che a sua volta si occuperà di incaricare i *Verificatori* di controllarne il contenuto e la forma. Nel caso tali *Verificatori* trovino degli errori, sarà loro compito riportarli al *Responsabile* di Progetto, che a sua volta incaricherà il redattore del documento di correggerli.

Questo ciclo va ripetuto fino a che il documento non è considerato completamente corretto dai *Verificatori*. A tal punto sarà sottoposto al *Responsabile* di Progetto, che potrà approvarlo o meno. Se approvato, il documento è da considerarsi come un documento formale. In caso contrario il *Responsabile* di Progetto dovrà comunicare le motivazioni per cui il documento non è stato approvato, esplicitando le modifiche da apportare.

### 3.1.5 Template

Il gruppo ha creato un template L<sup>A</sup>T<sub>E</sub>X per uniformare velocemente la struttura grafica e lo stile di formattazione dei documenti, in modo che i membri del gruppo debbano concentrarsi solo sulla stesura del contenuto e non sul suo aspetto.

### 3.1.6 Struttura dei documenti

#### 3.1.6.1 Prima pagina

La prima pagina di ogni documento è così strutturata:

- **Logo del gruppo:** visibile come primo elemento centrato orizzontalmente in alto;
- **Titolo:** nome del documento, visibile subito dopo il logo e centrato orizzontalmente;
- **Gruppo e progetto:** nome del gruppo e del progetto, appena sotto il titolo del documento e centrato orizzontalmente;
- **Recapito:** indirizzo di posta elettronica del gruppo, centrato orizzontalmente sotto il nome del gruppo e del progetto;
- **Tabella descrittiva:** visibile subito dopo il titolo, centrata orizzontalmente e contenente le seguenti informazioni:
  - versione;
  - nome e cognome dei membri del gruppo incaricati della redazione 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;
  - tipo di uso;
  - destinatari del documento.
- **Descrizione:** centrata orizzontalmente ed il più possibile sintetica.

#### 3.1.6.2 Registro delle modifiche

Ogni documento, eccezion fatta per i verbali, deve contenere questo registro a seguito della prima pagina. Tale registro deve contenere le modifiche apportate al documento stesso, indicando per ognuna:

- versione del documento dopo la modifica;
- data della modifica apportata;
- nome e cognome della persona coinvolta nella modifica;
- ruolo ricoperto dalla persona coinvolta nella modifica;
- descrizione concisa della modifica apportata.

### **3.1.6.3 Indice**

Ogni documento, eccezion fatta per i verbali, deve avere un indice che ne agevoli la consultazione e permetta una lettura *ipertestuale*<sub>G</sub> e non necessariamente sequenziale. La numerazione di ogni indice deve partire da 1; ciascuna sottosezione deve essere separata dalla sezione padre tramite un punto, e la numerazione deve ripartire di volta in volta da 1.

### **3.1.6.4 Contenuto principale**

I margini orizzontali e verticali previsti dal template devono essere rispettati in ogni pagina. Ad eccezione della prima, tutte le pagine devono contenere un'intestazione ed un piè di pagina. L'intestazione è così strutturata:

- Logo del gruppo posto a sinistra;
- Indirizzo di posta elettronica del gruppo posto a destra;

Il piè di pagina è così strutturato:

- Nome e versione del documento corrente, posti a sinistra;
- Numerazione progressiva della pagina rispetto al totale posta a destra.

### **3.1.6.5 Note a piè di pagina**

In caso di presenza in una pagina interna di note da esplicitare, esse vanno indicate nella pagina corrente, in basso a sinistra. Ogni nota deve riportare un numero e una descrizione.

### **3.1.7 Versionamento**

Ogni documento, eccezion fatta per i verbali, deve essere versionato, in modo che sia possibile avere una visione specifica della sua storia e delle sue modifiche. Ad ogni



versione deve corrispondere una riga nel registro delle modifiche.

Il formalismo da applicare è il seguente:

$$v\{X\}.\{Y\}.\{Z\}$$

dove:

- **X:**
  - Inizia da 0;
  - Viene incrementato da parte del *Responsabile* di Progetto all’approvazione del documento;
  - È limitato superiormente dal numero di revisioni.
- **Y**
  - Inizia da 0;
  - Viene incrementato da parte del *Verificatore<sub>G</sub>* ad ogni verifica;
  - Non è limitato superiormente;
  - Quando viene incrementato X, viene riportato a 0.
- **Z**
  - Inizia da 0;
  - Viene incrementato da parte del redattore del documento ad ogni modifica;
  - Non è limitato superiormente;
  - Quando viene incrementato Y, viene riportato a 0.

### 3.1.8 Norme tipografiche

#### 3.1.8.1 Stile del testo

- **Glossario:** ogni parola contenuta nel glossario deve essere marcata, alla sua prima occorrenza in ogni documento, in carattere corsivo e con una *G* maiuscola a pedice:

*repository<sub>G</sub>*;

- **Grassetto:** viene applicato ai titoli e agli elementi di un elenco puntato che riassumono il contenuto di tale voce;
- **Corsivo:** Il corsivo dev’essere utilizzato per le seguenti occorrenze:
  - citazioni;
  - parole inserite nel glossario;

- attività del progetto;
  - ruoli del progetto;
  - riferimenti ad altri documenti;
  - parole particolari solitamente poco usate o conosciute.
- **Maiuscolo:** le uniche parole che è consentito scrivere interamente in maiuscolo sono gli acronimi.

### 3.1.8.2 Elenchi puntati

Gli elenchi puntati vengono rappresentati graficamente da un *pallino* nel primo livello, da un *trattino* nel secondo e da un *asterisco* nel terzo.

