OBERON

Norme di Progetto v 2.1.0

A.A. 2021-2022

Componenti del gruppo:

Casazza Domenico, matr. 1201136

Casonato Matteo, matr. 1227270

Chen Xida, matr. 1217780

Pavin Nicola, matr. 1193215

Poloni Alessandro, matr. 1224444

Scudeler Letizia, matr. 1193546

Stojkovic Danilo, matr. 1222399

Indirizzo repository GitHub:

https://github.com/TeamOberon07/ShopChain

Università degli Studi di Padova

Indice

1 Registro delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
2.1.0	08/05/2022	Poloni Alessandro	Verificatore	Verifica sezioni Strumenti §(3.4), Analisi del flusso dati §(4.5.6), Verifica formale §(4.5.7), Metriche §(4.7) e Strumenti §(4.8)
2.0.1	06/05/2022	Casazza Domenico	Amministratore	Aggiunta sezioni Strumenti §(3.4), Analisi del flusso dati §(4.5.6), Verifica formale §(4.5.7), Metriche §(4.7) e Strumenti §(4.8)
2.0.0	07/03/2022	Poloni Alessandro	Responsabile	Approvazione del documento
1.1.0	07/03/2022	Casazza Domenico	Verificatore	Verifica del documento
1.0.8	06/03/2022	Poloni Alessandro	Responsabile	Aggiunto Convenzioni §(4.2), Verifica documentazione §(4.5.4), Verifica del codice §(4.5.5), Test §(4.6), aggiunte descrizioni ruoli
1.0.7	06/03/2022	Scudeler Letizia	Amministratore	Aggiunto scopo §(4.1.1), aspettative §(4.1.2), descrizione §(4.1.3), ciclo di vita del documento §(4.1.4) e gestione organizzativa §(5.1)

Versione	Data	Nominativo	Ruolo	Descrizione
1.0.6	05/03/2022	Casazza Domenico	Analista	Aggiunto fornitura §(3.1), aggiornato sviluppo §(3.2) e metriche §(4.4.2)
1.0.5	05/03/2022	Chen Xida	Progettista	Aggiunto template §(4.1.7), il suo uso §(4.1.8), i nomi file §(4.1.16) e le tabelle §(4.1.17)
1.0.4	04/03/2022	Stojkovic Danilo	Progettista	Riportato workflow codifica §(3.2.3), struttura codice §(3.3), rendiconto ore, infrastrutture interne §(5.2)
1.0.3	03/03/2022	Chen Xida	Progettista	Aggiunto verifica §(4.5) e validazione §(4.7)
1.0.2	03/03/2022	Casonato Matteo	Progettista	Aggiunto sviluppo §(3.2) e Gestione rischi §(5.7)
1.0.1	26/02/2022	Chen Xida, Stojkovic Danilo	Progettista, Progettista	Riportata scaletta Scrum §(5.4.3) e stesura struttura PdP §(4.1.12), struttura glossario §(4.1.9), PdQ §(4.1.10)
1.0.0	12/02/2022	Casonato Matteo	Responsabile	Approvazione del documento
0.1.1	10/12/2021	Stojkovic Danilo	Responsabile	Configurazione cruscotto §(5.5.5)
0.1.0	08/12/2021	Casazza Domenico	Verificatore	Verifica complessiva
0.0.3	07/12/2021	Stojkovic Danilo	Amministratore	Aggiunta struttura ai verbali §(3.3)

Versione	Data	Nominativo	Ruolo	Descrizione
0.0.2	30/11/2021	Stojkovic Danilo	Amministratore	Stesura introduzione, stile generale e strumenti §(1), §(4.1.5), §(5.5)
0.0.1	29/11/2021	Team Oberon	Analisti	Creazione bozza documento

2 Introduzione

2.1 Scopo del Documento

Questo documento verrà utilizzato dal *Team Oberon* come riferimento per il metodo di lavoro, in modo che sia uniforme e funzionale.
Pertanto, il documento si pone come obiettivo la descrizione di tutte le procedure e le regole che le governano, ogni strumento, ogni metrica di qualità, ogni tecnologia utilizzata. Tutti i membri del gruppo si impegnano a seguire le norme descritte di seguito in ogni istante del progetto.

2.2 Glossario

Per evitare possibili ambiguità che potrebbero sorgere durante la lettura dei documenti, tutti i termini di rilevante importanza e con significato particolare sono stati definiti nel documento Glossario 1.0.0.

2.3 Riferimenti normativi

- Regolamento del Progetto didattico:
  `https://www.math.unipd.it/~tullio/IS-1/2021/Dispense/PD2.pdf`
- Capitolato: `https://www.math.unipd.it/~tullio/IS-1/2021/Progetto/C2.pdf`

2.4 Riferimenti informativi

- Standard ISO/IEC 12207: `https://it.wikipedia.org/wiki/ISO_12207`
- Standard ISO/IEC 9126: `https://en.wikipedia.org/wiki/ISO/IEC_9126`

3 Processi primari

3.1 Fornitura

3.1.1 Rapporti con il Proponente

Il proponente offre al team diversi canali di comunicazione (e-mail, Discord) attraverso i quali è possibile organizzare una riunione con tempistiche molto brevi.
Non ci sono meeting regolari pianificati ma entrambe le parti si mantengono aggiornate con costanza.
Tra i motivi di discussione ci sono:

- dubbi sulle tecnologie utilizzate;
- dubbi su requisiti e vincoli del capitolato;
- feedback sul lavoro prodotto (software e documentazione);
- tempistiche progettuali;

Ognuno di questi meeting è poi riassunto nel verbale corrispondente da un membro del team e verificato da un altro.

3.1.2 Pianificazione

La pianificazione è suddiviso in sottosezioni corrispondenti ai scrum. In ogni scrum sono riportate le seguenti informazioni:

- Durata dello scrum
- Obiettivi da raggiungere nello scrum
- Attività da svolgere per l'avanzamento
- Grafico che illustra le durate previste per le attività

3.1.3 Bilancio

In questa sezione sono riportati i grafici che descrivono i preventivi e i consuntivi relativi ad ogni periodo di scrum.
Sono state evidenziate in verde le celle che rappresentano le ore produttive usate dai membri del team nei corrispondenti ruoli.
Vengono poi riportati le motivazioni per gli scostamenti verificati e i provvedimenti che verranno adottati attivamente al fine di migliorare l'accuratezza dei preventivi.
Inoltre è stato riportato anche il budget totale del progetto fino allo scrum n-esimo nella sezione di riepilogo.
Il budget totale preventivato è stato calcolato sommando i costi necessari per lo scrum n-esimo e il budget consuntivo totale dello scrum precedente.

3.2 Sviluppo

3.2.1 Scopo

Lo scopo di questo processo è definire nel dettaglio tutte le procedure da seguire durante lo sviluppo del software, definendo obiettivi, vincoli tecnologici e di design, e test/validazione.

