



Zeus Code - Progetto "P2PCS"

## Norme di Progetto

<b>Versione</b>	1.0.0
<b>Approvazione</b>	Andrea Pigatto
<b>Redazione</b>	Marco Dalla B� Riccardo Basso Irina Hornoiu
<b>Verifica</b>	Riccardo Dario Diba Meysami
<b>Stato</b>	Approvato
<b>Uso</b>	Interno
<b>Destinato a</b>	Zeus Code Prof. Tullio Vardanega Prof. Riccardo Cardin

### **Descrizione**

*Norme di Progetto* del gruppo Zeus Code.

zeuscode17@gmail.com

## Diario delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
1.0.0	2018-11-30	Andrea Pigatto	<i>Responsabile del progetto</i>	Approvazione del documento per RR.
0.5.0	2018-11-27	Samuele Giuliano Piazzetta	<i>Verificatore</i>	Quinta revisione generale.
0.4.1	2018-11-25	Giacomo Greggio	<i>Amministratore</i>	Aggiornamento § 3.
0.4.0	2018-11-24	Mattia Bolzonella	<i>Verificatore</i>	Quarta revisione generale.
0.3.2	2018-11-24	Federico Biciato	<i>Amministratore</i>	Aggiornato § 4.
0.3.1	2018-11-24	Federico Biciato	<i>Amministratore</i>	Aggiunta § 3.5.
0.3.0	2018-11-24	Mattia Bolzonella	<i>Verificatore</i>	Terza revisione generale.
0.2.5	2018-11-24	Giacomo Greggio	<i>Amministratore</i>	Aggiustamento delle tabelle e indici.
0.2.4	2018-11-24	Federico Biciato	<i>Amministratore</i>	Aggiornamento § 3.3.
0.2.3	2018-11-23	Sara Feltrin	<i>Amministratore</i>	Aggiornamento § 2.2.
0.2.2	2018-11-23	Paolo Pozzan	<i>Amministratore</i>	Completata stesura § 3.3.
0.2.1	2018-11-23	Giacomo Greggio	<i>Amministratore</i>	Iniziata stesura § 3.3.
0.2.0	2018-11-22	Mattia Bolzonella	<i>Verificatore</i>	Seconda revisione generale.
0.1.4	2018-11-22	Giacomo Greggio	<i>Amministratore</i>	Modifiche a § 4.
0.1.3	2018-11-21	Giacomo Greggio	<i>Amministratore</i>	Stesura § 3.
0.1.2	2018-11-21	Giacomo Greggio	<i>Amministratore</i>	Sistematte sottosezioni § 2.

Versione	Data	Nominativo	Ruolo	Descrizione
0.1.1	2018-11-21	Federico Biciato	<i>Amministratore</i>	Iniziata stesura § 3.
0.1.0	2018-11-20	Mattia Bolzonella	<i>Verificatore</i>	Prima revisione generale.
0.0.5	2018-11-20	Giacomo Greggio	<i>Amministratore</i>	Aggiornata struttura documento con integrazione di alcuni paragrafi.
0.0.4	2019-03-13	Irina Hornoiu, Riccardo Basso	<i>Amministratori</i>	Iniziata stesura § 3.
0.0.3	2019-03-12	Marco Dalla B�	<i>Amministratore</i>	Continuazione § 2.
0.0.2	2019-03-12	Irina Hornoiu	<i>Amministratore</i>	Stesura § 1 e iniziato stesura § 2.
0.0.1	2019-03-11	Irina Hornoiu	<i>Amministratore</i>	Creata struttura documento LATEX.

## Indice

<b>1</b>	<b>Introduzione</b>	<b>8</b>
1.1	Scopo del documento . . . . .	8
1.2	Scopo del prodotto . . . . .	8
1.3	Glossario . . . . .	8
1.4	Riferimenti . . . . .	8
1.4.1	Riferimenti normativi . . . . .	8
1.4.2	Riferimenti informativi . . . . .	8
<b>2</b>	<b>Processi primari</b>	<b>9</b>
2.1	Fornitura . . . . .	9
2.1.1	Scopo . . . . .	9
2.1.2	Aspettative . . . . .	9
2.1.3	Gestione della Qualità . . . . .	9
2.1.4	Descrizione . . . . .	9
2.1.5	Attività . . . . .	10
2.1.5.1	Studio di Fattibilità . . . . .	10
2.1.5.2	Piano di Progetto . . . . .	10
2.1.5.3	Piano di Qualifica . . . . .	10
2.1.6	Strumenti . . . . .	11
2.1.6.1	Microsoft Excel . . . . .	11
2.1.6.2	Microsoft Project . . . . .	11
2.2	Sviluppo . . . . .	12
2.2.1	Scopo . . . . .	12
2.2.2	Aspettative . . . . .	12
2.2.3	Descrizione . . . . .	12
2.2.4	Attività . . . . .	12
2.2.4.1	Analisi dei Requisiti . . . . .	12
2.2.4.2	Progettazione . . . . .	15
2.2.4.3	Codifica . . . . .	16
2.2.5	Strumenti . . . . .	17
2.2.5.1	PragmaDB . . . . .	17
2.2.5.2	Draw.io . . . . .	17
2.2.5.3	Android Studio . . . . .	17
2.2.5.4	Adobe Photoshop . . . . .	17
2.2.5.5	Movens . . . . .	18
2.2.5.6	Travis CI . . . . .	18
<b>3</b>	<b>Processi di Supporto</b>	<b>19</b>
3.1	Documentazione . . . . .	19
3.1.1	Scopo . . . . .	19
3.1.2	Aspettative . . . . .	19
3.1.3	Descrizione . . . . .	19
3.1.4	Template . . . . .	19
3.1.5	Struttura dei documenti . . . . .	19
3.1.5.1	Prima pagina . . . . .	19
3.1.5.2	Registro modifiche . . . . .	20

3.1.5.3	Indice . . . . .	20
3.1.5.4	Contenuto principale . . . . .	21
3.1.5.5	Verbali . . . . .	21
3.1.5.6	Note a piè di pagina . . . . .	22
3.1.6	Norme tipografiche . . . . .	22
3.1.6.1	Convenzioni sui nomi dei file . . . . .	22
3.1.6.2	Glossario . . . . .	22
3.1.6.3	Stile del testo . . . . .	23
3.1.6.4	Elenchi puntati . . . . .	23
3.1.6.5	Formati comuni . . . . .	23
3.1.6.6	Sigle . . . . .	24
3.1.7	Elementi grafici . . . . .	25
3.1.7.1	Tabelle . . . . .	25
3.1.7.2	Immagini . . . . .	25
3.1.7.3	Diagrammi UML . . . . .	25
3.1.8	Strumenti . . . . .	25
3.1.8.1	L <sup>A</sup> T <sub>E</sub> X . . . . .	25
3.1.8.2	T <sub>E</sub> Xstudio . . . . .	25
3.1.8.3	Draw.io . . . . .	26
3.2	Gestione della configurazione . . . . .	26
3.2.1	Scopo . . . . .	26
3.2.2	Ciclo di vita della documentazione . . . . .	26
3.2.2.1	Stadi del documento . . . . .	26
3.2.3	Versionamento . . . . .	27
3.2.3.1	Codice di versione del documento . . . . .	27
3.2.3.2	Tecnologie . . . . .	27
3.2.3.3	Repository . . . . .	27
3.2.3.4	Struttura del repository . . . . .	28
3.2.3.5	Tipi di file e .gitignore . . . . .	28
3.2.3.6	Utilizzo di Git . . . . .	28
3.2.3.7	Gestione delle modifiche . . . . .	29
3.3	Verifica . . . . .	29
3.3.1	Aspettative . . . . .	29
3.3.2	Descrizione . . . . .	29
3.3.3	Attività . . . . .	29
3.3.3.1	Analisi . . . . .	29
3.3.3.2	Test . . . . .	30
3.3.4	Strumenti . . . . .	31
3.3.4.1	Verifica ortografica . . . . .	31
3.4	Validazione . . . . .	32
3.4.1	Scopo . . . . .	32
3.4.2	Attività . . . . .	32
<b>4</b>	<b>Processi Organizzativi</b>	<b>33</b>
4.1	Gestione Organizzativa . . . . .	33
4.1.1	Scopo . . . . .	33
4.1.2	Aspettative . . . . .	33
4.1.3	Descrizione . . . . .	33
4.1.4	Ruoli di progetto . . . . .	33