Gli elenchi puntati servono ad esprimere in modo sintetico un concetto, evitando frasi lunghe e discorsive. Ogni voce di un elenco puntato deve terminare con un punto e virgola, ad eccezione dell'ultima, che va terminata con un punto. Questa regola può non venire applicata per enfatizzare concetti relativamente corti.

### 3.1.8.3 Formati comuni

Per le seguenti tipologie di concetto vengono applicati i seguenti formalismi:

- **Date:**

**GG-MM-AAAA**

- **GG:** rappresenta il giorno rappresentato utilizzando due cifre;
- **MM:** rappresenta il mese rappresentato utilizzando due cifre;
- **AAAA:** rappresenta l'anno rappresentato utilizzando quattro cifre.

- **Orari:**

**HH:MM**

- **HH:** rappresenta l'ora e può assumere valori da 0 a 23;
- **MM:** rappresenta i minuti e può assumere valori da 0 a 59.

- **Nomi ricorrenti:**

- **Ruoli di progetto:** ogni nome di ruolo di progetto viene scritto in corsivo e con l'iniziale maiuscola;
- **Nomi dei documenti:** ogni nome di documento viene scritto in corsivo e con l'iniziale di ogni parola che non sia un articolo maiuscola;

- **Nomi propri:** ogni nome proprio di persona deve essere scritto nella forma *Nome Cognome*.

#### 3.1.8.4 Sigle

È previsto l'utilizzo delle seguenti sigle:

- **AR:** *Analisi dei Requisiti*;
- **PP:** *Piano di Progetto*;
- **NP:** *Norme di Progetto*;
- **SF:** *Studio di Fattibilità*;
- **PQ:** *Piano di Qualifica*;
- **ST:** *Specifica Tecnica*;
- **MU:** *Manuale utente<sub>G</sub>*;
- **DP:** *Definizione di Prodotto*;
- **RR:** *Revisione dei requisiti*;
- **RP:** *Revisione di progettazione*;
- **RQ:** *Revisione di qualifica*;
- **RA:** *Revisione di accettazione*;
- **Re:** *Responsabile di Progetto*;
- **Am:** *Amministratore di Progetto*;
- **An:** *Analista*;
- **Pt:** *Progettista*;
- **Pr:** *Programmatore<sub>G</sub>*;
- **Ve:** *Verificatore*.

#### 3.1.9 Elementi grafici

##### 3.1.9.1 Tabelle

Ogni tabella deve essere centrata orizzontalmente nella pagina e deve presentare sotto di essa la propria didascalia; in tale didascalia deve comparire il numero della tabella, incrementale in tutto il documento per agevolarne il tracciamento, oltre che una breve descrizione del suo contenuto.

Fanno eccezione le tabelle del registro delle modifiche che non hanno nessuna descrizione e le tabelle dei casi d'uso presenti nell'*Analisi dei requisiti v1.0.0* che hanno anche un diverso layout.

Al fine di aumentare la leggibilità delle tabelle, nelle celle contenenti uno 0 (zero) che non sia significativo per la comprensione della tabella stessa verrà inserito un trattino (-). Qualora il redattore lo ritenesse necessario per aumentare la leggibilità è permesso anche l'uso di colori di sfondo.

### **3.1.9.2 Immagini**

Ogni immagine deve essere centrata orizzontalmente ed essere nettamente separata dai paragrafi che la seguono e la precedono, per marcare un netto distacco tra testo e grafica e migliorare conseguentemente la leggibilità. Le immagini devono essere accompagnate da una didascalia analoga a quella descritta per le tabelle. Tutti i diagrammi UML vengono inseriti nei documenti sotto forma di immagine.

## **3.1.10 Classificazione dei documenti**

### **3.1.10.1 Documenti informali**

Tutte le versioni dei documenti che non siano state direttamente approvate da parte del *Responsabile* di Progetto sono ritenute informali e in quanto tali sono considerate esclusivamente ad uso interno.

### **3.1.10.2 Documenti formali**

Una versione di un documento viene definita formale quando è stata validato dal *Responsabile* di Progetto. Solo i documenti formali possono essere distribuiti all'esterno del gruppo. Per arrivare a tale stato il documento deve aver già superato la verifica e la validazione.

### **3.1.10.3 Verbali**

Un verbale è un documento redatto da un segretario in occasione di incontri interni al gruppo o con altre entità esterne. I verbali non subiscono modifiche successive alla loro prima redazione, pertanto non prevedono *versionamento*. Ogni verbale deve essere approvato dal *Responsabile* di Progetto e deve indicare nel seguente ordine e con il formato indicato:

- **Luolo:** Città (Provincia), Via, Sede;
- **Data:** dd-mm-yyyy;

- **Ora:** hh-mm 24h;
- **Partecipanti del gruppo;**
- **Partecipanti esterni** (se presenti).

Tali informazioni devono essere contenute nel primo paragrafo *Informazioni Generali*, che deve specificare anche gli argomenti trattati durante l'incontro.

Il secondo paragrafo si distingue invece a seconda del tipo di verbale:

- **Interno:** deve contenere la sezione *Riassunto Incontro*, dove vengono esplicitati più in dettaglio gli argomenti trattati;
- **Esterno:** deve contenere la sezione *Domande e Risposte*, dove vengono riportate le domande poste ai partecipanti esterni, seguite dalle loro risposte.

Ogni verbale ha un codice univoco identificativo con il seguente formalismo:

$$V\{X\}_{Y}$$

Dove:

- **X:** identifica uno dei seguenti tipi di verbale:
  - *E*: verbale esterno
  - *I*: verbale interno
- **Y:** identifica la data nel formato YYYYMMDD.

