Obelix Group
obelixswe@gmail.com

Norme di Progetto

Versione	*v1_1_0*
Data creazione	2017/02/20
Redattori	Emanuele Crespan
	Tomas Mali
Verificatori	Silvio Meneguzzo
	Riccardo Saggese
Approvazione	Federica Schifano
Stato	Approvato
Uso	interno

Sommario

Documento contenente le norme di progetto che il gruppo Obelix seguirà durante tutte le fasi di realizzazione del prodotto Monolith.

Diario delle revisioni

Modifica	Autore e Ruolo	Data	Versione
Verifica sottosezioni Riferimenti e Sviluppo	Silvio Meneguzzo Verificatore	2017-03-01	1.1.0
Modifica sottosezioni Riferimenti e Sviluppo	Tomas Mali Amministratore	2017-03-01	1.0.1
Approvazione del documento	Federica Schifano Responsabile	2017-02-24	1.0.0
Verifica del documento	Riccardo Saggese Verificatore	2017-02-24	0.1.0
Stesura sezione Processi Organizzativi	Emanuele Crespan Amministratore	2017-02-23	0.0.4
Stesura sezione Processi di supporto	Tomas Mali Amministratore	2017-02-22	0.0.4
Stesura sezione Processi Primari	Emanuele Crespan Amministratore	2017-02-21	0.0.3
Stesura sezione Introduzione	Tomas Mali Amministratore	2017-02-20	0.0.2
Creato template	Tomas Mali Amministratore	2017-02-20	0.0.1

Indice

1 Introduzione

1.1 Scopo del documento

Il documento definisce le norme che i membri del gruppo Obelix dovranno seguire nello svolgimento del progetto Monolith. Tutti i membri del gruppo devono prendere visione del documento. É necessario adottare le norme in esso contenute per ottenere massima coerenza ed efficienza durante lo svolgimento delle attività.

1.2 Scopo del prodotto

Lo scopo del prodotto è quello di permettere la creazione di bolle interattive,che dovranno funzionare nell'ambiente Rocket.chat. Queste bolle permetteranno di aumentare l'interattività tra gli utenti della chat, aggiungendo nuove funzionalità che saranno accessibili direttamente dalla conversazione senza il bisogno di ricorrere all'apertura di applicazioni diverse. Il sistema offrirà agli sviluppatori un set di $API_{|G|}$ per creare e rilasciare nuove bolle e agli utenti finali la possibilità di usufruire di un insieme di bolle predefinite.

1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensibilità dei documenti. I termini che necessitano di essere chiariti saranno marcati con una |G| in pedice alla prima occorrenza e saranno riportati nel Glossario.

1.4 Riferimenti

1.4.1 Informativi

- **Javascript 6th edition**:
  `https://exploringjs.com/es6/ch_promises.html`
- **12 Factors app guidelines**:
  `https://12factor.net`
- **SCSS**:
  `http://sass-lang.com`
- **Piano di Progetto**:
  Piano di Progetto v1.1.0
- **Piano di Qualifica**:
  Piano di Qualifica v1.0.0

1.4.2 Normativi

- **The Airbnb Javascript style guide**:
  `https://github.com/airbnb/javascript`
- **ESLint**:
  `https://github.com/eslint/eslint`

2 Processi primari

2.1 Fornitura

2.1.1 Studio di fattibilità

Il documento inerente allo studio di fattibilità dovrà essere stilato dagli *Analisti* che effettueranno un resoconto delle prime riunioni tra i membri del gruppo. Saranno sviluppati i seguenti punti relativi al capitolato scelto:

- Scopo del progetto
- Dominio applicativo
- Dominio tecnologico
- Criticità e rischi durante lo sviluppo del prodotto

Infine saranno presentate le motivazioni che hanno condotto il gruppo nella scelta del capitolato.

2.2 Sviluppo

2.2.1 Analisi dei requisiti

Gli *Analisti* si occuperanno,inoltre, della stesura dell'analisi dei requisiti. Saranno analizzati e classificati i requisiti che il prodotto finale deve soddisfare.

Classificazione dei casi d'uso Gli *Analisti* dovranno analizzare i casi d'uso (UC - Use Case) procedendo dal generale al particolare. Ogni caso d'uso sarà identificato nel seguente modo:

UC[codice univoco del padre].[codice progressivo di livello]$-$[sigla identificativa]

Il codice progressivo può includere diversi livelli di gerarchia separati da un punto. L'uso della sigla identificativa è opzionale.