4.1.4.1	Responsabile di progetto . . . . .	34
4.1.4.2	Amministratore di progetto . . . . .	34
4.1.4.3	Analista . . . . .	34
4.1.4.4	Progettista . . . . .	34
4.1.4.5	Programmatore . . . . .	35
4.1.4.6	Verificatore . . . . .	35
4.1.5	Procedure . . . . .	35
4.1.5.1	Gestione delle comunicazioni . . . . .	35
4.1.5.2	Gestione degli incontri . . . . .	36
4.1.5.3	Gestione degli strumenti di coordinamento . . . . .	36
4.1.5.4	Gestione dei rischi . . . . .	37
4.1.6	Strumenti . . . . .	37

## Elenco delle figure

2.1.1 Microsoft Project: Gantt . . . . .	11
3.1.1 T <sub>E</sub> Xstudio - per la stesura dei documenti . . . . .	26

## Elenco delle tabelle



# 1 Introduzione

## 1.1 Scopo del documento

Questo documento ha lo scopo di definire le regole di base che tutti i membri di Zeus Code devono rispettare nello svolgimento del progetto, così da garantire uniformità in tutto il materiale. Verrà utilizzato un approccio incrementale<sub>G</sub>, volto a normare passo passo ogni decisione discussa e concordata tra tutti i membri del gruppo. Ciascun componente è obbligato a prendere visione di tale documento e a rispettare le norme in esso descritte allo scopo di perseguire la coesione<sub>G</sub> all'interno del team.

## 1.2 Scopo del prodotto

Il capitolato<sub>G</sub> C5 ha per obiettivo l'arricchimento delle funzionalità dell'app GaiaGo già esistente, inserendo un nuovo servizio di Car Sharing Peer-to-Peer<sub>G</sub> per la piattaforma Android. Il servizio, dunque, intende offrire la possibilità di condividere la propria macchina con altre persone amiche o meno sfruttando almeno 5 core drive del framework<sub>G</sub> Octalysis<sub>G</sub> allo scopo di motivare l'utente a condividere la propria auto con altri.

## 1.3 Glossario

Al fine di evitare ambiguità e facilitare la comprensione dei documenti formali viene incluso il Glossario v1.0.0, in cui saranno presenti acronimi, abbreviazioni e termini tecnici. Ogni termine presente nel glossario sarà marcato con una G a pedice.

## 1.4 Riferimenti

### 1.4.1 Riferimenti normativi

- **Standard ISO/IEC 12207:1995:**  
[https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO\\_12207-1995.pdf](https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf);
- **Capitolato<sub>G</sub> d'appalto C5 - Piattaforma Peer-to-Peer<sub>G</sub> car sharing:**  
<https://www.math.unipd.it/~tullio/IS-1/2018/Progetto/C5.pdf>;

### 1.4.2 Riferimenti informativi

- **Piano di Progetto:**
- **Piano di Qualifica:**
- **Software Engineering - Ian Sommerville - 10th Edition:**  
(formato cartaceo);
- **Octalysis<sub>G</sub> - Gamification<sub>G</sub> framework<sub>G</sub>**  
<https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/>

## 2 Processi primari

### 2.1 Fornitura

#### 2.1.1 Scopo

Il processo<sub>G</sub> di fornitura determina le procedure e le risorse necessarie allo svolgimento del progetto. Una volta comprese le richieste del proponente, *GaiaGo*, e aver stilato uno *Studio di Fattibilità*, si potrà procedere con il fine di soddisfare ognuna di queste richieste. Sarà necessario concordare con il proponente un contratto per la consegna del prodotto. Verrà sviluppato un *Piano di Progetto* che da seguire fino alla consegna del materiale prodotto. Il processo di fornitura è composto dalle seguenti fasi:

- avvio;
- approntamento di risposte alle richieste;
- contrattazione;
- pianificazione;
- esecuzione e controllo;
- revisione e valutazione;
- consegna e completamento.

#### 2.1.2 Aspettative

Al fine di avere un rapporto costante ed efficace con il proponente, il gruppo si impegna a:

- determinare aspetti chiave per far fronte ai bisogni del proponente;
- stilare requisiti e vincoli sui processi;
- stimare le tempistiche di lavoro;
- stimare i costi;
- promuovere una verifica continua;
- accordarsi sulla qualifica del prodotto.

#### 2.1.3 Gestione della Qualità

Il fornitore si impegna ad utilizzare strategie di verifica e validazione con l'obiettivo di garantire efficacia e qualità nei processi e nei prodotti. I progettisti hanno a carico la scelta di queste strategie e le descrivono nel Piano di qualifica v1.0.0. I verificatori invece si occupano di documentare l'esito delle verifiche e prove effettuate seguendo quanto deciso nel Piano di qualifica v1.0.0.

#### 2.1.4 Descrizione

Questa sezione tratta le norme che i membri del gruppo *Zeus Code* devono rispettare in tutte le fasi di progettazione, sviluppo e consegna del prodotto *P2PCS*, al fine di diventare fornitori nei confronti del proponente *GaiaGo* e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin.

## 2.1.5 Attività

### 2.1.5.1 Studio di Fattibilità

Il responsabile di progetto ha l'onere di organizzare riunioni tra i membri del gruppo per discutere sui capitolati proposti. Lo *Studio di Fattibilità*, redatto dagli analisti, indica per ogni capitolato<sub>G</sub>:

- **Informazioni generali:** vengono elencate le informazioni di base, come il nome del progetto, il proponente e il committente;
- **Descrizione e finalità del progetto:** viene fatta una presentazione del capitolato in generale, una descrizione delle caratteristiche principali richieste per il prodotto e viene definito l'obiettivo che si vuole raggiungere;
- **Tecnologie interessate:** viene fatto un elenco delle tecnologie richieste per lo svolgimento, che rientrano nel dominio tecnologico;
- **Aspetti positivi, criticità e fattori di rischio:** vengono espresse le considerazioni fatte dal gruppo sugli aspetti positivi e sui fattori di rischio del capitolato;
- **Conclusioni:** vengono espresse le ragioni per la quale il gruppo ha deciso di accettare o scartare il capitolato.

### 2.1.5.2 Piano di Progetto

Il responsabile, con l'aiuto degli amministratori, redige un *Piano di Progetto* da seguire durante il corso del progetto. Questo documento contiene:

- **Analisi dei rischi:** vengono analizzati nel dettaglio i rischi che potranno presentarsi e vengono espresse le misure e le modalità attraverso le quali i rischi vengono contenuti o mitigati. Viene anche fornita la probabilità con la quale questi possono presentarsi e il livello di gravità per ciascuno;
- **Modello di sviluppo<sub>G</sub>:** viene descritto il modello di sviluppo che è stato scelto, indispensabile per la pianificazione;
- **Pianificazione:** vengono pianificate le attività da eseguire nelle diverse fasi del progetto e vengono stabilite le loro scadenze temporali;
- **Preventivo e consuntivo:** viene data una stima di lavoro necessaria per ciascuna fase proponendo così un preventivo per il costo totale del progetto. Viene anche tracciato, un consuntivo di periodo relativo all'andamento rispetto a ciò che è stato preventivato.

### 2.1.5.3 Piano di Qualifica

I verificatori dovranno redigere un documento, detto *Piano di Qualifica* contenente le strategie da adottare per garantire la qualità del materiale prodotto dal gruppo, e dei processi attuati. Il piano è così suddiviso:

- **Qualità di processo:** vengono identificati dei processi dagli standard, stabiliti degli obiettivi, escogitate delle strategie per attuarli e individuate le metriche per misurarli e controllarli;

- **Qualità di prodotto:** vengono identificati gli attributi più rilevanti per il prodotto, definiti degli obiettivi per raggiungerli e delle metriche per misurarli;
- **Specifiche dei test:** definiscono una serie di test attraverso i quali il prodotto passa per garantire che soddisfi i requisiti;
- **Standard di qualità:** vengono esposti gli standard di qualità scelti;
- **Valutazioni per il miglioramento:** vengono riportati i problemi e le relative soluzioni nel ricoprire un determinato ruolo e nell'uso degli strumenti scelti;
- **Resoconto delle attività di verifica:** per ogni attività si riportano i risultati delle metriche calcolate in forma di resoconto.