3.2.2 Attività

Le attività che prevede il processo di sviluppo sono **Analisi dei Requisiti**, **Progettazione** e **Codifica**.

3.2.2.1 Analisi dei Requisiti Gli analisti del team si occuperanno di sviluppare questo documento (dopo lo studio del capitolato e confronto con il proponente) che dovrà descrivere nel dettaglio le funzionalità principali del prodotto. Nello specifico, il documento dovrà contenere:

- Tecnologie scelte - Intero "stack" con descrizione;
- Casi d'uso - Descritti graficamente (tramite diagrammi UML) e testualmente;
- Requisiti - Funzionali, Prestazionali, di Qualità;
- Vincoli di Progetto.

3.2.2.2 Progettazione Durante l'attività di **Progettazione**, il team dovrà definire una soluzione al problema del capitolato, partendo dallo studio svolto nell'attività **Analisi dei Requisiti** e quindi da tutto ciò che è stato definito nell'omonimo documento.
Le due parti che compongono questa attività sono:

- **Requirements & Technology Baseline** (RTB):
  - motivazione delle tecnologie scelte per lo sviluppo software;
  - stesura di Piano di Progetto, Piano di Qualifica, Norme di Progetto, Verbali;
  - sviluppo Proof of Concept, un prototipo che dimostri le funzionalità del prodotto;
- **Product Baseline**
  - definizione delle linee guida architetturali del prodotto, basate sulla RTB;
  - stesura di Manuale utente e Verbali.

3.2.2.3 Codifica Durante questa attività si andrà a tradurre in codice tutta la parte che riguarda l'analisi svolta durante le attività precedenti, concretizzando la soluzione software. Per quanto riguarda lo stile del codice, si adotteranno le seguenti convenzioni:

- Per i nomi delle variabili va utilizzato camelCase (iniziale minuscola).
- Il codice va indentato correttamente per favorire la leggibilità.
- Le funzionalità vanno separate in file semplici e con scopi unici.
- Limitare le dipendenze tra elementi del codice.

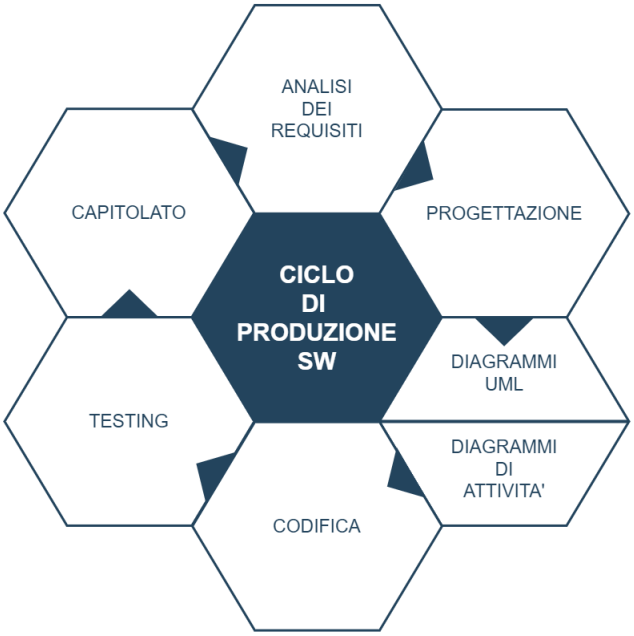
3.2.3 WorkFlow Codifica

Il team ha creato due repository separate da quella principale nelle quali lavorare rispettivamente al lato mobile e web dell'applicativo, ciò permette meno complicazioni di merge e dei commit più "puliti" sulla repository principale (vengono effettuati solo quando sono presenti grandi passi avanti).

Generalmente si è deciso che in ogni periodo o scrum il team verrà suddiviso in due gruppi

ai quali sono assegnate le due repository di sviluppo, questo permette ad ogni membro di concentrarsi su singoli aspetti del progetto e facilita la collaborazione essendo i sottogruppi meno numerosi.

L'obiettivo principale dei momenti di codifica deve essere la trasformazione in codice eseguibile della progettazione dedotta dai requisiti. Le idee ed il funzionamento devono essere già ben definiti prima di dedicarsi alla scrittura e compilazione. Lavorare a livello più teorico prima di codificare aiuta la produzione di codice più comprensibile e ben pensato perché appunto pianificato totalmente in anticipo.



Come si può notare dall'immagine precedente la codifica rappresenta uno degli ultimi passaggi del ciclo di produzione, questo perché essa va limitata alla semplice scrittura in codice di idee ben chiare sul funzionamento dell'applicativo.

3.3 Struttura Codice

3.3.1 Best Practice

- Per i nomi delle variabili va utilizzato camelCase (iniziale minuscola).
- Il codice va indentato correttamente per favorire la leggibilità.
- Le funzionalità vanno separate in file semplici e con scopi unici.
- Limitare le dipendenze tra elementi del codice.

3.4 Strumenti

Gli strumenti utilizzati in questo processo comprendono:

- **React**: libreria JavaScript per facilitare la creazione di interfacce utente, aumentare la manutenibilità e la facilità di testing;

  https://it.reactjs.org/

- **Ethers.js**: libreria JavaScript per interagire con le blockchain basate su Ethereum;

  https://docs.ethers.io/v5/

- **Npm**: packet manager per il linguaggio JavaScript;

  https://www.npmjs.com/

- **Flutter**: framework open-source per la creazione di interfacce native per iOS e Android;

  https://flutter.dev/development

- **Dart**: linguaggio di programmazione su cui è basato Flutter;

  https://dart.dev/

- **dart_web3**: Libreria dart che permette di collegarsi ed interagire con blockchain basate su Ethereum;

  https://pub.dev/packages/dart_web3/

- **wallet_connect**: Libreria dart che permette di collegarsi ad un wallet;

  https://pub.dev/packages/wallet_connect

- **qr_code_scanner**: Libreria dart che permette di scannerizzare QR code;

  https://pub.dev/packages/qr_code_scanner

- **MetaMask**: gestore di wallet;

  https://metamask.io/

- **Visual Studio Code**: IDE versatile ed estendibile scelto per lo sviluppo del codice.

  https://code.visualstudio.com/

4 Processi di supporto

4.1 Documentazione

4.1.1 Scopo

Tutti i processi e le attività di sviluppo necessitano di essere documentate. In questa sezione vengono riportate le convenzioni e le norme da applicare in fase di stesura e tutti gli strumenti di cui i componenti del Team Oberon hanno bisogno.

4.1.2 Aspettative

- Definire una struttura chiara che renda i documenti uniformi tra loro;
- Avere delle norme da seguire per ogni aspetto della stesura del documento, in modo da rendere capace di lavorare in modo autonomo ogni componente del gruppo;

4.1.3 Descrizione

La documentazione è essenziale per tenere traccia di tutte le informazioni relative alle attività svolte e alle decisioni prese durante lo svolgimento del progetto. Grazie ad essa, tutti gli stakeholder possono avere un resoconto del lavoro.