Per ogni caso d'uso saranno specificati:

- Titolo sintetico e significativo
- Attori principali
- Attori secondari, se presenti
- Precondizione
- Scenario principale e relativo flusso degli eventi
- Scenari secondari,se presenti
- Postcondizione
- Generalizzazioni, se presenti
- Inclusioni, se presenti
- Estensioni, se presenti

Classificazione dei requisiti Dopo aver stilato gli use case gli *Analisti* determineranno i requisiti che saranno organizzati per utilità strategica:

- Ob: Obbligatori
- De: Desiderabili
- Op: Opzionali

Ogni requisito dovrà essere distinto,inoltre ,in base al tipo e quindi al modo in cui dovrà essere verificato:

- Fu: Funzionale
- Qu: Qualitativo
- Di: Dichiarativo

Ogni requisito dovrà rispettare la seguente codifica:

$$R[\text{utilità strategica}][\text{tipo}][\text{codice}]$$

Il codice univoco dovrà essere espresso in forma gerarchica.

Per ogni requisito saranno inseriti una descrizione, il più possibile chiara e priva di ambiguità, e la fonte da cui è emerso:

- Capitolato
- Incontri/Conferenze con il proponente
- Discussioni interne al gruppo
- Casi d'uso

2.2.2 Progettazione

L'attività di progettazione consiste nel tracciamento delle linee guida che dovranno essere seguite per la codifica. Alla fine di questa attività sarà, dunque, definita l'architettura del sistema da realizzare. L'architettura sarà suddivisa in componenti distinte che dipendono il meno possibile le une dalle altre. Ogni componente dovrà soddisfare almeno un requisito e tutti i requisiti dovranno essere soddisfatti.

I documenti derivanti da questa attività sono: Specifica Tecnica e Definizione di Prodotto.

Specifica Tecnica I *Progettisti* devono descrivere la progettazione ad alto livello del sistema e dei singoli componenti nella Specifica Tecnica. Devono essere definiti, inoltre, opportuni test di integrazione tra le varie componenti.

I contenuti da coprire sono i seguenti:

- **Diagrammi UML**:
  - Diagrammi dei package
  - Diagrammi delle classi
  - Diagrammi di sequenza

  - Diagrammi di attività

- **Design Pattern**:

  Devono essere descritti i design patterns utilizzati per realizzare l'architettura. Si deve includere una breve descrizione e un diagramma che esponga il significato e la struttura di ciascuno.

- **Tracciamento dei componenti**: Ogni requisito deve essere tracciato al componente che lo soddisfa. In questo modo è possibile verificare che vengano soddisfatti tutti i requisiti e che ogni componente soddisfi almeno un requisito.

- **Test di integrazione**:

  Devono essere definite delle strategie di verifica per poter dimostrare la corretta integrazione tra le varie componenti.

Definizione di Prodotto I *Progettisti* produrranno la Definizione di Prodotto integrando quanto scritto nella Specifica Tecnica. Sarà descritta la progettazione di dettaglio del sistema: ogni singola unità di cui è composto il sistema sarà definita in modo dettagliato.

I contenuti da coprire sono i seguenti:

- **Diagrammi UML**:

  - Diagrammi delle classi
  - Diagrammi di sequenza
  - Diagrammi di attività

- **Definizioni delle classi**:

  Ogni classe progettata deve essere descritta in modo da spiegarne lo scopo e definirne le funzionalità ad essa associate. Per ogni classe dovranno anche essere definiti i vari metodi ed attributi che la caratterizzano.

- **Tracciamento delle classi**:

  Ogni classe deve essere tracciata ed associata ad almeno un requisito, in questo modo è possibile avere la certezza che tutti i requisiti siano soddisfatti e che ogni classe soddisfi almeno un requisito.

- **Test di unità**:

  Dovranno essere definiti dei test d'unità per poter verificare che i componenti del sistema funzionino in modo corretto.