## 2.1.6 Strumenti

Di seguito sono elencati gli strumenti utilizzati durante il processo di fornitura.

### 2.1.6.1 Microsoft Excel

Software della suite Microsoft Office per realizzare fogli elettronici. Usato per fare calcoli, produrre diagrammi, istogrammi e areogrammi, creare tabelle e grafici.

### 2.1.6.2 Microsoft Project

Per assistere i responsabili di progetto nella pianificazione, nell'assegnazione delle risorse, nella verifica del rispetto dei tempi, nella gestione dei budget e nell'analisi dei carichi di lavoro attraverso la creazione di diagrammi di Gantt, è stato utilizzato Microsoft Project.

<https://products.office.com/it-it/project/>

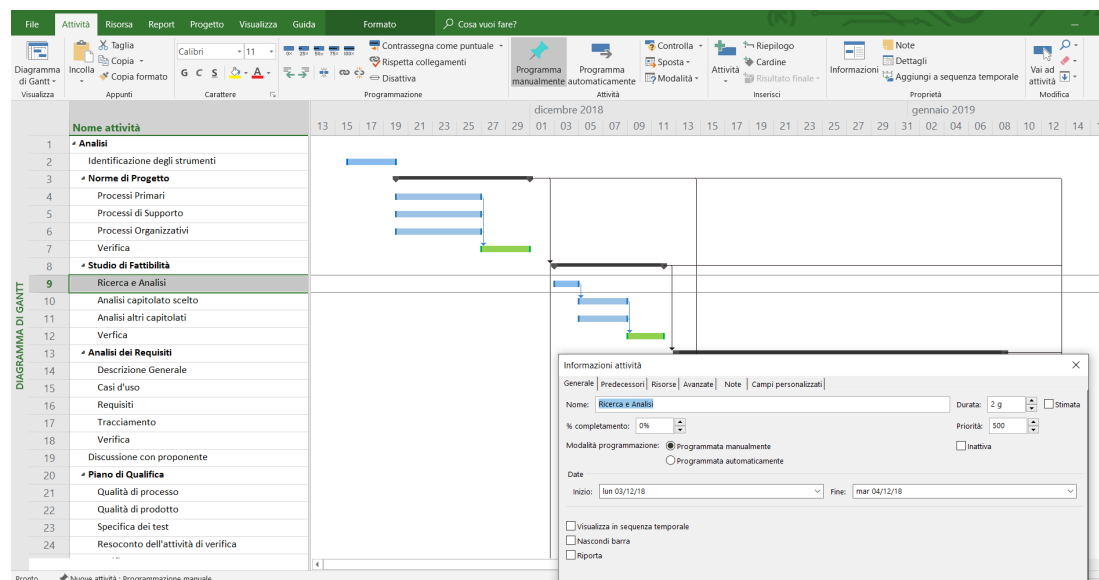


Figura 2.1.1: Microsoft Project: Gantt

## 2.2 Sviluppo

### 2.2.1 Scopo

Il processo contiene le attività e i compiti da svolgere, al fine di realizzare il prodotto finale richiesto dal proponente.

### 2.2.2 Aspettative

Le aspettative sono le seguenti:

- fissare gli obiettivi di sviluppo;
- fissare i vincoli tecnologici;
- fissare i vincoli di design;
- realizzare un prodotto finale che superi i test, che soddisfi i requisiti e le richieste del proponente.

### 2.2.3 Descrizione

Il processo di sviluppo si articola in:

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

### 2.2.4 Attività

#### 2.2.4.1 Analisi dei Requisiti

##### Scopo

Gli analisti hanno il compito di redigere il documento di *Analisi dei Requisiti* che individua ed elenca dunque, i requisiti. Lo scopo dei requisiti è quello di:

- definire lo scopo del lavoro;
- fornire ai progettisti riferimenti precisi ed affidabili;
- fissare le funzionalità e i requisiti concordati col cliente;
- fornire una base per raffinamenti successivi al fine di garantire un miglioramento continuo del prodotto e del processo di sviluppo;
- fornire ai verificatori riferimenti per l'attività di controllo dei test;
- calcolare la mole di lavoro per tracciare dei riferimenti per una stima dei costi.

## Aspettative

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

## Descrizione

I requisiti si raccolgono secondo modalità predefinite:

- lettura del capitolato<sub>G</sub>, analisi e approfondimento dello stesso;
- confronto con il proponente;
- confronto tra membri del team di progetto;
- analisi di uno o più casi d'uso.

## Casi d'uso

Rappresenta un diagramma che esprime un comportamento, offerto o desiderato, sulla base di risultati osservabili. La struttura dei casi d'uso è così suddivisa:

- codice identificativo;
- titolo;
- diagramma UML<sub>G</sub>;
- attori primari;
- attori secondari;
- descrizione;
- scenario principale;
- scenario alternativo (se presente);
- inclusioni(se presenti);
- estensioni(se presenti);
- specializzazioni(se presenti);
- precondizione;
- postcondizione.

## Codice identificativo dei casi d'uso

Il codice di ogni caso d'uso seguirà questo formalismo:

**UC[codice \_padre].[codice \_figlio]**

Dove:

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

## Requisiti

Ogni requisito è composto dalla seguente struttura:

- **codice identificativo**: ogni codice identificativo è univoco e conforme alla seguente codifica:

**R[Importanza][Tipologia][Codice]**

Il significato delle cui voci è:

- **Importanza**: ogni requisito può assumere uno dei seguenti valori:
    - \* *1*: requisito obbligatorio, ovvero irrinunciabile per gli stakeholder;
    - \* *2*: requisito desiderabile, ovvero non strettamente necessari ma a valore aggiunto riconoscibile;
    - \* *3*: requisito opzionale, ovvero relativamente utile oppure contrattabile più avanti nel progetto;
  - **Tipologia**: ogni requisito può assumere uno dei seguenti valori:
    - \* *F*: funzionale;
    - \* *Q*: prestazionale;
    - \* *P*: qualitativo;
    - \* *V*: vincolo.
  - **Codice**: è un identificatore univoco del requisito in forma gerarchica padre/figlio.
- **classificazione**: viene riportata l'importanza del requisito. Sebbene questa sia un'informazione ridondante ne facilita la lettura;
  - **descrizione**: descrizione breve ma completa del requisito, meno ambigua possibile;
  - **fonti**: ogni requisito può derivare da una o più tra le seguenti opzioni:
    - *capitolato<sub>G</sub>*: si tratta di un requisito individuato dalla lettura del capitolato;
    - *interno*: si tratta di un requisito che gli analisti hanno ritenuto opportuno aggiungere;
    - *caso d'uso*: il requisito è estrapolato da uno o più casi d'uso. In questo caso deve essere riportato il codice univoco del caso d'uso;
    - *verbale*: si tratta di un requisito individuato in seguito ad una richiesta di chiarimento con il proponente. Tali informazioni sono riportate nei verbali in cui ogni requisito individuato è segnato da un codice presente nella tabella dei tracciamenti.

## UML

I diagrammi UML<sub>G</sub> devono essere realizzati usando la versione del linguaggio v2.0.

### 2.2.4.2 Progettazione

#### Scopo

L'attività di progettazione definisce, in funzione dei requisiti specificati nel documento *Analisi dei Requisiti*, le caratteristiche del prodotto software richiesto. Il compito di questa fase è di definire una soluzione del problema che sia soddisfacente per tutti gli stakeholder. La progettazione segue il procedimento inverso rispetto all'*Analisi dei Requisiti* che divide il problema in parti per capirne completamente il dominio applicativo. La progettazione, infatti, rimette insieme le parti specificando le funzionalità dei sottosistemi in modo da ricondurre ad un'unica possibile soluzione.

#### Aspettative

Il processo ha come risultato la realizzazione dell'architettura del sistema.

#### Descrizione

Le parti principali sono due:

- **Technology baseline:** contiene le specifiche della progettazione ad alto livello del prodotto e delle sue componenti, l'elenco dei diagrammi UML<sub>G</sub> che saranno utilizzati per la realizzazione dell'architettura e i test di verifica;
- **Product baseline:** dettaglia ulteriormente l'attività di progettazione, integrando ciò che è riportato nella Technology baseline. Inoltre definisce i test necessari alla verifica.