4.1.4 Ciclo di vita del documento

1. Pianificazione: durante una prima fase di brain storming i componenti del gruppo raccolgono tutte le idee per la creazione del documento e vengono definite le parti che lo compongono;
2. Impostazione: viene prodotta una prima bozza del documento e il suo scheletro;
3. Realizzazione: il contenuto del documento viene redatto;
4. Verifica: il documento è sottoposto ad un'attenta revisione per individuare e correggere eventuali errori;
5. Approvazione: l'approvatore dà il suo consenso al documento, perciò è completo e pronto per il rilascio;

4.1.5 Stile Generale

Il materiale contenuto nei vari documenti sarà preferibilmente riassuntivo, non eccessivamente formale e di semplice comprensione e fruizione, saranno valorizzati grafici e schemi per facilitare e velocizzare l'acquisizione delle informazioni.
Per la creazione dei documenti di progetto verrà utilizzato il markup language LaTeX basando ogni istanza su un template predeterminato.

4.1.6 Intestazione

1. Ogni documento redatto dal gruppo avrà una pagina introduttiva contenente il logo, seguito dal titolo, dalla versione e dai componenti del gruppo. Infine conterrà il link alla repository GitHub.

2. La pagina successiva sarà interamente dedicata al registro delle modifiche degne di nota in un'apposita tabella, con le seguenti colonne: versione, data, nominativo, ruolo, descrizione dell'aggiornamento.
3. Infine, la terza e ultima pagina dell'intestazione conterrà il sommario LaTeX che si aggiornerà all'aggiunta di ogni sezione.

4.1.7 Template

Il team Oberon ha deciso di adottare un template in modo tale da aver ben definito la struttura base dei documenti.
Questo facilità di gran lunga la creazione e il mantenimento di nuovi documenti, infatti l'aspetto e la struttura base viene definito a priori.
Questo include il frontespizio, il registro delle modifiche e l'indice.

Informazioni riportate in:

Frontespizio

- nome del documento;
- anno accademico;
- dati relativi ai componenti del gruppo;
- indirizzo della repository di GitHub.

Registro delle modifiche

- Versione: il formato della versione è del tipo x.y.z dove x è il numero della versione approvata per la revisione;
- Data: il formato è DD/MM/YYYY;
- Nominativo: colui che ha modificato il documento;
- Ruolo: descrive il ruolo principale assunto nello scrum attuale;
- Descrizione: breve descrizione sulle modifiche effettuate.

Indice norme di progetto
L'indice è autogenerato ed è suddiviso il tre sezioni principali:

- processo primario;
- processo di supporto;
- processo organizzativo;

Ad ogni sezione è associata al tutte le norme correlate da seguire per mantenere un approccio sistematico, disciplinato e quantificabile allo sviluppo.

4.1.8 Uso del template

Per le modifiche dei progetti è fortemente consigliato seguire la seguente procedura:

1. Scaricare la cartella Template Latex in formato .zip;

2. Importare la cartella compressa su Overleaf (Nuovo Progetto –> Carica Progetto –> Upload .zip file);
3. Andare su Menu (in alto a sinistra) –> Documento Principale e selezionare main.tex;
4. Modificare il titolo del documento in main.tex;
5. Creare un file sezioneX.tex per ogni sezione e importarlo in main.tex;
6. Aggiornare import.tex su GitHub se modificato;
7. Una volta finito il documento, la cartella da caricare su GitHub dovrà contenere:
    - main.tex del documento;
    - files sezioneX.tex del documento;
    - pdf esportato del documento.

4.1.9 Struttura Glossario

Il glossario è un semplice documento che raggruppa i termini utilizzati nella stesura degli altri documenti affiancandoli alle loro definizioni, questo facilita la comprensioni di chiunque degli argomenti trattati.
La sua struttura è basilare, si limita ad ordinare alfabeticamente una serie di piccole descrizioni delle varie parole chiave del progetto.

4.1.10 Struttura Piano di Qualifica

Il piano di qualifica descrive tutti i mezzi adottati per misurare e monitorare sia la qualità dei processi (primari, secondari e organizzativi) che quella del software.
La qualità del prodotto è descritto attraverso metriche che hanno come obiettivo garantire certe qualità del software rispettandole.
Le metriche e gli obiettivi sono descritti in formato tabellare.
I valori delle metriche variano nel tempo e il loro andamento sono descritti come grafici.
Inoltre nella parte conclusiva del documento sono riportati tutti i test necessari da superare per raggiungere la maturità del prodotto.

4.1.11 Struttura Rendiconto delle ore

Sia il preventivo che il consuntivo delle ore rendicontate di lavoro vengono registrati sui rispettivi file Excel su Teams.

Preventivo

Il preventivo è suddiviso in vari fogli (2 per scrum):

- il primo contiene le ore che il team prevede di dedicare ad ogni ruolo per membro;
- il secondo contiene lo stato delle ore totali al momento della compilazione (la somma del consuntivo precedente e del preventivo attuale).

Consuntivo

Il consuntivo è suddiviso in vari fogli (2 per scrum):

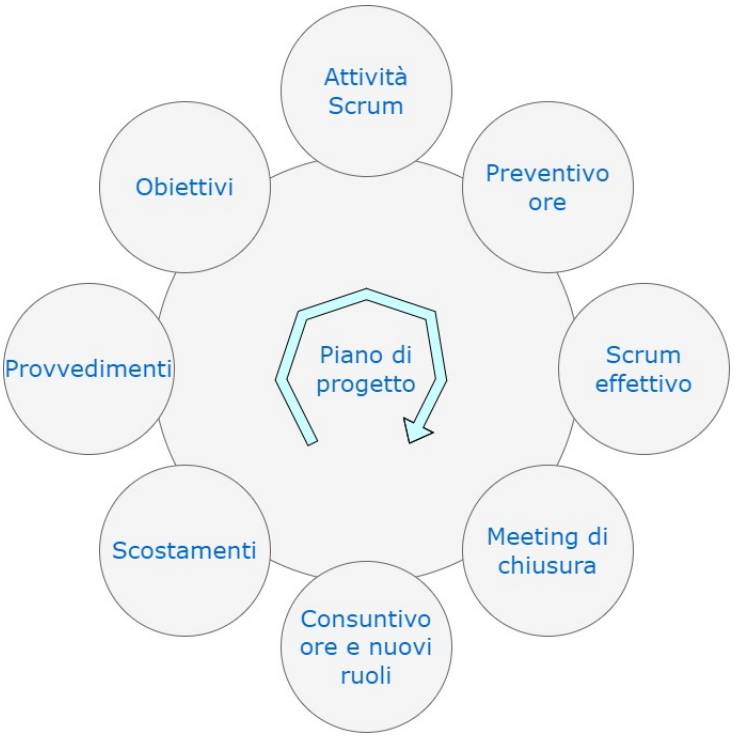
- il primo contiene le ore che il team ha effettivamente dedicato ad ogni ruolo per membro;
- il secondo contiene lo stato delle ore totali al momento della compilazione (la somma totale dei consuntivi finora). Questo permette di tenere bene traccia dell'andamento del progetto.