2.2.3 Codifica

Lo scopo dell'attività di codifica è quello di realizzare il prodotto software seguendo le linee guida delineate nell'attività di progettazione. É necessario che tutti i *Programmatori* utilizzino lo stesso stile di codifica per garantire la creazione di codice uniforme. Le norme stilistiche riguardano:

- Formattazione

- Intestazione
- Nomi di elementi
- Commenti
- Stile di codifica

Formattazione

- Indentazione: è richiesto l'uso di esattamente quattro spazi
- É necessario rientrare le righe di codice secondo la relativa costruzione logica
- É richiesto di inserire le parentesi di delimitazione dei costrutti di controllo in linea
- É richiesto di inserire le parentesi di delimitazione dei costrutti di dichiarazione di funzioni o classi al di sotto di essi e non in linea

Intestazione L'intestazione di ogni file deve contenere obbligatoriamente le seguenti informazioni:

- Name: nome del file
- Location: percorso di locazione del file
- Author: autore del file
- Creation data: data di creazione del file
- Description: descrizione del file
- History:
  - Version: codice univoco indicante la versione del file
  - Update data: data ultima di modifica
  - Description: descrizione della modifica
  - Author: autore della modifica

```
/*
* Name : { Nome del file }
* Location : {/ path / della / cartella /}
* Author: {Autore del file}
* Creation Data: {Data di creazione del file}
* Description: {Breve descrizione del file}
* History :
*     Version: {Versione del file}
*     Update data: {Data ultima modifica}
*     Description: {descrizione della modifica}
*     Author: {Autore della modifica}
*/
```

Nomi

- Assegnare ad ogni elemento un nome univoco significativo
- I nomi di variabili, metodi e funzioni devono avere la prima lettera minuscola e le successive iniziali delle parole, che ne compongono il nome, in maiuscolo (la cosiddetta notazione a cammello)
- I nomi delle classi devono avere la prima lettera maiuscola

Commenti L'inserimento di commenti all'interno del codice ne agevola la comprensione ed è incoraggiato. É opportuno inserire i commenti per chiarire il comportamento di funzioni e metodi o per precisare tutto ciò che può essere ambiguo nel codice.

Stile di codifica Vanno osservate le regole stilistiche descritte nei riferimenti sopra indicati.

2.2.4 Strumenti usati

Tracciabilità La Tracciabilità dei requisiti e dei casi d'uso avviene tramite l'utilizzo di un database MySQL

Astah Astah è l'editor scelto dal gruppo per la modellazione nel linguaggio UML. *Astah*$_{|G|}$ è un editor gratuito, scaricabile online sia per Microsoft Windows, sia per *Linux*$_{|G|}$ che per Mac OS. É un editor abbastanza semplice e user-friendly. Infatti ad un primo utilizzo, è facile capire come muoversi nell'ambiente di sviluppo, come aggiungere classi, diagrammi e relazioni. Strumento molto utile che l'editor mette a disposizione è quello di poter esportare il diagramma realizzato in un file immagine. Questo software verrà utilizzato nelle attività di analisi e progettazione per la creazione di tutti i diagrammi *UML*$_{|G|}$ necessari.

Microsoft Project Microsoft Project è un software di pianificazione sviluppato e venduto da Microsoft. É uno strumento 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. Questo software verrà utilizzato per la creazione di tutti i diagrammi di Gantt necessari.

3 Processi di supporto

3.1 Processo di documentazione

3.1.1 Descrizione

Lo scopo del processo di documentazione è quello di illustrare le convenzioni che riguardano la stesura, la verifica e l'approvazione dei documenti. La classificazione di tali documenti è come segue:

- Interni: utilizzo interno al team

- Esterni: distribuzione esterna al gruppo, per il *committente*$_{|G|}$ e/o per il proponente

3.1.2 Strumenti

Come linguaggio di markup per la stesura della documentazione è stato scelto LaTeX. Questa scelta è stata presa per evitare possibili incompatibilità che si possono presentare utilizzando software differenti.

3.1.3 Ciclo di vita di un documento

I documenti si possono trovare in uno degli stati seguenti:

- Documento in lavorazione
- Documento da verificare
- Documenti approvati

I documenti in lavorazione sono i documenti che si trovano in stesura da parte del redattore. Una volta terminata la loro realizzazione, questi documenti vanno segnati come da verificare. Solo dopo la verifica da parte del relativo *Verificatore* i documenti passano nello stato "approvato". In questo stato i documenti sono stati consegnati al *Responsabile* del Progetto che ha il compito di approvarli in via definitiva.