#### Technology baseline

Redatta dal progettista, dovrà includere:

- **Diagrammi UML<sub>G</sub>:**
  - diagrammi delle classi;
  - diagrammi dei package;
  - diagrammi di attività;
  - diagrammi di sequenza.
- **Tecnologie utilizzate:** devono essere descritte le tecnologie adottate specificandone l'utilizzo nel progetto, i vantaggi e gli svantaggi;
- **Design pattern<sub>G</sub>:** devono essere descritti i design pattern 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;
- **Test di integrazione:** l'unione delle parti, intese come classi di verifica, permette di verificare che ogni componente del sistema funzioni nella maniera voluta.



## Product baseline

A carico del progettista c'è anche la Product baseline che si sofferma su diversi aspetti tra i quali:

- **definizione delle classi:** ogni classe 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 per ognuno esista una classe che lo soddisfi. Questa operazione è fondamentale per permettere di risalire alle classi ad esso associate;
- **test di unità:** devono essere definiti al fine di verificare che le parti funzionino individualmente nel modo stabilito.

### 2.2.4.3 Codifica

#### Scopo

Questa attività ha come scopo quello di normare l'effettiva realizzazione del prodotto software richiesto. In questa fase si concretizza la soluzione attraverso la programmazione. I programmatori dovranno attenersi a queste norme durante la fase di programmazione ed implementazione.

#### Aspettative

Obiettivo dell'attività è la creazione di un prodotto software conforme alle richieste prefissate con il proponente. L'uso di norme e convenzioni in questa fase, è fondamentale per permettere la generazione di codice leggibile ed uniforme, agevolare le fasi di manutenzione, verifica e validazione e migliorare la qualità di prodotto.

#### Descrizione

La scrittura del codice dovrà rispettare quanto stabilito nella documentazione di prodotto. Dovrà perseguire gli obiettivi di qualità definiti all'interno del documento *Piano di Qualifica v1.0.0* per poter garantire una buona qualità del codice.

#### Stile di codifica

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

- **Indentazione:** i blocchi innestati devono essere correttamente indentati, usando per ciascun livello di indentazione quattro (4) spazi (fanno eccezione i commenti). Al fine di assicurare il rispetto di questa regola si consiglia di configurare adeguatamente il proprio editor o IDE;
- **Parentesizzazione:** è richiesto di inserire le parentesi di delimitazione dei costrutti in linea e non al di sotto di essi;
- **Scrittura dei metodi:** è desiderabile, ove possibile, mantenere i metodi brevi (poche righe di codice);

- **Univocità dei nomi:** classi, metodi, variabili devono avere un nome univoco ed esplicativo al fine di evitare ambiguità e incomprensione;
- **Classi:** i nomi delle classi devono iniziare sempre con una lettera maiuscola;
- **Costanti:** i nomi delle costanti devono essere scritte usando solo maiuscole;
- **Metodi:** i nomi dei metodi devono iniziare con una lettera minuscola. Nel caso siano composti da più parole, quelle successive devono iniziare con una lettera maiuscola (CamelCase<sub>G</sub>);
- **Lingua:** il codice, come anche i commenti, deve essere scritto in lingua inglese.

Una parte del front end<sub>G</sub> è normata dalla "Airbnb JavaScript style guide", il cui uso è implementato attraverso ESLint<sub>G</sub>.

## Ricorsione

L'uso della ricorsione va evitato quanto più possibile in quanto potrebbe indurre ad una maggiore occupazione di memoria rispetto a soluzioni iterative.

### 2.2.5 Strumenti

Di seguito sono elencati gli strumenti utilizzati dal gruppo durante il progetto per il processo di sviluppo.

#### 2.2.5.1 PragmaDB

Programma usato per il tracciamento dei requisiti, fondamentale dunque per la stesura del documento *Analisi dei Requisiti v1.0.0*.

<https://github.com/StefanoMunari/PragmaDB>

#### 2.2.5.2 Draw.io

Per la produzione di diagrammi UML<sub>G</sub> viene utilizzato Draw.io in quanto offre molte agevolazioni per la produzione veloce dei diagrammi e risulta semplice da usare.

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

#### 2.2.5.3 Android Studio

Android Studio è un ambiente di sviluppo integrato per lo sviluppo per la piattaforma Android. Il linguaggio utilizzato in questo ambiente è Kotlin.

<https://developer.android.com/studio>

#### 2.2.5.4 Adobe Photoshop

Adobe Photoshop è un software per l'elaborazione di immagini, sarà utilizzato per progettare la GUI.

<https://www.adobe.com/it/products/photoshop.html>

#### **2.2.5.5 Movens**

Movens è una piattaforma pensata per la gestione delle smart cities. Verrà utilizzata per la gestione degli utenti e dell'applicazione.

<https://www.adobe.com/it/products/photoshop.html>

#### **2.2.5.6 Travis CI**

Travis CI è un tool utilizzato per la Continuous Integration. È stato scelto perché open-source e gratuito.

<https://travis-ci.org/>

## 3 Processi di Supporto

### 3.1 Documentazione

#### 3.1.1 Scopo

Ogni processo e attività significativi volti allo sviluppo del progetto sono documentati. Lo scopo di questa sezione è definire gli standard che riguardano i documenti prodotti durante il ciclo di vita del software. I documenti sono consultabili nelle apposite sezioni della repository<sub>G</sub>:

<https://github.com/ZeusCode17/P2PCS>.

#### 3.1.2 Aspettative

Le aspettative su questo processo riguardano:

- un'idea precisa relativa alla struttura della documentazione che deve essere prodotta durante il ciclo di vita del software;
- l'individuazione di una serie di norme per la stesura di documenti coerenti e validi.

#### 3.1.3 Descrizione

Questo capitolo contiene le decisioni e le norme che sono state scelte per la stesura, verifica e approvazione della documentazione ufficiale. Tali norme sono tassative per tutti i documenti formali.

#### 3.1.4 Template

Il gruppo ha creato un template  $\text{\LaTeX}$  per uniformare velocemente la struttura grafica e lo stile di formattazione dei documenti, in modo che i membri del team possano concentrarsi maggiormente, nella stesura del contenuto degli stessi. Lo scopo dei template è quello di permettere, a colui che redige il documento, di adottare automaticamente le conformità previste dalle *Norme di Progetto*. Permette inoltre di agevolare la procedura di adeguamento alle nuove norme per la redazione, nel caso esse cambiassero.

#### 3.1.5 Struttura dei documenti

Un file "main.tex" (il cui termine "main" verrà sostituito dal nome del documento) raccoglie tramite comandi di input le sezioni di cui è composto il documento. Tra i file in input ci sono:

- "package.tex", che contiene i pacchetti necessari alla compilazione;
- "config.tex", contenente comandi  $\text{\LaTeX}$  creati dal team.

##### 3.1.5.1 Prima pagina

Il frontespizio è la prima pagina del documento ed è così strutturata:

- **Logo del gruppo:** logo di *Zeus Code* visibile come primo elemento centrato orizzontalmente in alto;
- **Gruppo e progetto:** nome del gruppo e del progetto *P2PCS*, visibile centralmente subito sotto il logo;

- **Titolo:** nome del documento, posizionato centralmente in grassetto;
- **Tabella:** presente sotto il titolo del documento, centrale e contenente le seguenti informazioni:
  - **Versione:** versione del documento;
  - **Approvazione:** nome e cognome dei membri del gruppo incaricati dell'approvazione del documento;
  - **Redazione:** nome e cognome dei membri del gruppo incaricati della redazione del documento;
  - **Verifica:** nome e cognome dei membri del gruppo incaricati della verifica del documento;
  - **Stato:** stadio corrente del ciclo di vita del documento;
  - **Uso:** tipo d'uso che può essere "interno" o "esterno";
  - **Destinatari:** destinatari del documento.
- **Descrizione:** descrizione sintetica relativa al documento, centrale, posta sotto la tabella descrittiva;
- **Recapito:** indirizzo di posta elettronica del gruppo, posizionato centralmente in fondo alla pagina.