Le tabelle prodotte ad ogni scrum vengono poi incorporate nel piano di progetto.

4.1.12 Struttura Piano di Progetto

Il piano di progetto è suddiviso in tre sezioni principali:

- Analisi dei rischi
- Pianificazione
- Bilancio



4.1.13 Analisi dei rischi

L'analisi dei rischi serve a prevenire e mitigare futuri imprevisti in modo tale da reagire con decisioni ragionevoli in tempi brevi.
Il grado di rischio considera sia la frequenza che la gravità dell'evento analizzato.

I rischi possono essere di natura umana e di carattere tecnologico.
Il piano di contingenza ordina i rischi in base al grado e descrive in modo sintetico le procedure da seguire che portano a risultati concreti.
Ogni qualvolta che si verifica un evento, se è presente nella lista degli eventi considerati si seguono le procedure e viene riportato il riassunto di ciò che si fatto nell'attualizzazione, altrimenti si aggiorna la tabella dei rischi e si ragiona un piano di contingenza adatto per futuri episodi.

4.1.14 Documentazione Scrum

1. Product Backlog, il documento che contiene la lista di tutti requisiti necessari per la realizzazione del progetto, la sua prima stesura definisce i requisiti inizialmente conosciuti, per poi evolversi in base a come evolve il prodotto, ai feedback ricevuti dal cliente e alle necessità che emergono;

2. Sprint Backlog, è l'insieme degli elementi del Product Backlog selezionati per lo sprint, è una previsione fatta dal Team in relazione alle priorità indicate e al lavoro necessario per raggiungere gli obiettivi dello sprint;

3. Incremento, è la somma di tutti gli elementi del Product Backlog completati durante uno sprint.

4.1.15 Struttura verbali

- Dati introduttivi: titolo verbale numerato ordinalmente a partire dal primo di quella categoria (es. incontri con il proponente, scrum review...), luogo, data, durata, partecipanti (per il gruppo è sufficiente indicare "membri del team Oberon");

- Sezione Informazioni Generali: breve resoconto degli argomenti trattati nell'incontro descritto dal verbale, utile a chi è alla ricerca di un'informazione in particolare e consulta in velocità questa sezione;

- Sezione Resoconto: nucleo del documento, qui verranno indicati tutti gli argomenti discussi in dettaglio e le conclusioni alle quali si è giunti.

4.1.16 Nomi file

Verbali: il nome di ogni verbale avrà la seguente struttura "*TipologiaVerbale_YYYY_MM_DD*" dove *TipologiaVerbale* potrà essere: *VerbaleInterno* per i verbali di incontri interni tra i membri del Team Oberon, *VerbaleEsterno* per i verbali di incontri con il proponente di C2, *Verbale_N_Scrum* per i verbali di conclusione dell'N-esimo Scrum.
Nomi file che necessitano manutenzione e versionamento:
NomeFile_x.y.z, dove x.y.z indica la versione.
Questa distinzione è stata fatta perché a differenza dei verbali, il PdQ, PdP, AnR e il glossario necessitano di aggiornamenti costanti ed è indispensabile quindi informazioni sul versionamento.

4.1.17 Tabelle

Le tabelle sono autogenerate dal sito: `https://www.tablesgenerator.com`.
Il sito offre un servizio di tipo SaaS per le tabelle LateX, usando un'interfaccia intuitiva WYSIWYG (What You See Is What You Get) riducendo i tempi di stesura.
Sono stati scelti colori gradevoli alla vista che richiamano gli stessi del logo del team Oberon (rosso chiaro, celeste chiaro e blu chiaro).
I dati sono su sfondi di colori alterni per facilitarne la distinzione nelle varie righe (soprattutto in tabelle lunghe).
Nell'immagine sottostante è riportato una tabella tipo:

Colonna1	Colonna2	Colonna3
Dato1	Dato2	Dato3
Dato4	Dato5	Dato6
Dato7	Dato8	Dato9
Dato10	Dato11	Dato12

4.2 Convenzioni

4.2.1 Data

I componenti del gruppo utilizzano il seguente formato per la rappresentazione delle date:

YYYY-MM-DD

dove **YYYY** specifica l'anno, **MM** il mese e **DD** il giorno.

4.3 Gestione della Configurazione

4.3.1 Scopo

Lo Scopo di questa sezione è definire come il Team Oberon attua il processo di gestione della configurazione, ovvero come il team di sviluppo ha deciso di mantenere tracciata la documentazione redatta ed il codice sviluppato.

4.3.2 Aspettative

I risultati attesi da questo processo sono:

- tracciamento delle modifiche;
- standardizzare la produzione di codice e documentazione;
- uniformare l'utilizzo degli strumenti di versionamento coinvolti;
- classificare i prodotti dei vari processi implementati;
- individuare e risolvere possibili conflitti ed errori;
- condividere tra i membri del gruppo il materiale configurato.

4.3.3 Descrizione

Il processo di gestione della configurazione viene attuato con l'obbiettivo di rendere la documentazione redatta e il codice sviluppato organizzati, tracciabili e disponibili, avendo a disposizione la storia di ogni file prodotto. Questo è possibile grazie all'utilizzo di un repository, in cui vengono gestiti e strutturati i file prodotti.

4.3.4 Versionamento

4.3.4.1 Codice di Versione La versione di ogni documento redatto viene identificata tramite un codice numerico di tre cifre:

[X].[Y].[Z]

dove:

- **X** indica una **versione stabile**, ovvero approvata dal responsabile di progetto;
- **Y** indica una **versione controllata**, ovvero revisionata da un verificatore;
- **Z** indica una **versione modificata**, ovvero modificata da un redattore.

Tutte le cifre iniziano dal valore 0, e vengono modificate come seguentemente descritto:

- dopo lo svolgimento di una modifica: **X** viene incrementato;
- dopo lo svolgimento di una revisione: **Y** viene incrementato, **X** viene azzerato;
- dopo lo svolgimento di una approvazione: **Z** viene incrementato, **X** e **Y** vengono azzerati.

4.3.4.2 Sistemi Software Il team ha deciso di utilizzare Git come sistema di versionamento, ed in particolare i servizi offerti da Github per la gestione del repository, in quanto:

- Tutti i membri hanno già familiarità con il software;
- Possibilità di utilizzo tramite browser, applicazione desktop, applicazione mobile o linea di comando;
- Integrazione di un issue tracking system;

È stata creata un'organizzazione denominata *TeamOberon07* in cui vengono gestiti i repository che vengono utilizzati, accessibili in scrittura e lettura a tutti i membri. Per il versionamento della documentazione viene usata una repository dal nome *ShopChain*, in cui vengono riposti tutti i documenti creati durante lo svolgimento del progetto didattico.