3.1.4 Documenti finali

Studio di Fattibilità Gli *Analisti* analizzano tutti i capitolati valutandone i fattori positivi e negativi. É un'attività critica perché porterà alla scelta del progetto sul quale il gruppo andrà a lavorare. Questo documento è utilizzato internamente al *team*$_{|G|}$ e la lista di distribuzione comprende solo il committente.

Norme di Progetto In questo documento vengono riportati gli strumenti, le norme e le convenzioni che il team adotterà nello sviluppo del progetto. Questo documento è utilizzato internamente al team e la lista di distribuzione comprende solo il committente.

Piano di Progetto In questo documento vengono descritte le strategie che il team applicherà per la gestione delle risorse umane e temporali. La lista di distribuzione di questo documento comprende il committente ed il proponente.

Piano di Qualifica Lo scopo di questo documento è quello di illustrare un piano con il quale il team punta a soddisfare gli obiettivi di qualità. La lista di distribuzione di questo documento comprende il committente ed il proponente.

Analisi dei Requisiti In questo documento viene descritta la raccolta dei requisiti e dei casi d'uso nel modo più dettagliato possibile. Questo documento conterrà dunque tutti i casi d'uso ed i diagrammi di attività che rappresentano l'interazione tra utente e sistema. Questo documento è utilizzato sia esternamente che internamente.

Specifica Tecnica I *Progettisti* devono descrivere la progettazione ad alto livello del sistema e dei singoli componenti nella Specifica Tecnica. Devono essere definiti, inoltre, opportuni test di integrazione tra le varie componenti.

Definizione di Prodotto Lo scopo di questo documento è di descrivere i dettagli implementativi del prodotto. Queste informazioni saranno la base per il processo di codifica.

Glossario In questo documento verranno elencati tutti i termini che necessitano disambiguazione provenienti da tutta la documentazione. Ciascuno di questi termini sarà accompagnato dalla spiegazione del significato. Questo documento è utilizzato anche esternamente e la lista di distribuzione comprende committenti e proponenti.

Manuale Utente Con questo documento verrà fornito all'utente una guida dettagliata coprendo tutte le funzionalità che il prodotto offre. Su richiesta del *proponente*$_{|G|}$ verrà stilato in lingua inglese.

Verbali In questo documento vengono descritte formalmente tutte le discussioni e le decisioni prese durante le riunioni. I verbali possono essere interni oppure esterni. Quelli esterni in particolare dovranno essere redatti dal *Responsabile* di Progetto. Ogni verbale dovrà essere denominato nel seguente modo :

Verbale_Numero del verbale_Tipo di verbale_Data del verbale

dove:

- Numero del verbale: numero identificativo univoco del verbale
- Tipo di verbale: identifica se si tratta di un Verbale Interno oppure Verbale Esterno
- Data del verbale: identifica la data nella quale si è svolta la riunione corrispondente al verbale. Il formato è YYYY-MM-DD

Nella parte introduttiva del verbale dovranno essere specificate le seguenti informazioni:

- Data incontro: data in cui si è svolta la riunione
- Ora inizio incontro: ora di inizio della riunione
- Ora termine incontro: ora di terminazione della riunione
- Luogo incontro: luogo in cui si è svolta la riunione
- Durata: durata della riunione
- Oggetto: argomento della riunione
- Segretario: cognome e nome del membro incaricato a redigere il verbale
- Partecipanti: cognome e nome di tutti i membri partecipanti alla riunione

3.1.5 Struttura del documento

Prima pagina Tutti i documenti dovranno contenere nella prima pagina le seguenti informazioni:

- Nome del gruppo
- Logo del progetto
- Nome del progetto
- Nome del documento
- Versione del documento
- Data di creazione del documento
- Stato del documento
- Nome e cognome del redattore del documento
- Nome e cognome del *Verificatore* del documento
- Nome e cognome del *responsabile* approvatore del documento
- Uso del documento
- Email di riferimento del gruppo
- Un sommario contenente una breve descrizione del documento

Diario delle modifiche Il diario delle modifiche sarà presente nella seconda pagina dove verranno inserite tutte le modifiche del documento. Tutte queste informazioni verranno riportate in una tabella che in ogni riga conterrà le seguenti informazioni:

- Versione: versione del documento dopo la modifica
- Descrizione: descrizione della modifica
- Autore e Ruolo: autore della modifica e il suo ruolo
- Data: data della modifica

Indice In ogni documento, dopo il diario delle modifiche, deve essere presente un indice di tutte le sezioni. In presenza di tabelle e/o immagini queste devono essere indicate con i relativi indici.