### 3.1.5.2 Registro modifiche

Ogni documento dispone di un changelog<sub>G</sub>: una tabella posta a seguito della prima pagina che contiene le modifiche apportate al documento. Nel changelog<sub>G</sub> sono indicati:

- versione del documento dopo la modifica;
- data della modifica;
- nominativo di chi ha modificato;
- ruolo di chi ha modificato;
- descrizione sintetica della modifica.

### 3.1.5.3 Indice

Gli indici hanno lo scopo di riepilogare e dare una visione macroscopica della struttura del documento, mostrando le parti gerarchiche di cui è composto.

Ogni documento è corredato dall'indice dei contenuti, posizionato dopo il changelog<sub>G</sub>. Se sono presenti tabelle o immagini all'interno del documento, l'indice dei contenuti è seguito prima dalla lista delle figure, poi dalla lista delle tabelle.

#### 3.1.5.4 Contenuto principale

La struttura delle pagine di contenuto è così strutturata:

- in alto a sinistra è presente il logo del gruppo;
- in alto a destra è riportato il nome del documento;
- una riga divide l'intestazione dal contenuto;
- il contenuto della pagina è posto tra l'intestazione e il piè di pagina;
- una riga divide il contenuto dal piè pagina;
- in basso a destra è posto il numero della pagina corrente.

#### 3.1.5.5 Verbali

I verbali vengono prodotti dal/i soggetto/i incaricato/i alla loro stesura in occasione di incontri tra i membri del team con o senza la presenza di esterni. Per i verbali è prevista un'unica stesura. Tale scelta è motivata dal fatto che apportare modifiche implica una modifica delle decisioni prese in modo retroattivo. I verbali seguono la struttura degli altri documenti, ma il corpo del verbale è suddiviso in introduzione e contenuto. L'introduzione contiene:

- **Luogo:** luogo di svolgimento dell'incontro;
- **Data:** data dell'incontro(in formato YYYY-MM-DD);
- **Ora di inizio:** l'orario di inizio dell'incontro;
- **Ora di fine:** l'orario di fine dell'incontro;
- **Partecipanti:** l'elenco dei membri del gruppo che erano presenti all'incontro e, se presenti, i nominativi di persone esterne al gruppo che hanno partecipato. Un esempio, sono i nomi dei componenti di *GaiaGo*, a fianco al quale deve essere indicato "(proponente)";
- **Argomenti affrontati:** ciò di cui si è discusso durante l'incontro in forma riassuntiva e facendo risaltare gli argomenti principali.

Il contenuto è composto da:

- una descrizione più approfondita in merito agli argomenti trattati durante l'incontro;
- un riepilogo dei tracciamenti in forma tabellare che elenca le decisioni emerse: assegna ad ognuna di loro un codice e le descrive. Sarà ripresa successivamente.

Ogni *Verbale* dovrà essere denominato secondo il seguente formato:

**TipologiaYYYY-MM-DD**

dove per "Tipologia" si intende il tipo di verbale:

- **Interno:** concentrato sul riassunto dell'incontro dei membri del team;
- **Esterno:** concentrato sulla trattazione di argomenti con partecipanti esterni al gruppo, in particolare domande e risposte riguardanti il progetto in sè.

La sezione "Riepilogo tracciamenti" avrà la funzione di tenere traccia delle decisioni emerse da ogni incontro, sotto forma di tabella riassuntiva a fine di ogni verbale. Il formalismo utilizzato dalla tabella dovrà essere il seguente:

### Tipologia\_X.Y

Dove:

- per "Tipologia" basterà indicare l'iniziale della tipologia del verbale (I=Interno/E=Esterno);
- "X.Y": X si riferisce al numero dell'incontro mentre per Y si intende il numero progressivo della decisione presa dal gruppo (partendo da 1).

#### 3.1.5.6 Note a piè di pagina

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

#### 3.1.6 Norme tipografiche

##### 3.1.6.1 Convenzioni sui nomi dei file

I nomi di file (estensione esclusa) e cartelle utilizzano la convenzione "Snake case<sub>G</sub>" e alcune regole aggiuntive elencate di seguito:

1. i nomi dei file composti da più parole usano il carattere underscore come carattere separatore;
2. i nomi sono scritti interamente in minuscolo;
3. le preposizioni non si omettono.

Alcuni esempi **corretti** sono:

- studio\_di\_fattibilità;
- analisi\_dei\_requisiti.

Alcuni esempi **non corretti** sono:

- Norme\_di\_progetto (usa maiuscole);
- norme-di-progetto (carattere separatore errato);
- norme\_progetto (omette "di").

##### 3.1.6.2 Glossario

- ogni termine del *Glossario* è marcato con una **G** maiuscola a pedice in ogni sua occorrenza;
- se la voce è presente ripetutamente nello stesso paragrafo non è necessario marcarla in ogni sua occorrenza, ma è possibile indicare la **G** a pedice solo nella prima occorrenza;

- se nel *Glossario v1.0.0* un termine presenta una descrizione che utilizza termini da glossario, è necessario trattare questi termini come tali, segnando la **G** a pedice e aggiungendoli al documento con la relativa descrizione;
- non vengono segnate con la **G** a pedice le parole da *Glossario* presenti nei titoli e nelle didascalie di immagini e tabelle.

### 3.1.6.3 Stile del testo

- **Grassetto:** viene applicato se necessario alle voci di un elenco puntato, a titoli o a termini di frasi su cui si vuol far ricadere l'attenzione del lettore;
- **Corsivo:** vengono scritti in corsivo il nome del progetto *P2PCS*, il nome del gruppo *Zeus Code* ed il nome del gruppo dei proponenti, ovvero *GaiaGo*;
- **Maiuscolo:** vengono scritti con sole lettere maiuscole tutti gli acronimi. Nel caso di nomi o titoli composti da più parole verrà indicato con la lettera maiuscola solamente la prima lettera della prima parola, lasciando in minuscolo il restante (esclusi nomi dei documenti che sono normati secondo quanto segue);
- **Nomi dei documenti:**
  - ogni volta che si cita un documento, si deve indicare con la lettera maiuscola le iniziali dei nomi di cui è composto (*e.g.* Analisi dei Requisiti), ma senza specificare la versione, riportando il tutto in corsivo;
  - quando si fa riferimento al documento vero e proprio o a qualcosa in esso contenuto si segue la prima convenzione ma si aggiunge la versione del documento v\*.0.0 (separata da uno spazio), anch'essa in corsivo;
  - ogni volta che si utilizza il documento come titolo o in una voce di elenco seguita da una descrizione, si deve seguire la prima convenzione ma senza utilizzare il corsivo.

### 3.1.6.4 Elenchi puntati

Ogni voce di un elenco comincia per lettera **minuscola**, a patto che non subentrino altre norme, e termina per ";", eccetto l'ultima che termina per ".". I sottoelenchi innestati dentro una voce di elenco, rispettano le medesime regole, poiché la loro funzione è analoga.

Se le voci dell'elenco sono della forma *termine - descrizione*, si pongono i termini in grassetto.

### 3.1.6.5 Formati comuni

In conformità allo standard ISO 8601, le date devono essere scritte secondo il formato:

YYYY-MM-DD

- **YYYY:** rappresentazione dell'anno con quattro (4) cifre;
- **MM:** rappresentazione del mese con due (2) cifre;
- **DD:** rappresentazione del giorno con due (2) cifre.



### 3.1.6.6 Sigle

Il progetto prevede la redazione di un insieme di documenti, suddivisi in documenti interni e documenti esterni. Essi sono elencati di seguito con le rispettive sigle.

I documenti esterni sono:

- **Analisi dei Requisiti - AdR**: stabilisce le caratteristiche che il software deve rispettare;
- **Manuale Utente - MU**: ad uso degli utilizzatori del software;
- **Manuale Sviluppatore - MS**: per gli sviluppatori e manutentori;
- **Piano di Progetto - PdP**: concerne la gestione del progetto, evidenziandone la fattibilità e le criticità; tratta di tempi, costi, obiettivi, rischi, vincoli;
- **Piano di Qualifica - PdQ**: : descrive la qualità del software e dei processi, e come la si intende raggiungere mediante l'uso di strumenti, metriche e processi stessi.