Inoltre il team utilizza i servizi offerti da Microsoft Teams come repository interno ed intermedio, dove vengono caricati ed aggiornati i vari documenti prima di essere regolarmente caricati su Github a fine di ogni ciclo scrum.

4.3.5 Struttura repository

4.3.5.1 Repositori Utilizzati Tutti i documenti redatti sono presenti nel repository pubblico *TeamOberon07/ShopChain*. I file sorgenti sono disponibili ai componenti del gruppo presso il repository interno presente sulla piattaforma Microsoft Teams, da essa vengono caricati periodicamente su Github in formato PDF a fine di ogni ciclo scrum.
I documenti nel repository Github sono organizzati in due cartelle separate, in base al loro utilizzo, interno o esterno:

- **Documentazione Esterna**, contenente:
  - *Analisi dei Requisiti*;
  - *Glossario*;
  - *Piano di Progetto*;
  - *Piano di Qualifica*.
- **Documentazione Interna**, contenente:
  - *Norme di Progetto*;
  - *Verbali*, cartella contenente sottocartelle in cui vengono posti tutti i verbali prodotti:
    - *Verbali Interni*;
    - *Verbali Esterni*;
    - *Verbali Fine Scrum*.

4.3.5.2 Gestione dei Cambiamenti Come già precedentemente accennato, i documenti in fase di redazione si trovano sulla piattaforma Microsoft Teams. Successivamente quando gli autori completano il loro lavoro, lo sottopongono all'attenzione del verificatore che si occupa del controllo e si assicura che rispetti il way of working del Team Oberon. Infine il responsabile si occupa del caricamento del documento sul repository pubblico su GitHub.
In questo modo il repository viene mantenuto pulito ed entro gli standard di qualità, in quanto i documenti vengono resi disponibili soltanto dopo revisione o accettazione.

4.4 Qualità

4.4.1 Scopo

Lo scopo di questa sezione è definire gli obiettivi che il team si è prefissato nel processo di supporto per la gestione della qualità.

4.4.2 Metriche

Lo scopo della seguente sezione è descrivere le metriche adottate dal *Team Oberon* per misurare la qualità del proprio prodotto.

M1 Correttezza grammaticale

La correttezza grammaticale deve essere garantita dai controlli del Verificatore ad ogni ispezione.

M2 Indice di Gulpease

Indice che riporta il grado di leggibilità di un testo redatto in lingua italiana. La formula adottata è la seguente:

$$GUL = 89 + \frac{300*(totfrasi)-10*(totlettere)}{(totparole)}$$

M3 Numero di SLOC

Source lines of code (SLOC) è una metrica software che misura le dimensioni di un software basandosi sul numero di linee di codice sorgente.

M4 Densità dei commenti

Questa metrica misura la densità dei commenti all'interno del codice sorgente prodotto dal team di sviluppo.

M5 Comprensibilità del codice

Questa metrica misura la comprensibilità del codice ad una persona che non ha scritto quel codice, fa riferimento ai nomi autoesplicativi delle variabili, delle classi e dei metodi.

M6 Complessità ciclomatica

Questa metrica è utilizzata per misurare la complessità di un programma. Misura direttamente il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso.

M7 Copertura dei requisiti

Indice che misura in ogni istante la percentuale di requisiti obbligatori soddisfatti. La formula adottata è la seguente:

$$CRO = \frac{ROC}{RO} * 100$$

dove:

- **ROC** indica il numero di requisiti obbligatori coperti dall'implementazione;
- **RO** indica il numero totale dei requisiti obbligatori.

M8 Varianza rispetto al preventivo

Questa metrica misura la varianza del prezzo del consuntivo finale rispetto al preventivo iniziale.

M9 Errori per linee di codice

Questa metrica indica la correttezza del codice prodotto dal team di sviluppo ed è data dal numero di errori diviso il numero di righe di codice totale.

M10 Numero di click

Questa metrica misura il numero di click (o tocchi) necessari per attivare una determinata funzionalità dell'applicativo prodotto dal *Team Oberon*.

M11 Tempo di accesso ad una funzionalità

Questa metrica misura il tempo necessario per accedere ad una determinata funzionalità dell'applicativo prodotto dal *Team Oberon*.

M12 Uniformità al cambio del browser

Questa metrica indica l'uniformità della webApp al variare del browser e della versione su cui viene eseguita.

M13 Uniformità al cambio del sistema operativo

Questa metrica indica l'uniformità della mobile App al variare del sistema operativo e della versione su cui viene eseguita.

M14 Test Coverage

Questa metrica indica la percentuale di copertura dei test di unità di un determinato file di codice sorgente.

4.5 Verifica

4.5.1 Scopo

Lo scopo di questa sezione è definire le metodologie che il gruppo ha deciso di adottare per il processo di verifica.
La verifica è essenziale affinché non siano stati introdotti errori durante lo sviluppo, ed è necessario che sia ripetuta su tutti i processi in esecuzione e sulla documentazione.

4.5.2 Aspettative

- Verificare ogni fase, rispettando precisi e validi criteri;
- Verificare in modo corretto per ottenere successo in fase di validazione;
- Rispettare gli obiettivi di copertura individuati nel *Piano di Qualifica*;
- Automatizzare, se possibile, le attività del processo.

4.5.3 Descrizione

All'interno di tale processo è possibile distinguere due tipi di attività principali:

- **Analisi statica**: può essere svolta sia sulla documentazione che sul codice sorgente, poiché non ne richiede l'esecuzione. Valuta la conformità alle convenzioni del gruppo e agli standard stabiliti;
- **Analisi dinamica**: può essere svolta solamente sul codice sorgente, poiché è richiesta l'esecuzione dell'oggeto da verificare. Assicura che ogni unità del codice, individualmente e nel complesso, funzioni in modo atteso, tramite una serie di test.

4.5.4 Verifica documentazione

La verifica della documentazione consiste in un insieme di attività di analisi statica. Questa verifica può essere svolta manualmente oppure tramite strumenti automatici. Nel primo caso sono possibili due diverse modalità di verifica:

- **Walkthrough**: necessaria al primo controllo, sia per errori grammaticali e ortografici, sia per rilevare errori semantici, il verificatore esegue un controllo di tutto il documento.
- **Inspection**: revisione complessiva in tempi rapidi, soprattutto questo è possibile perché ogni parte della documentazione è stata già verificata almeno una volta e quindi si tende a verificare soprattutto le ultime modifiche.

4.5.5 Verifica del codice

La verifica del codice prevede attività sia di analisi statica che di analisi dinamica:

- **Analisi statica**: programmatori e verificatori controllano che il codice in esame sia stato prodotto rispettato i principi di buona programmazione individuati dal gruppo;
- **Analisi dinamica**: programmatori e verificatori eseguono il codice alla ricerca di eventuali errori e bug.