Per facilitare il tracciamento delle decisioni emerse da ogni incontro, sia interno che esterno, è richiesta una tabella riassuntiva alla fine di ogni verbale. Tali decisioni sono tracciate con il seguente formalismo:

$$\text{CodiceVerbale.}\{X\}$$

Dove:

- **CodiceVerbale:** codice del verbale, come sopra indicato;
- **X:** numero progressivo che identifica le decisioni prese, partendo da 1.

### 3.1.11 Strumenti

#### 3.1.11.1 $\text{\LaTeX}$

Per la stesura della documentazione il gruppo ha utilizzato il linguaggio  $\text{\LaTeX}$  per via delle possibilità che esso offre:

- creazione di documenti formali e divisi in sezioni molto velocemente;

- separazione del contenuto dalla formattazione tramite un file template separato e condiviso da tutti i documenti;
- personalizzazione del documento grazie all'elevato numero di librerie.

Inoltre è stata redatta una guida  $\text{\LaTeX}$  nella quale vengono descritti dei comandi ad uso interno per facilitare la stesura dei documenti e di conseguenza standardizzarne la forma. Per maggiori dettagli sul loro funzionamento si rimanda a *Guida ai comandi  $\text{\LaTeX}$  v1.0.0*.

### 3.1.11.2 TexStudio

Per la stesura del codice  $\text{\LaTeX}$  è stato utilizzato l'editor *TexStudio<sub>G</sub>* nella versione 2.12.2. Questo strumento oltre ad integrare un compilatore e visualizzatore PDF, fornisce suggerimenti per completare i comandi di  $\text{\LaTeX}$ .

<http://www.texstudio.org/>

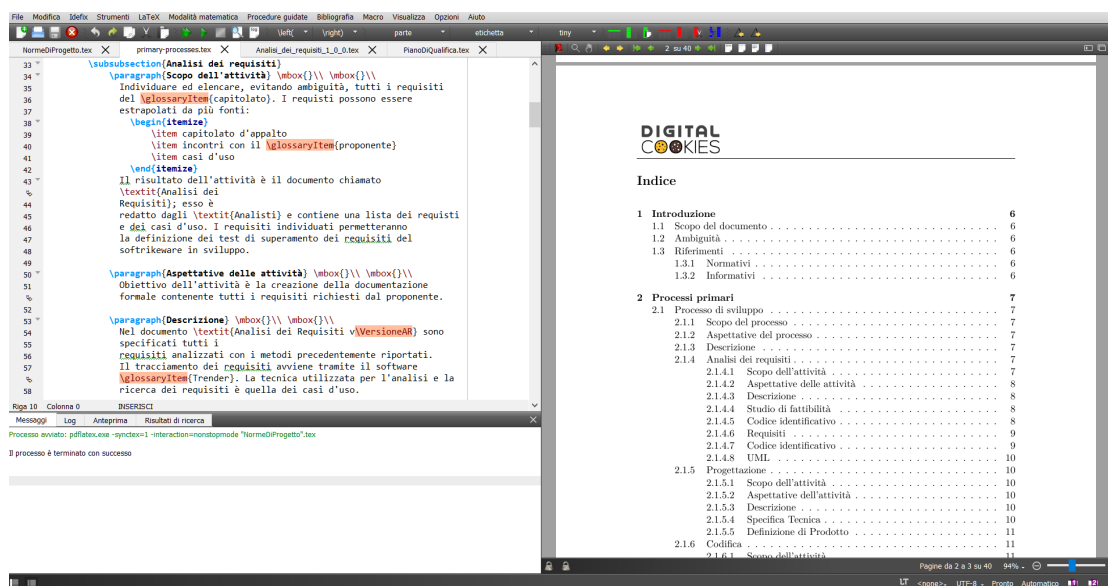


Figura 4: TexStudio Desktop per Windows

### 3.1.11.3 Lucidchart

Per la realizzazione di diagrammi illustrativi per i documenti viene utilizzata la piattaforma web *Lucidchart<sub>G</sub>*. Questo strumento è versatile e portatile, essendo utilizzabile direttamente via web.

<https://www.lucidchart.com/>

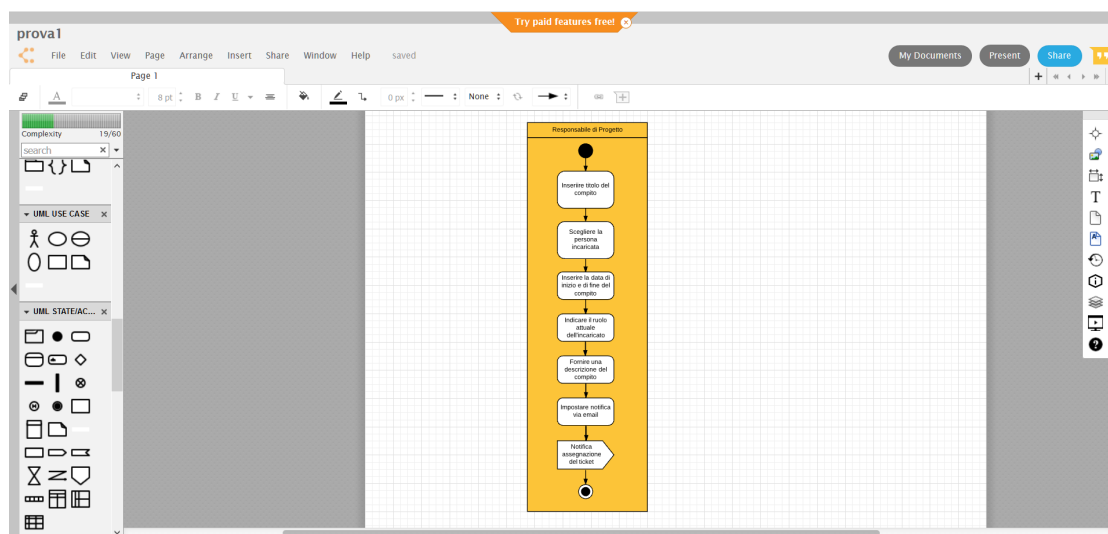


Figura 5: Lucidchart

## 3.2 Verifica

### 3.2.1 Scopo

Si occupa di accertare che non vengano introdotti errori nel prodotto a seguito dell'esecuzione delle attività dei processi svolti nella fase in esame.