I documenti interni sono:

- **Glossario - G**: raccoglie i termini di interesse per il team di sviluppo e sui quali è necessaria una descrizione che ne chiarisca il significato;
- **Norme di Progetto - NdP**: sono un riferimento normativo per lo svolgimento delle attività di progetto;
- **Studio di Fattibilità - SdF**: descrive sommariamente i capitolati e spiega la loro scelta o esclusione.

Un caso particolare di documenti, sono i verbali, che possono essere esterni o interni:

- **Verbale - V**: descrive le interazioni avvenute durante un incontro con il proponente (verbale esterno) o tra i membri del team (verbale interno); sono orientati a dare informazioni semplici, di veloce lettura e complete.

Le diverse fasi del progetto sono le seguenti, accompagnate dalle relative sigle:

- **Revisione dei Requisiti - RR**: studio iniziale del capitolato<sub>G</sub>, se ben fatto permette al gruppo di aggiudicarselo;
- **Revisione di Progettazione - RP**: riguarda la definizione dell'architettura del software e di una Proof of Concept<sub>G</sub> per mostrare la fattibilità;
- **Revisione di Qualifica - RQ**: riguarda la definizione dettagliata e la codifica del prodotto;
- **Revisione di Accettazione - RA**: se il prodotto soddisfa i requisiti del proponente, viene accettato e rilasciato.

Altre sigle presenti nei documenti interessano i ruoli assunti dai componenti del team:

- **responsabile di progetto - Re**;
- **amministratore - Ad**;
- **analista - An**;
- **progettista - Pt**;
- **programmatore - Pr**;
- **verificatore - Ve**;

### 3.1.7 Elementi grafici

#### 3.1.7.1 Tabelle

In tutti i documenti  $\text{\LaTeX}$  le tabelle sono scritte allo stesso modo: fare riferimento alla Wiki su GitHub per i comandi necessari.

Ogni tabella deve essere accompagnata dalla propria didascalia descrittiva, la "caption", da posizionare subito sopra la tabella a cui si riferisce. È stata scelta questa convenzione per adattarsi agli standard usati dalla maggioranza della community di  $\text{\LaTeX}$ ; nella didascalia deve comparire il numero della sezione a cui si riferisce, seguita in modo incrementale dal numero progressivo delle tabelle di quella sezione.

- **X.Y**: rappresenta la sezione;
- **Z**: rappresenta il numero progressivo della tabella nella sezione.

Fanno eccezione le tabelle dei changelog<sub>G</sub> che non hanno didascalia e le tabelle dei casi d'uso presenti nel documento *Analisi dei Requisiti*.

#### 3.1.7.2 Immagini

Le immagini sono centrate e hanno una didascalia descrittiva.

#### 3.1.7.3 Diagrammi UML

Tutti i diagrammi  $\text{UML}_G$  vengono inseriti nei documenti sotto forma di immagine.

### 3.1.8 Strumenti

#### 3.1.8.1 $\text{\LaTeX}$

Lo strumento scelto per la scrittura di documenti è  $\text{\LaTeX}$ , un linguaggio basato sul programma di composizione tipografica  $\text{\TeX}$  che permette di scrivere documenti in modo ordinato, modulare, collaborativo e scalabile.

#### 3.1.8.2 $\text{\TeX}$ studio

Per la stesura del codice  $\text{\LaTeX}$  è stato utilizzato l'editor  $\text{\TeX}$ studio. Questo strumento, oltre ad integrare un compilatore e un visualizzatore PDF, fornisce suggerimenti di completamento per comandi  $\text{\LaTeX}$ .

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



Figura 3.1.1: T<sub>E</sub>Xstudio - per la stesura dei documenti

### 3.1.8.3 Draw.io

Draw.io, già citato in precedenza, viene utilizzato per la produzione degli UML<sub>G</sub>.

<https://www.draw.io/>

## 3.2 Gestione della configurazione

### 3.2.1 Scopo

Lo scopo ultimo della configurazione è di creare coesione tra software e documenti. Ogni file è caratterizzato da un identificativo univoco, si trova in uno stato di versionamento ed è modificato secondo determinate procedure.

### 3.2.2 Ciclo di vita della documentazione

#### 3.2.2.1 Stadi del documento

Ogni documento deve necessariamente passare per i seguenti stadi:

- **Sviluppo:** il documento entra in questo stato da quando viene creato fino al momento della verifica. In questo stato il documento viene redatto e sviluppato in ogni sua parte da un Redattore.

- **Verifica:** il documento entra in questo stato dopo la sua stesura fino al momento dell'approvazione. In questo stato il documento viene controllato in ogni sua parte e in caso di esito positivo passa alla fase di approvazione, in caso contrario verrà riassegnato ad un Redattore per apportare le eventuali migliorie o aggiunte.
- **Approvazione:** il documento entra in questo stato dopo la sua approvazione da parte di un Verificatore. In questo stato il documento viene approvato dal Responsabile del progetto a seguito del superamento della parte di verifica.

### 3.2.3 Versionamento

#### 3.2.3.1 Codice di versione del documento

Ogni documento è sottoposto a versionamento, ciò permette ai membri del gruppo, o ad utenti esterni, di risalire all'insieme delle modifiche effettuate su quest'ultimo. Ogni versione è riportata nel registro delle modifiche presente ad inizio documento. Il numero di versione è composto da tre cifre:

X.Y.Z

- **X:** versione ufficiale del documento resa tale dopo l'approvazione da parte del Responsabile di progetto:
  - inizia da 0;
  - viene incrementato dal responsabile di progetto all'approvazione del documento.
- **Y:** versione del documento soggetta a verifica da parte di un Verificatore:
  - inizia da 0;
  - viene incrementato dal Verificatore ad ogni verifica;
  - quando viene incrementato X, viene riportato a 0.
- **Z:** versione del documento in fase di lavorazione da parte dei Redattori:
  - inizia da 0;
  - viene incrementato dal redattore del documento ad ogni modifica;
  - quando viene incrementato Y, viene riportato a 0.

#### 3.2.3.2 Tecnologie

E' stato scelto il sistema Git per il versionamento dei file di progetto, la repository remota è invece ospitata dal servizio GitHub.

#### 3.2.3.3 Repository

L'accesso alla repository da parte dei membri del gruppo è possibile tramite linea di comando o con l'uso di software come GitHub Desktop e simili. La versione ufficiale e aggiornata della repository la si trova al seguente indirizzo di GitHub

<https://github.com/ZeusCode17/P2PCS>

### 3.2.3.4 Struttura del repository

Sono presenti due tipi di repository con uguale struttura:

- **locale**: repository locale clonata sul pc di ogni membro del gruppo nella quale vengono eseguite le modifiche.
- **remoto**: repository remota ospitata su GitHub aggiornata con tutte le modifiche apportate da ogni membro del gruppo.

Entrambi i tipi di repository<sub>G</sub> sono organizzati in cartelle:

- **latex**: contiene tutti i file che definiscono il template L<sup>A</sup>T<sub>E</sub>X per la creazione di un nuovo documento. Tra questi vi sono i file che definiscono i comandi personalizzati utilizzati dal gruppo e i package che devono essere inclusi per poter compilare i documenti;
- **RR**: raccoglie i file sorgenti per la compilazione dei documenti, suddivisi tra esterni ed interni, realizzati per la Revisione dei Requisiti;
- Saranno aggiunte in futuro cartelle distinte nominate **RP**, **RQ** e **RA** contenenti i file delle rispettive consegne.

Abbiamo scelto la suddivisione dei file per revisione, ciò evidenzia in modo chiaro il lavoro svolto prima di ogni consegna e rende efficace la tracciabilità dei file.

### 3.2.3.5 Tipi di file e .gitignore

I file utilizzati per la documentazione del progetto sono:

- file con estensione .tex di L<sup>A</sup>T<sub>E</sub>X, contenenti il codice sorgente del file;
- file con estensione .pdf che sono oggetto di consegna;
- file testuali e immagini impiegati nei file L<sup>A</sup>T<sub>E</sub>X e .pdf.

Il file ".gitignore" è presente al livello più esterno della repository<sub>G</sub> ed elenca tutti i file da escludere dal versionamento.