4.5.6 Analisi del flusso dati

Il gruppo effettua periodicamente un'analisi statica del codice prodotto per accertarsi che questo non acceda a variabili inesistenti o prive di valore.

4.5.7 Verifica formale

Il gruppo effettua un'analisi statica del codice per esplorare tutti i rami possibili di esecuzione senza dover eseguire il codice.

4.6 Test

L'attività di test consente al programmatore di individuare errori o bug a run-time. I test devono essere quanto più possibile automatizzati, poiché la loro esecuzione deve essere ripetibile.

4.6.1 Classificazione dei test

Ogni test è identificato rispettando la seguente notazione:

T[Tipologia test][Importanza requisito][ID]

Ogni campo ha il significato di seguito descritto:

- **Tipologia test**: identifica il tipo di test, può assumere i seguenti valori:
  - **U**: unità;
  - **I**: integrazione;
  - **S**: sistema;
  - **A**: accettazione;
  - **R**: regressione.
- **Importanza requisito**: se il campo è assente, il test viene utilizzato per verificare un requisito obbligatorio, altrimenti può assumere i valori **D** (requisito desiderevole) oppure **O** (requisito opzionale).
- **ID**: codice numerico crescente a partire da 1, consente di distinguere test dello stesso tipo.

4.7 Metriche

Metriche adottate nella fase si testing del codice:

- **Code Coverage (CC)**: Descrive quanto il codice prodotto è coperto da test dinamici;

$$\frac{LCE}{LC} * 100$$

- **Passed Test Cases Percentage (PTCP)**: Corrisponde alla percentuale di test passati rispetto al totale dei test dinamici;

$$\frac{TP}{TT}$$

- **Failed Test Cases Percentage (FTCP)**: Corrisponde alla percentuale di test falliti rispetto al totale dei test dinamici;

$$\frac{TF}{TT}$$

Legenda:

- **Lines of Code Executed (LCE)**: linee di codice sorgente eseguite dai test;
- **Lines of Code (LC)**: linee di codice sorgente totali;
- **Test Passati (TP)**: numero di test dinamici passati dal codice sorgente;
- **Test Falliti (TF)**: numero di test dinamici falliti dal codice sorgente;
- **Test Totali (TT)**: numero totale di test dinamici.

4.8 Strumenti

In questo processo sono stati utilizzati i seguenti strumenti:

- **Texmaker**: mette a disposizione un correttore automatico che individua le parole grammaticalmente scorrette sottolineandole in rosso;
- **HardHat**: ambiente di sviluppo Ethereum in cui è possibile testare smart contract;

  https://hardhat.org/

- **Chai**: libreria JavaScript per il testing di smart contract;

  https://www.chaijs.com/

- **React Testing Library**: libreria JavaScript per testare e renderizzare componenti React;

  https://testing-library.com/

- **flutter_test**: libreria Dart per il testing in Flutter;

  https://api.flutter.dev/flutter/flutter_test/flutter_test-library.html

- **Avalanche Fuji Testnet**: testnet utilizzata per testare l'applicativo con la prima versione dello smart contract;

  https://docs.avax.network/quickstart/fuji-workflow/

- **Rinkeby Testnet**: testnet utilizzata per testare l'applicativo con la seconda versione dello smart contract (dopo l'aggiunta della funzionalità della conversione in stable coin).

  https://rinkeby.etherscan.io/apidoc

4.9 Validazione

4.9.1 Scopo

Lo scopo di questa sezione è descrivere come avverrà il processo di validazione.
Verranno descritte le attività di controllo che serviranno per garantire che il prodotto sia conforme ai requisiti accordati con il proponente.

4.9.2 Processo di validazione

1. Superamento dei test unità (propedeutico)
2. Superamento d'integrazione (propedeutico)
3. Superamento di sistema (propedeutico)
4. Collaudo supervisionato

5 Processi organizzativi

5.1 Gestione organizzativa

5.1.1 Scopo

In questa sezione sono indicate le modalità di coordinamento del Team Oberon:

- Adottare un modello organizzativo che consente di individuare anticipatamente i rischi;
- Specificare un modello di sviluppo da seguire;
- Rispettare le scadenze e le spese concordate;

5.1.2 Aspettative

- Assicurarsi una pianificazione ragionevole dei compiti da svolgere;
- Comunicazione chiara e priva di ambiguità tra i componenti del gruppo;
- Rientrare nei costi previsti regolando le attività;
- Distribuzione corretta ed equa dei ruoli di progetto;

5.1.3 Descrizione

- Assegnazione dei ruoli e dei compiti;
- Inizio e definizione dello scopo;
- Istanziazione dei processi;
- Stima e pianificazione di costi, tempi e risorse;
- Esecuzione e controllo;
- Revisione e valutazione delle attività in modo periodico;

5.1.4 Rapporti interpersonali tra i componenti del gruppo

I componenti nel gruppo si impegnano a collaborare in modo pacifico e ben coordinato per facilitare la buona riuscita del progetto.
In particolare si cercherà il più possibile di lavorare in meeting a distanza per evitare errori o sviste e per motivare i vari membri.
I ruoli verranno assegnati ai componenti del gruppo in modo equo, affinché ogni persona possa svolgere tutti gli incarichi almeno una volta e per un numero di ore produttive adeguato. I ruoli di progetto vengono assegnati in concomitanza con l'inizio di un nuovo ciclo scrum per garantire coesione nel lavoro.
Tutte le comunicazioni riguardanti il progetto avverranno utilizzando il gruppo Telegram dedicato, queste possono includere:

- pubblicazione/aggiornamento di un documento;
- richiesta di verifica di un documento;

- invito a collaborare su una particolare sezione del lavoro;
- confronto sulle disponibilità con i tempi per incontri con il proponente o i committenti;
- richiesta di feedback immediato su un particolare dubbio.

5.2 Utilizzo delle infrastrutture interne

5.2.1 Gestione delle comunicazioni

Il team considera la comunicazione regolare uno strumento vitale per il progetto, che sia essa tra i componenti, con il proponente o con il committente.
Qualsiasi novità, progresso o dubbio va innanzitutto esposto ai compagni di gruppo che cercheranno di mantenersi spesso disponibili. Inoltre è fondamentale discorrere spesso con il proponente per ogni dettaglio o problema da chiarire.
Le tecnologie adoperate a questi fini sono spiegate nelle sezioni §5.5.2 e §5.5.3.

5.2.2 Gestione degli incontri

Ogni incontro è generalmente preceduto da un incontro privato dei membri del gruppo, per chiarirne gli scopi e gli argomenti da affrontare, per spartire le parti del discorso, per preparare eventuali presentazioni. Questo permette un flusso del discorso più fluido e ottimizza il tempo con il proponente/committente.

