404NotFound

Premi: better than Prezi.

Norme di Progetto

Informazioni sul documento

Versione	3.0
Redazione	Cossu Mattia Rettore Andrea De Lazzari Enrico Camborata Marco
Verifica	Vegro Federico
Responsabile	Cossu Mattia
Uso	Interno
Ultima modifica	20 Agosto 2015
Lista di distribuzione	404NotFound

Descrizione

Documento contenente le norme stabilite dal gruppo 404NotFound per la realizzazione del progetto Premi

Registro delle modifiche

Versione	Autore	Data	Descrizione
3.0	Cossu Mattia	2015-08-21	approvazione del documento
2.5	Vegro Federico	2015-08-20	Verifica del documento
2.4	Vegro Federico	2015-06-26	aggiunta intestazione del codice nelle Procedure
2.3	Manuto Monica	2015-06-23	Riorganizzato il documento, raggruppati li strumenti nella sezione Ambiente di lavoro
2.2	Gobbo Ismaele	2015-06-03	Modifiche a sezione Procedure (progettazione architetturale, progettazione di dettaglio)
2.1	Manuto Monica	2015-06-01	Incremento sezione Procedure. Aggiunta sezione Gestione di Progetto
2.0	Gobbo Ismaele	2015-05-26	Approvazione documento
1.7	Rettore Andrea	2015-05-12	Verifica finale documento
1.6	Manuto Monica	2015-03-19	Verifica documento
1.5	Cossu Mattia	2015-03-25	Aggiunta sezione relativa alle norme che regolano il documento Specifica Tecnica
1.4	Cossu Mattia	2015-03-23	Correzione errori ortografici e lessicali rilevati dalla verifica precedente
1.3	Rettore Andrea	2015-03-15	Riorganizzazione struttura del documento
1.2	Camborata Marco	2015-03-11	Correzione struttura e contenuti basata su errori segnalati dal committente
1.1	De Lazzari Enrico	2015-03-09	Spostamento sezione Risorse dal PdQ a NdP
1.0	Vegro Federico	2014-12-17	Approvazione documento
0.12	De Lazzari Enrico	2014-12-17	Verifica documento
0.11	Manuto Monica	2014-12-14	Correzioni errori lessicali ed ortografici e stesura ambiente di lavoro
0.10	Gobbo Ismaele	2014-12-13	Correzione norme dei requisiti
0.9	Cossu Mattia	2014-12-12	Stesura norme dei requisiti
0.8	Camborata Marco	2014-12-09	Stesura sezione glossario
0.7	Camborata Marco	2014-12-08	Stesura sezione riunioni
0.6	Cossu Mattia	2014-12-05	Stesura metodi di condivisione
0.5	Manuto Monica	2014-12-05	Stesura sezione documenti
0.4	Manuto Monica	2014-12-05	Correzioni sottosezione comunicazioni
0.3	Gobbo Ismaele	2014-12-04	Stesura sezione comunicazioni
0.2	Camborata Marco	2014-12-03	Stesura Introduzione

0.1	Cossu Mattia	2014-12-02	Stesura scheletro documento

Tabella 1: Storico versioni del documento.

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di delineare le norme e le convenzioni che verranno seguite dal gruppo 404NotFound nel corso di tutte le attività di progetto quali comunicazioni, stesura della documentazione, riunioni e uso degli strumenti. Questo documento è disponibile a tutti i componenti di 404NotFound che dovranno seguirne le convenzioni al fine di ottimizzare il lavoro di gruppo. Durante lo svolgimento delle attività sarà il *Responsabile del Progetto* a verificare l'applicazione delle norme di seguito definite e ad approvare nuove convenzioni proposte dai componenti del gruppo.