### 3.2.3.6 Utilizzo di Git

Il repository<sub>G</sub> di Git è composto da diversi branch, ciò permette di lavorare in modo parallelo tra i membri del team. I passaggi che vengono eseguiti sono i seguenti:

- viene scelto il branch su cui si intende lavorare;
- si esegue il pull dal repository<sub>G</sub> remoto che effettua l'aggiornamento del proprio repository<sub>G</sub> locale;
- si svolge il lavoro assegnato, che varia dalla modifica all'aggiunta di nuovi file;
- si aggiungono i nuovi file o file modificati all'area di staging<sub>G</sub> prima di eseguire il commit;
- si esegue il comando di commit dei file aggiunti, con annesso un messaggio che identifica il lavoro effettuato;
- si esegue il push del commit sul repository<sub>G</sub> remoto.

### 3.2.3.7 Gestione delle modifiche

Per effettuare semplici modifiche quali errori grammaticali o migliorie di lessico non è necessaria alcuna approvazione dal responsabile del progetto, è necessario però allegare al commit un messaggio chiaro che indica le modifiche effettuate. Per quanto riguarda modifiche maggiori la procedura da seguire è la seguente:

- contattare il responsabile del file da modificare;
- suggerire la modifica da effettuare;
- se il responsabile valuta positivamente la modifica, allora la applica o dà il permesso di modifica.

## 3.3 Verifica

Lo scopo del processo di verifica è la realizzazione di prodotti corretti, coesi e completi. La verifica viene effettuata sia sul software che sui documenti.

### 3.3.1 Aspettative

Il processo di verifica rispetta i punti seguenti:

- effettuato seguendo procedure definite;
- criteri chiari e affidabili da seguire per verificare il prodotto;
- ogni fase successiva attraversata dal prodotto è verificata;
- la verifica porta il prodotto in uno stato stabile;
- rende possibile la validazione.

### 3.3.2 Descrizione

Il processo di verifica prende in input ciò che è già stato prodotto e lo restituisce in uno stato conforme alle aspettative. Per ottenere tale risultato ci si affida a processi di analisi e test.

### 3.3.3 Attività

#### 3.3.3.1 Analisi

Il processo di analisi consiste nell'analisi del codice sorgente e nella sua successiva esecuzione. Si suddivide in statica e dinamica.

#### Analisi statica

L'analisi statica effettua controlli su documenti e codice, di cui valuta e applica la correttezza (intesa come assenza di errori e difetti), la conformità a regole e la coesione<sub>G</sub> dei componenti. Per effettuare analisi statica esistono metodi manuali di lettura (attuati da persone) e metodi formali (attuati da macchine). Quelli manuali sono due:

- **Walkthrough:** i vari componenti del team analizzano gli oggetti nella loro totalità per cercare anomalie, senza sapere inizialmente se vi siano difetti, quali e dove siano;
- **Inspection:** i verificatori usano liste di controllo per fare ispezione cercando errori specifici in parti specifiche.

tipici errori all'interno dei documenti

- **Formato data:** Si utilizza il formato americano YYYY-MM-DD;
- **Forme verbali:** Si deve utilizzare il più possibile l'indicativo, cercando di rispettare le concordanze verbali;
- **Punteggiatura degli elenchi:** Ogni voce dell'elenco termina con ";" fatta eccezione per l'ultima, la quale termina con ".";
- **Sintassi:** Le frasi devono essere semplici e poco articolate per facilitarne la comprensione;
- **Parti mancanti:** Controllare che non ci siano errori di formattazione o sezioni e titoli vuoti.

#### Analisi dinamica

Il processo di analisi dinamica verrà applicato solamente al codice del software prodotto e consisterà nella creazione ed esecuzione di una serie di test sullo stesso.

#### 3.3.3.2 Test

Per ogni test devono essere definiti i seguenti parametri:

- **ambiente:** sistema hardware e software sul quale viene eseguito il test;
- **stato iniziale:** lo stato iniziale dal quale viene eseguito il test;
- **input:** input inserito;
- **output:** output atteso;
- **istruzioni aggiuntive:** ulteriori istruzioni su come e dove eseguire il test e su come interpretare i risultati ottenuti.

Test ben scritti devono:

- essere ripetibili;
- specificare l'ambiente di esecuzione;
- identificare input e output richiesti;
- avvertire di possibili effetti indesiderati;
- fornire informazioni sui risultati dell'esecuzione in forma di file di log.

Ci sono vari tipi di test del software, ognuno dei quali ha un diverso oggetto di verifica e scopo.

- **Test di unità**

I test di unità si eseguono su unità di software. Questi test si concentrano sul funzionamento delle unità individuali. Dati gli input possibili in un unità, si suddividono gli input in partizioni di equivalenza e si prova se gli input attesi danno gli output previsti. Le singole unità possono essere testate con l'ausilio di  $\text{driver}_G$  e  $\text{stub}_G$  ;

- **Test di integrazione**

Dopo aver superato i test di unità, le stesse unità vengono assemblate in agglomerati progressivamente più grandi. Il test di integrazione si concentra sulle interfacce tra i componenti. Un agglomerato che supera il test di integrazione costituisce quindi una nuova unità per un agglomerato di grandezza maggiore. Questa procedura si ripete fino a raggiungere la dimensione totale del sistema;

- **Test di sistema**

Il sistema viene testato nella sua interezza, una volta integrati tutti i suoi componenti. Ci si concentra sulle interazioni tra le parti, sul comportamento emergente $_G$  delle caratteristiche del sistema e sulla copertura $_G$  di tutte le funzionalità. Questo tipo di test porta alla luce le ipotesi errate che i diversi sviluppatori fanno su parti del software sviluppate da altri. In questa fase ci si assicura che il sistema rispetti tutte le specifiche definite nell'*Analisi dei Requisiti*;

- **Test di regressione**

Si effettua di seguito ad una modifica del sistema e consiste nella riesecuzione dei test esistenti: si combina bene con l'automazione dei test;

- **Test di accettazione**

Anche detto "test di collaudo", è simile al test di sistema per l'oggetto testato, ma viene eseguito con la collaborazione dei committenti. Si occupa di verificare il prodotto e, in particolare, il soddisfacimento del cliente. Il superamento del test di collaudo garantisce che il software sia pronto per essere rilasciato.

### 3.3.4 Strumenti

#### 3.3.4.1 Verifica ortografica

Per la verifica ortografica vengono utilizzati i tool dell'editor i quali sottolineano in rosso gli errori secondo il vocabolario della lingua italiana, facendo attenzione ad eventuali termini in inglese.



## **3.4 Validazione**

### **3.4.1 Scopo**

Lo scopo della validazione è stabilire se il prodotto soddisfa il compito per cui è stato creato. Dopo la validazione, è garantito che il software rispetti i requisiti e che soddisfa i bisogni del committente.

### **3.4.2 Attività**

Per validare il prodotto si devono rispettare i seguenti punti:

- identificare gli oggetti da validare;
- identificare una strategia con delle procedure di validazione in cui le procedure di verifica possono essere riutilizzare;
- applicare la strategia;
- valutare che i risultati rispettino le aspettative.

## 4 Processi Organizzativi

### 4.1 Gestione Organizzativa

#### 4.1.1 Scopo

Lo scopo di questo processo è definire le linee guida che sono raccolte nel documento *Piano di Progetto*. In particolare:

- definire un modello di sviluppo<sub>G</sub> comune a tutti i membri del gruppo;
- creare un modello organizzativo volto alla prevenzione e correzione di errori;
- pianificare il lavoro in base alle scadenze;
- calcolare il piano economico in base al ruolo coperto;
- effettuare il bilancio finale sulle spese.

#### 4.1.2 Aspettative

Le aspettative del processo sono:

- ottenere una pianificazione efficace delle attività da svolgere;
- suddividere per ruoli i membri del gruppo così da poter ricoprire tutte le attività;
- garantire un controllo diretto su ogni parte del progetto.

#### 4.1.3 Descrizione

Viene trattata la gestione dei seguenti argomenti:

- ruoli di progetto;
- gestione delle comunicazioni;
- pianificazione degli incontri;
- gestione degli strumenti di coordinamento;
- gestione dei rischi;

#### 4.1.4 Ruoli di progetto

Ogni membro del gruppo ricopre un ruolo che viene assegnato a rotazione, le attività svolte da ogni ruolo sono definite in modo chiaro nel documento *Piano di Progetto*. I ruoli che ogni componente è tenuto a rappresentare sono descritti in generale di seguito.