Formattazione generale delle pagine La formattazione della pagina, oltre al contenuto, prevede un'intestazione e un piè di pagina. L'intestazione della pagina contiene:

- Nome e logo del gruppo
- Nome della sezione corrente

Il piè di pagina contiene:

- Il numero di pagina e il numero totale di pagina

3.1.6 Norme tipografiche

Le seguenti norme tipografiche indicano i criteri riguardanti l'ortografia e la tipografia di tutti i documenti.

Stili di testo

- Corsivo: il testo corsivo verrà utilizzato nelle seguenti situazioni:
  - Ruoli: ogni riferimento a ruoli di progetto va scritto in corsivo
  - Documenti: ogni riferimento ad un documento
  - Glossario: ogni parola presente nel glossario deve essere scritta in corsivo. Inoltre presentano una |G| a pedice
- Font codice: il font per il codice ("font macchina da scrivere") deve essere utilizzato per indicare qualsiasi parte riguardi posizione del codice.

Punteggiatura

- Punteggiatura: ogni simbolo di punteggiatura non può seguire un carattere di spazio
- Lettere maiuscole: le lettere maiuscole vanno utilizzate dopo il punto, il punto interrogativo, il punto esclamativo.

Formati

- Date: le date presenti nei documenti devono seguire lo standard **ISO 8601:2004: YYYY-MM-DD**
  dove:
  - YYYY: rappresenta l'anno
  - MM: rappresenta il mese
  - DD: rappresenta il giorno
- Ore: le ore presenti nei documenti devono seguire lo standard **ISO 8601:2004** con il sistema a 24 ore:
  dove:
  - hh: rappresentano le ore
  - mm: rappresentano i minuti

3.1.7 Composizione email

In questo paragrafo verranno descritte le norme da applicare nella composizione delle email.

Destinatario

- Interno: l'indirizzo da utilizzare è obelixswe@gmail.com
- Esterno: l'indirizzo del destinatario varia a seconda si tratti del Prof. Tullio Vardanega Prof. Riccardo Cardin o i proponenti del progetto

Mittente

- Interno: l'indirizzo è di colui che scrive e spedisce la email
- Esterno: l'indirizzo da utilizzare è obelixswe@gmail.com ed è utilizzabile unicamente dal *Responsabile* di Progetto

Oggetto L'oggetto della mail deve essere chiaro, preciso e conciso in modo da rendere semplice il riconoscimento di una mail tra le altre.

3.1.8 Componenti grafiche

Tabelle Tutte le tabelle in tutti i documenti devono avere una didascalia ed un indice identificativo univoco per il loro tracciamento nel documento stesso.

Immagini Le immagini inserite nel documento sono nel formato PNG. Le immagini hanno una didascalia e compaiono nell'indice dedicato.

3.1.9 Versionamento

Ogni documento prodotto deve essere identificato dal nome e dal numero di versione nel seguente modo:

$$\_vX\_Y\_Z$$

dove:

- X: indica il numero di uscite formali del documento e viene incrementato in seguito all'approvazione finale da parte del *Responsabile* di Progetto. L'incremento dell'indice X comporta l'azzeramento degli indici Y e Z
- Y: indica il numero crescente delle verifiche. L'incremento viene eseguito dal *Verificatore* e comporta l'azzeramento dell'indice Z
- Z: indica il numero di modifiche minori apportate al documento prima della sua verifica. Viene aumentato in modo incrementale

3.2 Processo di verifica

3.2.1 Descrizione

Lo scopo del Processo di verifica è quello di verificare che ogni documento prodotto rispecchi quanto previsto dai requisiti.

3.2.2 Analisi

Analisi statica L'analisi statica è il processo di valutazione di un sistema senza che esso venga eseguito. Ci sono diverse tecniche di analisi statica. Le due più applicate sono Walkthrough e Inspection.

- Walkthrough: riguarda un analisi informale del codice svolta da vari partecipanti umani che 'operano come il computer': in pratica, si scelgono alcuni casi di test e si simula l'esecuzione del codice a mano (si attraversa (walkthrough) il codice). Si presta attenzione sulla ricerca dei difetti, piuttosto che sulla correzione.