### 3.2.2 Aspettative

Una corretta implementazione di tale processo permette di individuare:

- una procedura di verifica;
- i criteri per la verifica del prodotto;
- i difetti, da catalogare per essere corretti.

### 3.2.3 Descrizione

Il processo è suddiviso in due attività:

- **Analisi:** consiste nell'analisi del codice sorgente e la sua successiva esecuzione. Viene effettuata tramite due tecniche, l'analisi statica e l'analisi dinamica;
- **Test:** definisce tutti i test che vengono eseguiti sul prodotto software.

### 3.2.4 Attività

#### 3.2.4.1 Analisi

##### 3.2.4.1.1 Analisi statica

L'analisi statica è una tecnica che permette di individuare anomalie all'interno di documenti e codice sorgente durante tutto il loro ciclo di vita. È applicabile tramite due tecniche diverse:

- **Walkthrough:** viene svolta effettuando una lettura a largo spettro. Si tratta di un'attività onerosa e collaborativa che richiede la cooperazione di più persone, essendo una tecnica non efficiente. Verrà utilizzata principalmente durante la prima parte del progetto, quando non tutti i membri del gruppo hanno piena padronanza e conoscenza delle *Norme di Progetto* e del *Piano di Qualifica*. Utilizzando questa tecnica è possibile stilare una lista di controllo contenente gli errori più comuni.
- **Inspection:** viene svolta una lettura mirata e strutturata, volta a localizzare gli errori segnalati nella lista di controllo, con il minor costo possibile. Tramite l'acquisizione di esperienza la lista di controllo viene progressivamente estesa, rendendo l'inspection via via più efficace. Normalmente è effettuata da una persona sola.

##### 3.2.4.1.2 Analisi dinamica

L'analisi dinamica è una tecnica di analisi del prodotto software che richiede la sua esecuzione.

Viene effettuata mediante dei test volti a verificare il funzionamento del prodotto e nel caso in cui vengano riscontrate anomalie ne permette l'identificazione.

I test devono essere ripetibili, cioè deve essere possibile, dato lo stesso input e nello stesso ambiente, risalire allo stesso output. Per ogni test devono dunque essere definiti i seguenti parametri:

- **Ambiente:** il sistema hardware e software sul quale verrà eseguito il test del prodotto;
- **Stato iniziale:** lo stato iniziale dal quale il test viene eseguito;
- **Input:** l'input inserito;
- **Output:** l'output atteso;
- **Istruzioni aggiuntive:** ulteriori istruzioni su come va eseguito il test e su come vanno interpretati i risultati ottenuti.



### **3.2.4.2 Test**

#### **3.2.4.2.1 Test di unità**

Il test di unità si pone come obiettivo primario l'isolare dal resto del codice la parte più piccola di software testabile nell'applicazione, chiamata unità, per stabilire se essa funziona esattamente come previsto. Ogni unità deve essere sottoposta a test prima di integrarla in modulo per l'esecuzione del test delle interfacce tra i diversi moduli.

L'approccio più comune al test di unità necessita della scrittura di *driver<sub>G</sub>* e *stub<sub>G</sub>*, dove il driver simula un'unità chiamante mentre lo stub simula un'unità chiamata.

#### **3.2.4.2.2 Test di integrazione**

Il test di integrazione rappresenta l'estensione logica del test di unità. La forma più semplice di questo tipo di test prevede la combinazione di due unità già sottoposte a test in un unico componente e il test dell'interfaccia presente tra le due. Il concetto alla base in questo approccio consiste nell'esecuzione di test per la combinazione di più parti, espandendo progressivamente il processo al test di moduli di un gruppo con quelli di altri gruppi. L'obiettivo finale è di sottoporre al test tutti i moduli che compongono un processo contemporaneamente.

#### **3.2.4.2.3 Test di sistema**

Il test di sistema sancisce la validazione del prodotto software finale, giunto ad una versione definitiva, e verifica dunque che esso soddisfi in modo completo i requisiti.

#### **3.2.4.2.4 Test di regressione**

Il test di regressione deve essere eseguito ad ogni modifica di un'implementazione del sistema. A tal fine è necessario eseguire sul codice modificato i test esistenti, in modo da stabilire se le modifiche apportate hanno alterato elementi precedentemente funzionanti.

#### **3.2.4.2.5 Test di accettazione**

Il test di accettazione prevede il collaudo del prodotto in presenza del proponente e, in caso del superamento di tale collaudo, ne consegue il rilascio ufficiale del prodotto sviluppato.

### 3.2.5 Strumenti

#### 3.2.5.1 Verifica ortografica

Viene utilizzata la verifica dell'ortografia in tempo reale, strumento integrato in TexStudio che sottolinea in rosso le parole errate secondo la lingua italiana.