#### 4.1.4.1 Responsabile di progetto

Il responsabile di progetto è incaricato di gestire le comunicazioni, fa da referente sia per il committente<sub>G</sub> che per il fornitore.

Il responsabile si occupa di:

- pianificazione delle attività di progetto;
- gestione e coordinamento tra membri del team;
- studio ed analisi dei rischi;
- approvare la documentazione.

#### 4.1.4.2 Amministratore di progetto

L'amministratore coordina l'ambiente di lavoro, assumendosi la responsabilità di gestire la capacità operativa.

Egli si fa carico dei seguenti aspetti:

- amministra i servizi di supporto, come documentazione e strumenti;
- risolvere problemi legati alla gestione dei processi;
- effettua controlli volti alla correzione, verifica, aggiornamento e approvazione della documentazione;
- controlla versionamento e configurazione dei prodotti.

#### 4.1.4.3 Analista

L'analista è la figura incaricata di studiare il problema indicato nel modo più approfondito possibile, così da poter fornire eventuali strumenti e metodologie per affrontarlo. Partecipa per un periodo limitato di tempo, è di grande importanza durante la stesura del documento *Analisi dei requisiti*.

Le sue responsabilità sono:

- studio del dominio del problema e della sua complessità;
- analisi delle richieste implicite ed esplicite;
- stesura dei documenti: *Analisi dei Requisiti* e *Studio di Fattibilità*.

#### 4.1.4.4 Progettista

Il progettista ha il compito di trovare una soluzione ai problemi rilevati dall'analista, fornendo aspetti tecnici e tecnologici coerenti.

Il progettista deve:

- applicare soluzioni note ed ottime.
- operare scelte che portino ad una soluzione efficiente rispetto ai requisiti, considerando costi e risorse;
- sviluppare l'architettura seguendo un insieme di best practice per ottenere un progetto solido e facilmente mantenibile.

#### 4.1.4.5 Programmatore

Il programmatore è la figura responsabile delle attività di codifica e delle componenti necessarie per effettuare le prove di verifica. Il programmatore si occupa di:

- implementare le decisioni del Progettista;
- scrivere codice che rispetti le metriche predefinite, sia versionato e documentato;
- creare e gestire componenti di supporto per la verifica e validazione del codice.

#### 4.1.4.6 Verificatore

Il verificatore ha il compito di supervisionare il prodotto del lavoro degli altri membri del team, sia esso codice o documentazione. Segue delle linee guida volte al controllo e correzione di errori presenti nel documento *Norme di progetto*, nonché alla propria esperienza e capacità di giudizio. Il verificatore deve:

- controllare la conformità del prodotto in ogni suo stadio di vita;
- segnalare al Responsabile di progetto eventuali problemi causati dalla violazione del documento *Norme di progetto*.
- segnalare errori meno importanti all'autore dell'oggetto in questione per un eventuale correzione.

#### 4.1.5 Procedure

Insieme di linee guida che i membri del gruppo seguiranno durante lo sviluppo del progetto.

##### 4.1.5.1 Gestione delle comunicazioni

###### Comunicazioni interne

Le comunicazioni interne sono gestite utilizzando il sistema di messaggistica multi piattaforma Slack<sub>G</sub>. Questo servizio implementa la suddivisione in canali e permette quindi la comunicazione tra membri interessati alle singole attività. Ogni membro ha accesso a tutti i canali e partecipa alle conversazioni che sono di suo diretto interesse. I canali presenti sono variabili, ad eccezione di alcuni che saranno presenti dall'inizio alla fine del progetto. I canali utilizzati sono: b

- **general:** vengono discusse tutte le informazioni off-topic<sub>G</sub>, include la gestione generale del progetto;
- **git:** vengono comunicate le pull eseguite al termine o in esecuzione di un qualsiasi file;
- **documento "...":** più canali nominati in base al nome del documento in fase di redazione e non ancora approvato. Consiste in un insieme di canali in continua variazione in quanto vengono chiusi dopo l'approvazione del documento relativo;
- **spam:** utilizzato per discutere liberamente di ciò che è esterno al progetto.

## Comunicazioni esterne

Le comunicazioni con soggetti esterni al gruppo, quali committente e proponente, sono di competenza del responsabile. Gli strumenti predefiniti sono la posta elettronica, utilizzando l'indirizzo di posta elettronica del gruppo, di cui tutti i membri hanno le credenziali di accesso [zeuscode17@gmail.com](mailto:zeuscode17@gmail.com). Per le comunicazioni con *Gaiago* si utilizza il servizio Google Meet per le riunioni. In caso di assenza di uno o più membri del gruppo un membro presente, a turno, assume l'onere di creare un riassunto scritto per gli altri membri.

### 4.1.5.2 Gestione degli incontri

#### Incontri interni del team

Le riunioni interne del team sono organizzate dal responsabile in accordo con i membri del gruppo. Viene usato Google Calendar se si deve effettuare un incontro importante ed è necessaria la presenza di tutti i membri, in questo modo ogni membro comunica i giorni liberi e si decide. In caso di incontri di minore importanza si utilizza Telegram per organizzarsi con gli orari

#### Verbali di riunioni interne

Al termine di ogni riunione viene nominato un segretario incaricato di produrre il *Verbale* nel quale vengono riportate le decisioni prese e le varie idee.

#### Incontri esterni del team

Gli incontri esterni sono gestiti dal responsabile, il quale ha il compito di organizzare le date e le modalità di incontro relazionandosi con le terze parti. Vengono usati i canali sopra citati per prendere accordi.

#### Verbali di riunioni esterne

Anche per le riunioni esterne viene redatto un *Verbale*, il quale presenta una struttura analoga al verbale redatto per le riunioni interne.

### 4.1.5.3 Gestione degli strumenti di coordinamento

#### Tickeeting

Strumento utilizzato per la suddivisione dei compiti all'interno del team. Permette al responsabile di progetto di monitorare le attività in corso e di assegnare le risorse disponibili ad una o più *schedule<sub>G</sub>*.

Utile e versatile in quanto presenta anche una controparte mobile accessibile in qualunque momento. Ad ogni task sono associati una data di scadenza e un insieme di membri assegnatari. Ogni compito passa attraverso i seguenti stati:

- da fare;
- in lavorazione;

- in revisione;
- completato.

Il gruppo ha deciso di usare Trello a discapito di altri tool vista la sua semplicità di utilizzo ed apprendimento.

#### 4.1.5.4 Gestione dei rischi

La rilevazione dei rischi è compito del responsabile di progetto, questa attività è documentata nel *Piano di Progetto*. La procedura da seguire per la gestione dei rischi è la seguente:

- individuare possibili problemi e monitorare i rischi in modo preventivo;
- tenere traccia dei singoli rischi con eventuali contromisure nel *Piano di Progetto*;
- ridefinire e implementare le strategie di gestione dei rischi.

#### Codifica dei rischi

Le tipologie di rischi sono così codificate:

- **RT**: Rischi Tecnologici;
- **RO**: Rischi Organizzativi;
- **RI**: Rischi Interpersonali.

#### 4.1.6 Strumenti

Il gruppo, nel corso del progetto, ha utilizzato o utilizzerà i seguenti strumenti:

- **Telegram<sub>G</sub>**: strumento di messaggistica utilizzato inizialmente per la gestione del gruppo;
- **Slack<sub>G</sub>**: per la comunicazione interna del team, composta anche da chiamate vocali;
- **Trello**: per suddividere le varie task tra i membri;
- **Git**: sistema di controllo di versionamento;
- **GitHub Desktop**: applicazione desktop che aiuta la gestione del versionamento;
- **GitHub**: per il versionamento e il salvataggio in repository<sub>G</sub> remota dei file;
- **Mega**: utilizzato per contenere i file che hanno un basso reteo di modifica e un alto rateo di visualizzazione;
- **Google Calendar**: utile per pianificare gli incontri soprattutto in caso di comunicazioni esterne;
- **Google Meet**: servizio che offre la possibilità di fare videoconferenze e chiamate VoIP, utilizzato per parlare con il proponente e per alcuni incontri interni;