- Inspection: questa tecnica di ispezione di codice può rilevare ed eliminare anomalie fastidiose e rendere più precisi i risultati. Inspection è una tecnica completamente manuale per trovare e correggere errori. Spesso l'analisi dei difetti effettuata viene discussa e vengono decise le eventuali azioni da intraprendere. Il codice è analizzato usando checklist dei tipici errori di programmazione.

Analisi dinamica L'analisi dinamica consiste nell'esaminare il software quando è in esecuzione ed è tipicamente effettuata dopo aver compiuto l'analisi statica di base.

3.2.3 Test

Test di unità Lo scopo del test di unità è quello di verificare che tutte le componenti del software funzionino correttamente. Questo test aiuta a ridurre il più possibile la presenza di errori nelle componenti. I test di unità sono identificati dalla seguente sintassi:

$$TU[\text{Codice Test}]$$

Test di Integrazione Questo test permette di verificare che ciascuna componente che è stata validata singolarmente, funzioni correttamente anche dopo averla integrata con l'intero sistema. I test di integrazione sono identificati dalla seguente sintassi:

$$TI[\text{Codice Test}]$$

Test di sistema I test di sistema hanno lo scopo di verificare che tutti i requisiti siano soddisfatti. Questo test viene effettuato quando il prodotto è giunto alla versione definitiva. I test di sistema sono identificati dalla seguente sintassi:

$$TR[\text{Codice Requisito}]$$

Test di regressione Il test di regressione consiste nell'effettuare i test per tutte le componenti che sono state modificate. I test di regressione sono identificati dalla seguente sintassi:

$$TR[\text{Codice Test}]$$

3.3 Processo di validazione

3.3.1 Test di validazione

Il test di validazione coincide con il collaudo del software in presenza del proponente. In caso di esito positivo si procede con il rilascio del software. I test di validazione sono identificati dalla seguente sintassi:

$$TV[\text{Codice requisito}]$$

3.4 Strumenti

3.4.1 Strumenti per l'analisi statica

- JSHint: è uno strumento che aiuta a rilevare errori possibili nel codice JavaScript. Verrà installato come modulo per Node.js
- CSSHint: è uno strumento che aiuta a rilevare possibili errori nel codice CSS. Verrà installato come modulo per Node.js
- Complexity-report: è un'applicazione che verrà installata come modulo per *Node.js*$_{|G|}$ e serve a misurare metriche riguardanti codice JavaScript

3.4.2 Strumenti per l'analisi dinamica

- Google Chrome DevTools: gli strumenti offerti da Google Chrome agli sviluppatori servono a tenere sotto controllo l'utilizzo di CPU e di memoria da parte degli oggetti e funzioni JavaScript
- Karma: è uno strumento per eseguire test di unità e sarà utilizzato per eseguire test sugli script realizzati. Verrà installato come modulo per Node.js.

4 Processi Organizzativi

4.1 Processo di Gestione

4.1.1 Scopo del Processo

Lo scopo di questo processo è quello della produzione del documento Piano di Progetto che serve ai membri del gruppo per la pianificazione e la gestione dei ruoli.

4.1.2 Aspettative del Processo

- Realizzazione del Piano di Progetto
- Definizione ruoli del team
- Definire il piano per l'esecuzione dei compiti programmati

4.1.3 Descrizione

Viene trattata le gestione dei seguenti argomenti:

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

4.1.4 Ruoli di Progetto

Nello sviluppo del progetto, ogni membro del team dovrà ricoprire, a turno, ogni ruolo. Nel Piano di Progetto vengono pianificate le attività da assegnare ai seguenti ruoli:

***Amministratore* di Progetto** L' *Amministratore* di Progetto ha il compito di controllare ed amministrare l'ambiente di lavoro assumendosi le seguenti responsabilità:

- studio e ricerca di strumenti per migliorare l'ambiente lavorativo
- minimizzare il carico di lavoro umano automatizzando dove possibile
- garantire un controllo della qualità del prodotto fornendo procedure e strumenti di segnalazione e monitoraggio
- risolvere i problemi legati alla gestione dei processi e alle risorse disponibili
- gestire il versionamento e l'archiviazione della documentazione di progetto