5.3 Formazione

5.4 Metodo di Sviluppo

5.4.1 Scrum

Il Way of Working del team Oberon si basa sul framework Scrum, che prevede il raggiungimento dell'obiettivo finale del progetto tramite una serie di sprint, ognuno dei quali fissa un obiettivo intermedio da raggiungere e dura due settimane.

5.4.2 Eventi Sprint

1. Sprint Planning, è la riunione che segna l'inizio dello sprint: il Team concorda l'obiettivo principale dello sprint e individua gli elementi del Product Backlog necessari al suo raggiungimento;
2. Sprint Review, viene organizzata al termine di ogni sprint per valutare se l'obiettivo prefissato è stato raggiunto;
3. Sprint Retrospective, è una riunione che serve al Team per identificare le cause di eventuali problemi che si sono verificati durante l'ultimo sprint e individuare dei miglioramenti da apportare al processo.

5.4.3 Scaletta Scrum

Durante ogni chiamata di fine scrum il team seguirà i seguenti passaggi:

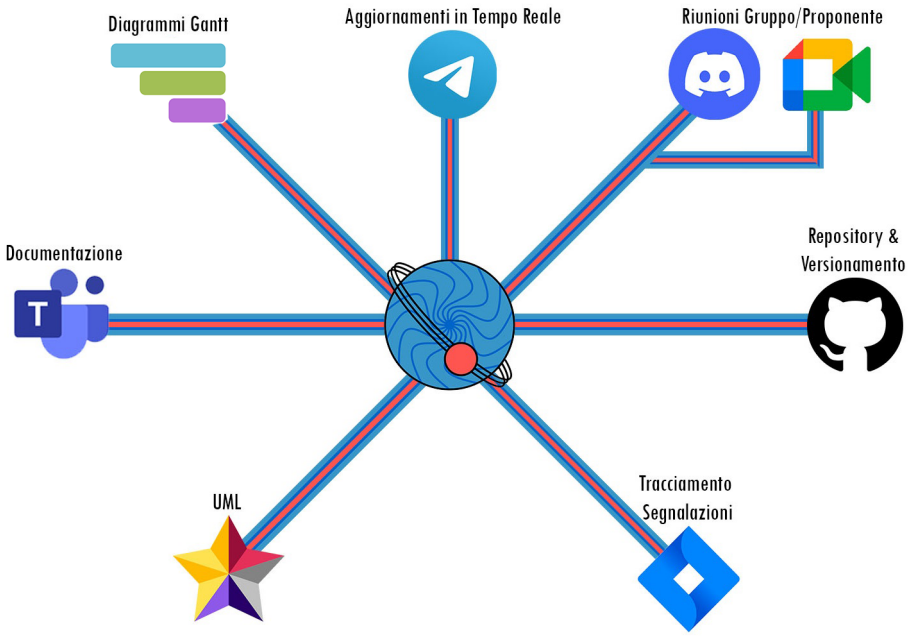
1. Aggiornamento dati sulla piattaforma Jira;

2. Confronto tra obiettivi prefissati e raggiunti;
3. Discussione delle problematiche riscontrate dai membri;
4. Caricamento sulla repository dei file aggiornati;
5. Definizione degli obiettivi dello scrum successivo;
6. Assegnazione dei ruoli di ogni membro.

5.5 Tecnologie e Software

5.5.1 Elenco

Il Team Oberon lavorerà in modalità asincrona utilizzando i seguenti software: Telegram, Discord e Google Meet, GitHub, Jira, StarUML, Microsoft Teams, Microsoft Project.



Le tecnologie sono state scelte in accordo tra i singoli membri del gruppo, che si sono basati sulle proprie preferenze e conoscenze. In caso di problematiche nell'utilizzo di uno di questi software, il gruppo si riunirà per sostituirla.

5.5.2 Telegram

La nota app di messagistica contiene un gruppo di cui ogni membro può usufruire per comunicare istantaneamente con tutti gli altri. Lo scopo principale consiste nel porre domande, chiarire dubbi e organizzare le riunioni.

5.5.3 Discord e Google Meet

La prima di queste due piattaforme è stata consigliata dal proponente, il quale ha suggerito la creazione di un canale dedicato al gruppo, con il fine di facilitare la comunicazione con l'azienda stessa ed avere a disposizione un programma affidabile per eventuali meeting.
Il team ha usufruito di Google Meet durante le prime riunioni con l'azienda.

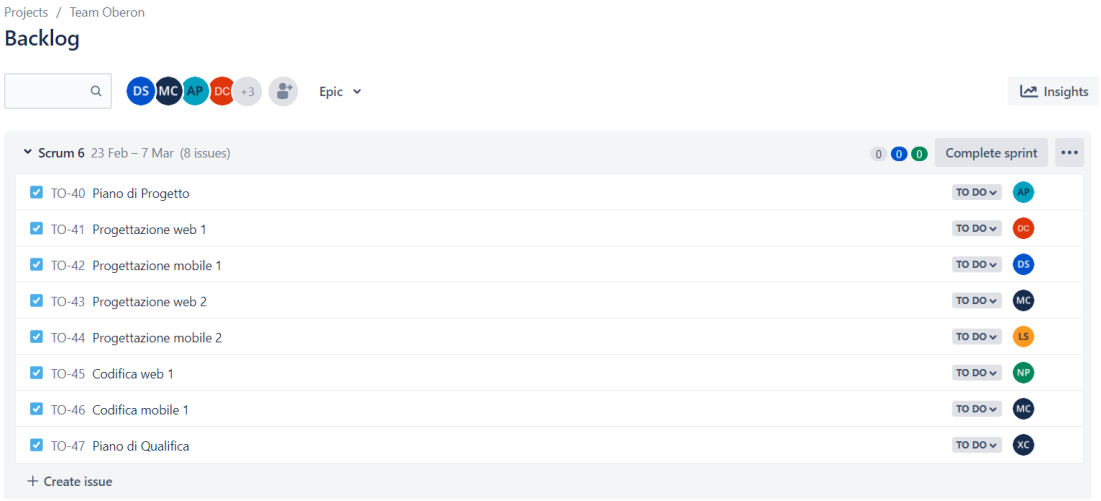
5.5.4 GitHub

Il servizio hosting per progetti software GitHub è parso l'ideale per il versionamento del capitolato assegnato al team in quanto molto popolare per questi utilizzi. Inoltre il gruppo possiede una certa familiarità con esso. Ogni membro avrà una copia aggiornata del repository sulla quale lavorare. La regola principale imposta nell'utilizzo di GitHub consiste nell'effettuare commit solo nel caso di modifiche sostanziali ai file.

5.5.5 Jira

Ticketing Jira offre un sistema di ticketing con il quale è possibile assegnare i vari compiti legati al progetto ai membri. La maggior parte del tracking della dashboard è proprio legata alle issue create dal project manager del periodo e assegnate di comune accordo durante il meeting di chiusura scrum precedente.