#### 3.2.5.2 Validazione W3C

Per la validazione delle pagine di markup HTML viene utilizzato lo strumento offerto dal  $W3C_G$ , raggiungibile al seguente indirizzo:

<https://validator.w3.org/>

Per la validazione dei fogli di stile  $CSS_G$  viene utilizzato lo strumento offerto dal W3C, raggiungibile al seguente indirizzo:

<https://jigsaw.w3.org/css-validator/>

#### 3.2.5.3 Analisi statica

Per l'analisi statica del codice JavaScript vengono utilizzati i seguenti strumenti:

- **JSHint**: uno strumento *OpenSource<sub>G</sub>* atto a rilevare gli errori e i potenziali problemi nel codice JavaScript, oltre che ad imporre delle convenzioni di codifica al team. JSHint è accessibile al seguente indirizzo:

<http://www.jshint.com/>

- **Closure Compiler**: uno strumento aggiuntivo a JSHint, che permette di compilare il codice JavaScript e analizzarlo alla ricerca di errori. Closure Compiler è accessibile al seguente indirizzo:

<https://closure-compiler.appspot.com/home>

#### 3.2.5.4 Analisi dinamica

Per l'esecuzione dei test di analisi dinamica vengono utilizzati i seguenti strumenti:

- **Mocha**: un *framework<sub>G</sub>* ricco di funzionalità per l'esecuzione di test JavaScript, scritto in *Node.js<sub>G</sub>*; permette l'esecuzione di test asincroni e in serie, consentendo segnalazioni flessibili e accurate. Mocha è raggiungibile al seguente indirizzo:

<http://mochajs.org/>

- **Karma:** un ambiente di testing, utile ad effettuare test su browser e dispositivi. Per descrivere i test vengono utilizzati dei framework esterni, come per esempio Mocha, assieme a cui viene utilizzato per l'esecuzione dei test di unità. Karma è raggiungibile al seguente indirizzo:

<http://karma-runner.github.io/0.13/index.html>

### 3.2.5.5 Metriche

Per il calcolo delle varie metriche vengono utilizzati i seguenti strumenti:

- **JSMeter:** uno strumento che permette di calcolare diversi indici della complessità del codice JavaScript. Viene utilizzato per calcolare volume e potenziale di  $Halstead_G$  per funzione e l'indice di *manutenibilità*<sub>G</sub>. JSMeter è accessibile al seguente indirizzo:

<http://jmeter.info/>

- È stato inoltre utilizzato per calcolare l'indice  $Gulpease_G$  uno strumento disponibile al seguente indirizzo:

[http://farfalla-project.org/readability\\_static/](http://farfalla-project.org/readability_static/)

## 4 Processi organizzativi

### 4.1 Gestione

#### 4.1.1 Scopo

Lo scopo di questo processo è la creazione del documento *Piano di Progetto*, utile ai membri del gruppo per organizzare e gestire i ruoli di ogni componente.

#### 4.1.2 Aspettative

Le aspettative del processo sono:

- produzione del documento *Piano di Progetto*;
- definizione ruoli dei membri del gruppo;
- definizione di un piano per l'esecuzione dei compiti programmati.

#### 4.1.3 Descrizione

Orari di lavoro:

- dalle 9.00 alle 13.00;
- dalle 14.00 alle 18.00.

Viene trattata la gestione dei seguenti argomenti:

- Ruoli di progetto
- Comunicazioni
- Incontri
- Strumenti di coordinamento
- Strumenti di versionamento
- Rischi

#### 4.1.4 Ruoli di progetto

Tutti i ruoli saranno ricoperti da ciascun componente del gruppo in rotazione, in modo che ogni membro possa assumere almeno una volta ciascuno di essi. Nel documento *Piano di Progetto v1.0.0* vengono organizzate e pianificate le attività assegnate ai specifici ruoli previsti nell'attività di progetto, che sono:

#### 4.1.4.1 Amministratore di Progetto

L'*Amministratore* di Progetto controlla e amministra tutto l'ambiente di lavoro con piena responsabilità sulla capacità operativa e sull'efficienza.

Le sue responsabilità sono:

- ricerca di strumenti che migliorino l'ambiente di lavoro e che lo automatizzino ove possibile;
- gestione del versionamento;
- controllo di versioni e configurazioni del prodotto software;
- risoluzione dei problemi di gestione dei processi;
- controllo della *qualità<sub>G</sub>* sul prodotto.

#### 4.1.4.2 Responsabile di Progetto

Il *Responsabile* di Progetto è il punto di riferimento sia per il *committente<sub>G</sub>* che per il fornitore. Inoltre, approva le scelte prese dal gruppo e se ne assume la responsabilità.

Le responsabilità di tale ruolo sono:

- approvazione della documentazione;
- approvazione dell'offerta economica;
- gestione delle risorse umane;
- coordinamento e pianificazione delle attività di progetto;
- studio e gestione dei rischi.

#### 4.1.4.3 Analista

L'*Analista* si occupa dell'analisi dei problemi e del dominio applicativo. La sua presenza non è sempre necessaria durante il progetto.

Le sue responsabilità sono:

- comprensione del problema e della sua complessità;
- produzione dello *Studio di Fattibilità* e dell'*Analisi dei Requisiti*.

#### 4.1.4.4 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;
- rendere facilmente mantenibile il progetto.

#### 4.1.4.5 Verificatore

Il *Verificatore* avendo delle solide conoscenze delle normative di progetto garantirà una esaustiva verifica dello esso.

Le responsabilità assunte da tale ruolo sono:

- controllo delle attività di progetto secondo le normative stabilite.

#### 4.1.4.6 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*;
- scrittura di un codice pulito e facile da mantenere, che rispetti le *Norme di Progetto*;
- versionamento del codice prodotto;
- realizzazione degli strumenti per la verifica e la validazione del software.