***Responsabile* di Progetto** Il *Responsabile* di Progetto funge da punto di riferimento per il gruppo, il committente ed il fornitore.
Approva le scelte prese dal gruppo e si assume le seguenti responsabilità:

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

Analista L' *Analista* deve effettuare ricerche e studi approfonditi sul dominio del problema. Non è richiesta la sua presenza per l'intera durata del progetto e si assume le seguenti responsabilità:

- comprensione della natura e della complessità del problema
- produzione dello Studio di Fattibilità e dell'Analisi dei Requisiti delineando specifiche comprensibili ad ogni figura coinvolta (proponente e committente)

Progettista Il *Progettista* deve avere profonde conoscenze delle tecnologie utilizzate e competenze tecniche aggiornate, in modo tale da poter interagire con il committente per arrivare ad un chiarimento progressivo delle varie caratteristiche del sistema finale.
Il *Progettista* si assume le seguenti responsabilità:

- effettuare scelte efficienti ed ottimizzate su aspetti tecnici del progetto
- effettuare scelte che rendano il prodotto più facilmente mantenibile

Verificatore Il *Verificatore* deve avere ampia conoscenza delle normative di progetto.
Il suo compito è controllare che le attività di progetto siano svolte secondo le norme stabilite.

Programmatore Il *Programmatore* è *responsabile* delle attività di codifica e di creazione delle componenti di supporto, utili a effettuare le prove di verifica e validazione sul prodotto software.
Si assume le seguenti responsabilità:

- implementare le soluzioni previste dal *Progettista*
- scrivere codice pulito, mantenibile e conforme alle norme di progetto
- versionare il codice prodotto
- realizzare gli strumenti utili per poter compiere le prove di verifica e validazione

4.2 Gestione Organizzativa

4.2.1 Comunicazione

Comunicazioni interne Per le comunicazioni interne è stato deciso di utilizzare l'applicazione *Slack*$_{|G|}$ dove è stato creato un gruppo di lavoro Obelix all'interno del quale i membri del gruppo possono comunicare attraverso la chat principale.
Per quanto riguarda la comunicazione su argomenti specifici è possibile la creazione di alcune chat interne per mandare messaggi solo ai membri interessati. Nel caso fosse necessaria una video-chiamata verrà utilizzata l'applicazione *Hangout* di Google.

Comunicazioni esterne Per le comunicazioni esterne è stata creata un indirizzo email apposito: obelixswe@gmail.com.
Il Responsabile *del Progetto* avrà l'onere di gestire le comunicazioni esterne riservandosi di condividere le mail che lo richiedono con il resto del team attraverso Slack o una mailing list.

4.2.2 Riunioni

Il Responsabile *di progetto* ha il compito di indire le riunioni sia interne che esterne.

Riunioni interne Per ogni incontro interno il Responsabile *di progetto* dovrà accordarsi con il team inviando un messaggio sull'adeguata chat di Slack almeno 2 giorni prima della data scelta.
Una volta che si sarà accordato coi membri del gruppo potrà creare l'evento dell'incontro sul calendario di *Slack* indicando:

- data
- ora

- luogo
- argomento di discussione

Vi possono essere casi eccezionali, come una milestone, in cui il vincolo sull'avviso (2 giorni prima) può essere ignorato.
Gli altri componenti del gruppo possono richiedere una riunione interna straordinaria, presentando al Responsabile *di progetto* le motivazioni per le quali si ritiene necessaria una riunione. In questi casi il Responsabile *di progetto* può:

- autorizzare lo svolgimento della riunione
- negare lo svolgimento della riunione, nel caso in cui non ritenga le motivazioni presentate valide abbastanza da richiedere una riunione
- suggerire mezzi di comunicazione differenti.

Riunioni esterne Come concordato con RedBabel, le riunioni faccia a faccia programmate sono due:

1. fine Aprile/inizio Maggio
2. Luglio.

Per tutte le altre comunicazioni tra le parti si utilizzeranno Slack e Skype.