I ticket creati rappresentano singole attività da svolgere dal componente a cui sono state assegnate nel periodo di uno scrum specifico. Nel caso di due individui a cui è assegnata la stessa attività si applica una numerazione ordinata.



Dashboard Jira offre una funzionalità dashboard inclusa che raccoglie automaticamente i dati dalle issue create ed è particolarmente adatta al metodo di sviluppo di tipo scrum. Mantenere un'aggiornata rappresentazione grafica della distribuzione della mole di lavoro è fondamentale per tenere sotto controllo l'andamento del progetto e facilita la suddivisione dei compiti.

Il project manager ha il compito di dedicare parte della sua attenzione alla dashboard per assicurare un buono svolgimento.

Nel cruscotto del team Oberon si trovano:

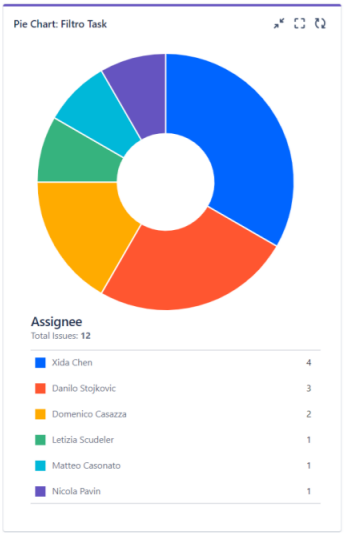


Figura 1 - Diagramma a torta contenente la distribuzione dei compiti attivi nello scrum attuale

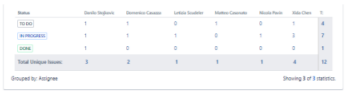


Figura 4 - Stato Issue assegnate ad ogni membro

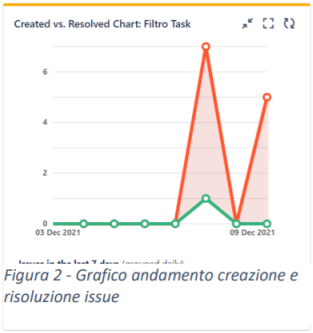


Figura 2 - Grafico andamento creazione e risoluzione issue

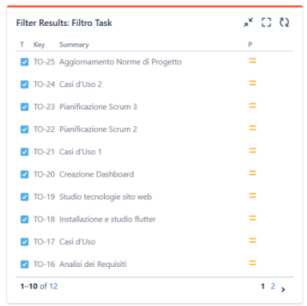


Figura 5 - Elenco issue dello scrum

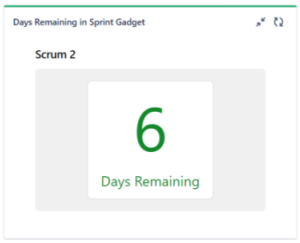


Figura 3 - Contatore giorni rimanenti nello scrum

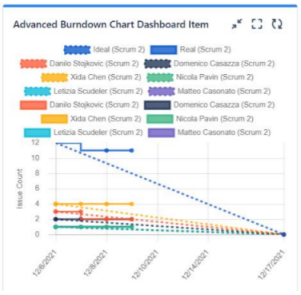


Figura 6 - Stato burndown totale e di ogni membro

5.5.6 TeamGantt

Questo software è stato scelto per la produzione di diagrammi di Gantt per la sua semplicità e per il fatto che è gratuito; è largamente sufficiente per gli scopi del gruppo.

5.5.7 StarUML

Gli schemi UML, diagrammi dei casi d'uso e di attività verranno prodotti con questo software.

5.6 Ruoli di progetto

5.6.1 Rotazione dei ruoli

Al termine di ogni scrum avverrà una rotazione dei ruoli. I ruoli disponibili sono Responsabile, Amministratore, Analista, Progettista, Programmatore e Verificatore.

Ciascun componente del team dovrà:

- assicurarsi di svolgere equamente ciascun ruolo (ogni ruolo almeno una volta, e non persistere sugli stessi ruoli);
- lavorare un numero di ore equo a quello degli altri componenti del team;

- mantenere una certa flessibilità, ossia in caso di necessità svolgere altri ruoli per aiutare i componenti del team in difficoltà.

5.6.2 Responsabile di progetto

Il responsabile di progetto assicura che le attività pianificate vengano portate a termine nei tempi stabiliti e rispettando il Way of Working del gruppo. Egli inoltre gestisce la comunicazione con il proponente ed il committente. Altri compiti che lo interessano sono:

- Assegnazione delle attività ai componenti del gruppo;
- Analisi e correzione dei processi interni al gruppo;
- Prevenzione dei rischi;
- Approvazione della documentazione.

5.6.3 Amministratore di progetto

L'amministratore di progetto gestisce l'ambiente di lavoro ed i relativi strumenti utilizzati dai componenti del gruppo. Le attività che lo interessano sono:

- Gestione versionamento ed archiviazione della documentazione;
- Gestione versionamento ed archiviazione del codice sorgente;
- Controllo dell'efficienza degli strumenti di lavoro;
- Redazione del documento *Norme di Progetto*.

5.6.4 Analista

L'analista ha il compito di determinare i requisiti del progetto. In particolare, egli deve:

- Analizzare il dominio del problema;
- Individuare i requisiti obbligatori e desiderabili;
- Redigere il documento *Analisi dei Requisiti*.

5.6.5 Progettista

Il progettista gestisce l'insieme di tecniche e tecnologie del progetto, seguendone ogni fase di sviluppo. I suoi compiti consistono sono:

- Individuare un'architettura opportuna alla realizzazione del prodotto;
- Garantire il soddisfacimento dei requisiti individuati dagli analisti.

5.6.6 Programmatore

Il programmatore realizza il codice necessario allo sviluppo del progetto, facendo particolare attenzione a rispettare le specifiche individuate dal progettista. Responsabilità del programmatore sono:

- Produrre codice quanto più possibile mantenibile;
- Documentare in maniera appropriata il suddetto codice.

5.6.7 Verificatore

Il verificatore è un ruolo presente in ogni istante del progetto ed ha le seguenti responsabilità:

- Controllare che ogni artefatto prodotto dal gruppo rispetti quanto stabilito dal documento *Norme di Progetto*;
- Segnalare eventuali errori all'autore dell'artefatto al fine di correggerli.

5.7 Gestione e codifica dei rischi

I rischi di progetto vanno individuati dal responsabile, che dovrà renderli noti e descriverli all'interno del piano di progetto. Occorrerà inoltre monitorare i rischi già individuati, e ridefinire eventualmente le strategie di gestione.

Per quanto riguarda la codifica, il formato del rischio è il seguente:

[Sigla][Numero]: Nome rischio

e le sigle possibili sono UM (rischio di natura umana) o TE (rischio di natura tecnologica).