#### 4.1.5 Procedure

##### 4.1.5.1 Gestione delle comunicazioni

###### 4.1.5.1.1 Comunicazioni interne

Le comunicazioni interne sono gestite utilizzando un tool denominato *Slack<sub>G</sub>*. Questo servizio offre al suo interno una suddivisione dei canali per lo scambio di informazioni tra i componenti del gruppo. Ogni canale è specifico per una determinata attività. All'interno di ogni canale possono essere creati dei *thread<sub>G</sub>* privati tra due membri del gruppo. Molto più comodo rispetto ad un servizio di semplice messaggistica istantanea come Telegram, Slack offre un ambiente puramente creato per gestire al meglio un gruppo di lavoro.

###### 4.1.5.1.2 Comunicazioni esterne

Le comunicazioni esterne sono affidate al *Responsabile* di Progetto, il quale si avvale di



una apposita casella di posta elettronica:

digitalcookies.group@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.

#### **4.1.5.2 Gestione degli incontri**

##### **4.1.5.2.1 Incontri Interni**

Il *Responsabile* di progetto ha il compito di organizzare gli incontri interni utilizzando i canali di comunicazione. Il *Responsabile* dovrà essere certo della partecipazione di ogni membro del gruppo, al fine di fissare definitivamente l'evento sulla piattaforma Slack. Ogni componente ha diritto di presentare una richiesta al *Responsabile* di Progetto di organizzazione di un incontro, il quale dovrà decidere se accettare o meno la proposta. Inoltre, il *Responsabile* dovrà incaricare un membro del gruppo a stendere il verbale dell'incontro avvenuto secondo le norme previste.

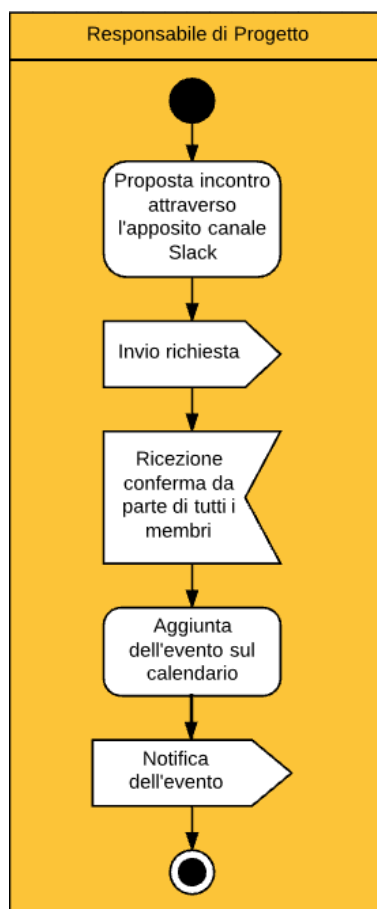


Figura 6: Organizzazione di un incontro interno

#### 4.1.5.2.2 Incontri Esterni

Il *Responsabile* di Progetto ha il compito di comunicare, ed organizzare gli incontri esterni con il proponente o committente. Come per gli incontri interni, ogni componente del gruppo ha diritto ad una richiesta di organizzazione di un incontro esterno. Il *Responsabile* di Progetto deciderà se organizzare o meno l'incontro una volta accertati i motivi della richiesta. Nel caso decida di procedere a contattare l'entità esterna (proponente o committente), e quest'ultima sia d'accordo, deve comunicare tutte le informazioni riguardanti data, ora e luogo dell'incontro a tutti i componenti del gruppo per mezzo dei canali di comunicazione interni. Inoltre, il *Responsabile* dovrà incaricare un membro del gruppo a stendere il verbale dell'incontro avvenuto secondo le norme previste.





Figura 7: Organizzazione di un incontro esterno

#### 4.1.5.3 Gestione degli strumenti di coordinamento

##### 4.1.5.3.1 Ticketing

Per suddividere il carico di lavoro in  $Task_G$  che saranno divisi tra tutti i componenti del gruppo viene utilizzata la piattaforma  $Wrike_G$ . Questo compito spetta al *Responsabile di Progetto*. Wrike permette di avere un quadro completo delle attività, delle relative scadenze prefissate e dello stato di ogni singolo Task (completo o ancora da svolgere). La procedura per l'assegnazione di un Task segue il seguente schema:

- inserire un titolo al task;
- indicare la/e persona/e a cui è stato assegnato tale compito;
- inserire la data di inizio e di fine di tale compito;
- inserire una descrizione che contiene un breve riassunto del compito assegnato e il ruolo assunto in quella fase del progetto;

- inviare una notifica via email alle persone a cui è stato assegnato il task.