4.2.3 Strumenti di coordinamento

Ticketing Il sistema di ticketing permette di assegnare le mansioni ai vari membri del team assegnandone la priorità e permettendo di impostarne lo stato.
Ciò permette al *Responsabile* di tenere sotto controllo l'avanzamento delle attività di progetto in ogni momento. A lui spetta il compito di assegnare i vari ticket ai membri del gruppo seguendo la procedura:

- inserire un titolo riassuntivo del task
- indicare la/e persona/e incaricate a svolgere tale compito
- indicare la data in cui finire il compito assegnato
- inserire una descrizione contenente la specifica del compito
- aggiungere eventuali allegati utili allo svolgimento del task
- inserire eventuali subtask ed assegnarli ai membri del team;

4.2.4 Strumenti di versionamento

Repository Per il versionamento ed il salvataggio dei file prodotti durante l'attività di progetto è stato deciso di affidarsi ad alcune $repository_{|G|}$ su GitHub.
L' Amministratore *di Progetto* si occupa della creazione di questa repository e, successivamente, gli altri membri del gruppo accederanno alla repository tramite il loro account personale.
Sono previste le seguenti repository:

- **docs**: cartella contenente la documentazione di progetto
- **Monolith**: cartella contenente i file di progetto

Struttura Repository I membri del gruppo sono tenuti a rispettare le seguenti norme sull'organizzazione dei file nei repository.
Struttura del repository docs:

- **RR**: contiene i file riguardanti la *Revisione dei Requisiti*;

La struttura del repository Monolith verrà determinata durante la fase di codifica.

Norme sui commit Ogni volta che vengono effettuate delle modifiche ai file del repository bisogna specificarne le motivazioni. Questo avviene utilizzando il comando `git commit` accompagnato da un messaggio riassuntivo e una descrizione in cui va specificato:

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

Prima di eseguire tale procedura, va aggiornato il diario delle modifiche, secondo le regole viste nella sezione 3.1.5.

4.2.5 Gestione dei rischi

Il Responsabile *di Progetto* ha il compito di individuare i rischi indicati nel *Piano di Progetto*. Quando questi vengono individuati, il *Responsabile* ha il compito di estendere l'analisi dei rischi ai nuovi casi rilevati.
Complessivamente la procedura prevede i seguenti passi:

1. rilevazione ed individuazione attiva dei rischi previsti e dei problemi non calcolati
2. registrazione del riscontro effettivo dei rischi nel *Piano di Progetto*
3. trattazione e pianificazione per la gestione dei nuovi rischi individuati
4. ridefinizione e aggiornamento delle strategie in base alle necessità e alle previsioni.

4.2.6 Strumenti

4.2.7 Sistema operativo

Il gruppo di progetto opera sui seguenti sistemi operativi:

- MacOS 10.12
- MacOS 10.9
- Ubuntu x64 16.04 LTS
- Windows 10 Pro
- Linux Mint 17.1
- Windows 7 Pro

4.2.8 Telegram

Telegram è un servizio di messaggistica istantanea erogato senza fini di lucro dalla società *Telegram*$_{|G|}$ LLC. I *client* ufficiali di Telegram sono distribuiti come software libero per diverse piattaforme.
Viene utilizzato dai membri del gruppo per comunicare liberamente sul progetto e su dettagli organizzativi.

4.2.9 Git

Git è un sistema software di controllo di versione distribuito, creato da *Linus Torvalds*$_{|G|}$ nel 2005. La versione utilizzata è maggiore o uguale alla 2.10.1.

4.2.10 GitHub

GitHub è un servizio web di hosting per lo sviluppo di progetti software, che usa il sistema di controllo di versione Git. Può essere utilizzato anche per la condivisione e la modifica di file di testo e documenti revisionabili (sfruttando il sistema di versionamento dei file di Git). *GitHub*$_{|G|}$ ha diversi piani per repository privati sia a pagamento, sia gratuiti, molto utilizzati per lo sviluppo di progetti open-source.

4.2.11 Google Drive

Il sevizio *cloud* based *Google Drive* viene utilizzato per un rapido scambio di documenti e file non soggetti a versionamento che non hanno necessità di risiedere nel *repository*.

4.2.12 Google Calendar

Google Calendar viene utilizzato dai membri del team per segnare le date di importanti scadenze, come *milestone*, incontri o riunioni.

4.2.13 Skype

Skype è un software proprietario freeware di messaggistica istantanea e VoIP. Esso unisce caratteristiche presenti nei client più comuni (chat, salvataggio delle conversazioni, trasferimento di file) ad un sistema di telefonate basato su un network Peer-to-peer.