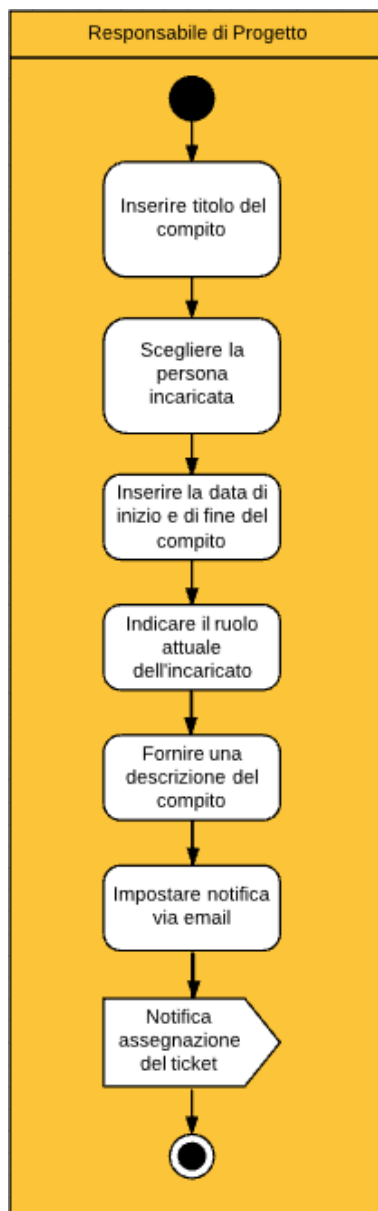


Figura 8: Assegnazione di un ticket

#### 4.1.5.4 Gestione degli strumenti di versionamento

##### 4.1.5.4.1 Repository

Per il versionamento e il salvataggio dei file è previsto l'utilizzo di diversi repository su *GitHub<sub>G</sub>*. L'*Amministratore* di Progetto si deve occupare della creazione dei repository, che rappresentano il gruppo del progetto. Successivamente l'*Amministratore* inserirà tutti i componenti del gruppo, i quali dovranno essere in possesso di un account personale, come collaboratori.

E' previsto l'utilizzo di due repository:

- **Documents:** contiene tutta la documentazione dell'attività di progetto;
- **SWEDesigner:** contiene tutti i file implementativi del progetto.

#### 4.1.5.4.2 Struttura del repository

I membri del gruppo sono tenuti a rispettare le seguenti norme sull'organizzazione dei file nel repository.

Struttura del repository Documents:

- **StyleLatex:** contiene i file del template e il layout dei documenti  $\text{\LaTeX}$ ;
- **Cartella documenti:** contiene due sottocartelle che dividono i documenti interni ed esterni al gruppo. All'interno è presente una suddivisione dei singoli documenti in cartelle dove in ciascuna di esse sono contenuti tutti i relativi file  $\text{\LaTeX}$  i file pdf e una cartella images che contiene le immagini presenti nei documenti;
- **Script:** contiene tutti gli script utilizzati.

La struttura del repository SWEDesigner sarà definita in modo dettagliato nelle successive revisioni.

#### 4.1.5.4.3 Tipi di file e .gitignore

Nelle cartelle contenenti tutti i documenti saranno presenti solamente i file .tex, .pdf, .jpg, .png. Le estensioni dei file generati automaticamente dalla compilazione sono stati aggiunti a .gitignore, e quindi vengono ignorati e resi invisibili a Git.

#### 4.1.5.4.4 Norme sui commit

Ogni volta che vengono effettuate delle modifiche ai file del repository, le quali poi vengono caricate su di esso, bisogna specificarne le motivazioni. Questo avviene utilizzando il comando `commit` accompagnato da un messaggio riassuntivo e una descrizione in cui va specificato:

- la lista dei file coinvolti;
- la lista delle modifiche effettuate, ordinate per ogni singolo file.

Prima di eseguire tale procedura, va aggiornato il diario delle modifiche, secondo le regole viste in 3.1.6.2.

#### **4.1.5.5 Gestione dei rischi**

Il *Responsabile* di Progetto ha il compito di rilevare i rischi indicati nel *Piano di Progetto v1.0.0*. Nel caso ne vengano individuati di nuovi dovrà aggiungerli nell'analisi dei rischi. La procedura da seguire per la gestione dei rischi è la seguente:

- individuare problemi non calcolati e monitorare i rischi già previsti;
- registrare ogni riscontro previsto dei rischi nel *Piano di Progetto v1.0.0*;
- aggiungere i nuovi rischi individuati nel *Piano di Progetto v1.0.0*;
- ridefinire, se necessario, le strategie di progetto.

#### **4.1.6 Strumenti**

##### **4.1.6.1 Sistema operativo**

Il gruppo di progetto lavora sui seguenti sistemi operativi:

- Ubuntu 16.04 *LTS<sub>G</sub>* x64
- Windows 10 Pro x64
- Windows 10 Home x64
- MAC OS Sierra 10.12.3 x64
- Manjaro 17.0 x64

##### **4.1.6.2 Slack**

Slack è uno strumento di collaborazione utile per le comunicazioni interne ad un gruppo di lavoro. L'applicazione appare come una comune sistema di messaggistica, con in più la possibilità di integrare numerosi servizi tra cui GitHub e Wrike. Gli utenti del team possono suddividere gli argomenti di discussione utilizzando degli *hashtag<sub>G</sub>* per organizzare al meglio la comunicazione. L'applicazione è gratuita ed è necessaria la registrazione di un dominio per il proprio team. Una volta registrati gli utenti possono essere invitati all'interno del gruppo.

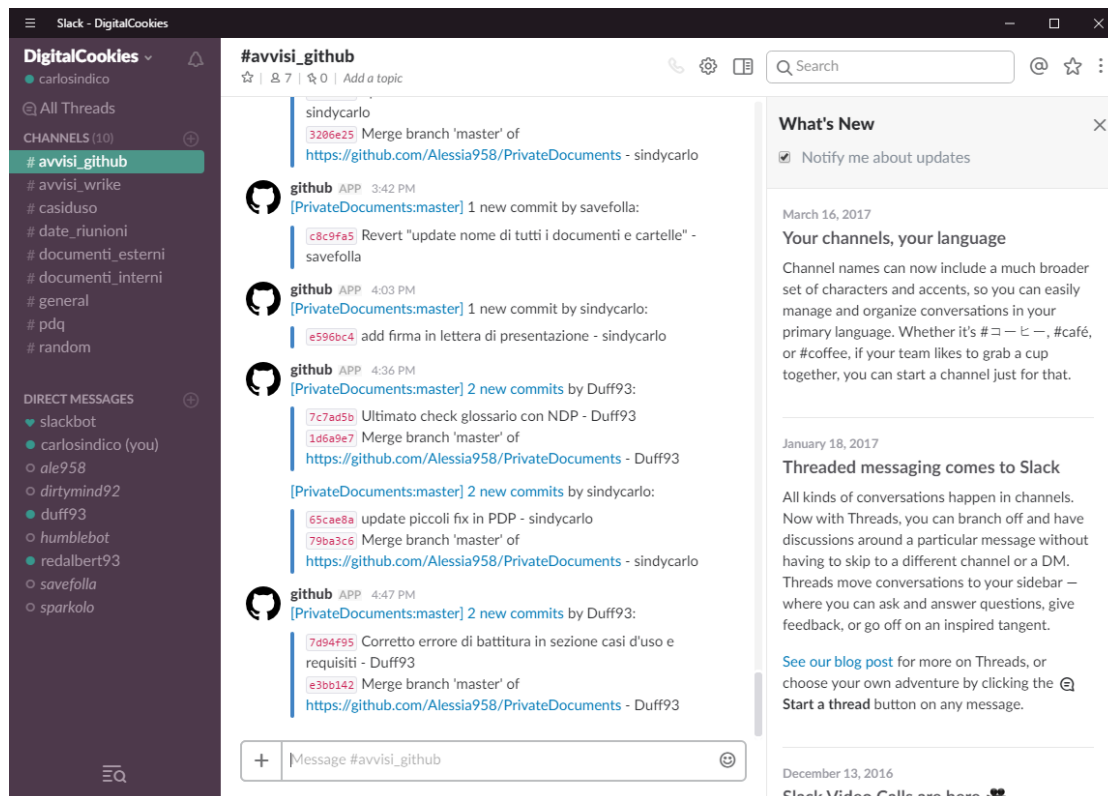


Figura 9: Slack Desktop per Windows

#### 4.1.6.3 Wrike

Wrike è uno strumento online per la collaborazione e il *project management*<sub>G</sub>. Permette ai suoi utenti di modificare progetti, classificare le attività per importanza, tenere traccia dei programmi e collaborare online con altri utenti dello stesso gruppo. Wrike ha diverse funzionalità, tra le più importanti ci sono:

- flusso dei movimenti in tempo reale;
- diagrammi di *Gantt*<sub>G</sub>;
- condivisione di file e modifiche online;
- gestione del carico di lavoro e rilevamento temporale;
- discussioni;
- classificazione delle attività per importanza.

Il programma Student di Wrike consente agli studenti universitari di sfruttare gratuitamente un abbonamento Wrike Professional per 15 utenti.

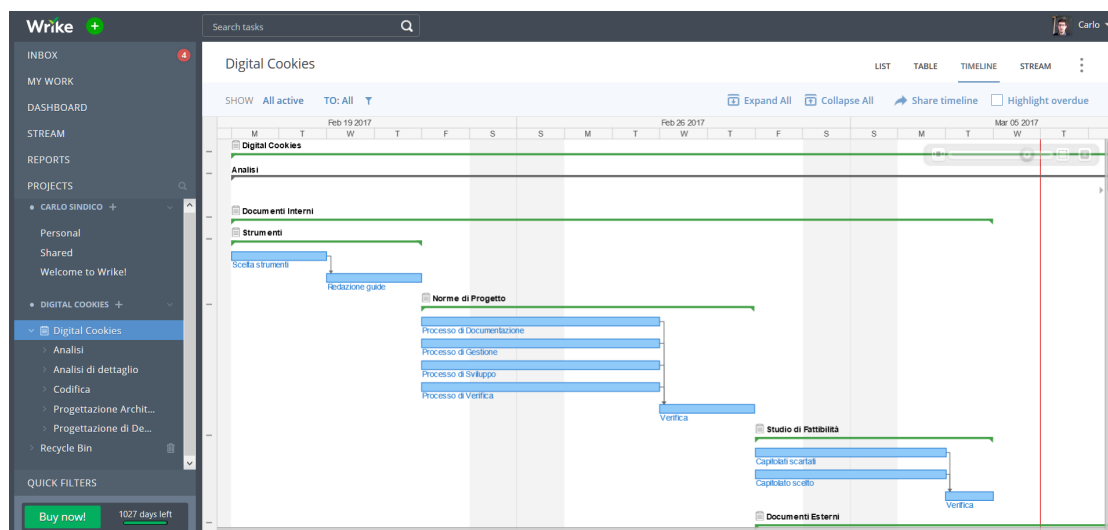


Figura 10: Wrike Web application per Windows

#### 4.1.6.4 Git

Git è un sistema software di controllo di versione distribuito, creato da Linus Torvalds nel 2005. La versione utilizzata è maggiore o uguale alla 2.7.4.

<https://git-scm.com/>

Inoltre è stata redatta una guida Git volta a formare in maniera uniforme i componenti del gruppo sull'utilizzo dello strumento. Per maggiori dettagli sul funzionamento dei comandi si rimanda a *Guida ai comandi Git v1.0.0*.

#### 4.1.6.5 GitHub

GitHub è un servizio web di *hosting<sub>G</sub>* per lo sviluppo di progetti software, che usa il sistema di controllo di versione Git. Può essere utilizzato anche per la condivisione e la modifica di file di testo e documenti revisionabili (sfruttando il sistema di versionamento dei file di Git). GitHub offre diversi piani per repository privati sia a pagamento, sia gratuiti, molto utilizzati per lo sviluppo di progetti OpenSource.