1.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un software di presentazione di slide non basato sul modello di PowerPoint$_G$, sviluppato in tecnologia HTML5$_G$ e che funzioni sia su desktop che su dispositivo mobile. Il software dovrà permettere la creazione da parte dell'autore e la successiva presentazione del lavoro, fornendo effetti grafici di supporto allo storytelling e alla creazione di mappe mentali.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio e ai termini utilizzati nei documenti formali tutti i termini e gli acronimi presenti nel seguente documento che necessitano di definizione saranno seguiti da una "G" in pedice e saranno riportati in un documento esterno denominato Glossario.pdf. Tale documento accompagna e completa il presente e consiste in un listato ordinato di termini e acronimi con le rispettive definizioni e spiegazioni.

1.4 Riferimenti

- **Specifiche UTF-8**$-G$**:** http://www.unicode.org/versions/Unicode6.1.0/ch03.pdf;
- **Specifiche HTML5-8**$-G$**:** http://www.w3.org/html/wg/drafts/html/master/.

2 Collaborazione

2.1 Comunicazioni

2.1.1 Comunicazioni esterne

Le comunicazioni esterne avvengono obbligatoriamente tramite l'utilizzo della mail associata al gruppo

404notfound.unipd@gmail.com

tale indirizzo sarà l'unico utilizzabile per tali comunicazioni; non sono ammesse comunicazioni individuali verso l'esterno. La suddetta casella postale sarà utilizzabile solamente dal *Responsabile del Progetto*, il quale avrà quindi l'onere di comunicare con il committente del progetto.

2.1.2 Comunicazioni interne

Le comunicazioni interne formali avvengono attraverso l'utilizzo di una mailing list$_G$ creata all'interno della rubrica di Gmail$_G$. Al momento della composizione andrà indicato come destinatario della mail il nome della mailing list$_G$ creata. Con comunicazioni formali si intende:

- Comunicazione riunione interna;
- Comunicazione riunione esterna;
- Diffusione verbale ultima riunione.

Per altri tipi di comunicazione vengono utilizzati servizi di instant messaging quali Whatsapp$_G$ o Skype$_G$ e un gruppo su Facebook$_G$ appositamente creato per lo scambio di informazioni utili varie ed eventuali.

2.1.3 Compisizione e-mail

In questo paragrafo verranno illustrate le norme secondo le quali comporre le e-mail, in riferimento sia a comunicazioni esterne che interne.

Destinatario Nel caso di più destinatari è possibile utilizzare le e-mail in copia utilizzando il campo "Cc". Vietato l'uso del campo di copia nascosta "Ccn".

- **Mail interne:** il nome della mailing list creata;
- **Mail esterne:** il destinatario potrà essere il committente, il prof. Tullio Vardanega e il prof. Riccardo Cardin.

Mittente

- **Mail interne:** il mittente sarà l'indirizzo del gruppo 404notfound.unipd@gmail.com;
- **Mail esterne:** dovranno sempre avere come mittente l'indirizzo del gruppo di lavoro 404notfound.unipd@gmail.com.

Oggetto L'oggetto deve essere chiaro e conciso e non modificato una volta avviata una comunicazione. Nel caso di risposta va preceduto da "Re:", mentre per un inoltro va utilizzato "I:";

Corpo Il corpo della mail dovrà essere scritto chiaramente e dovrà contenere tutte le informazioni necessarie al fine di rendere comprensibile l'argomento trattato. Nel caso di risposta va mantenuto il corpo delle mail precedenti in modo da creare uno storico della conversazione.

Firma In fondo al corpo della mail andrà indicata la firma "404NotFound".

Allegati Sono permessi allegati solo nel caso in cui fossero strettamente necessari, ad esempio per divulgare il verbale di una riunione. I formati permessi sono PDF_G, JPEG_G, PNG_G, SVG_G.

2.2 Riunioni

Le riunioni del gruppo di lavoro dovranno avere una frequenza almeno quindicinale.

2.2.1 Riunioni interne

Il *Responsabile del Progetto* ha il compito di convocare le riunioni generali, ovvero le riunioni in cui vengono convocati tutti i membri del gruppo, ed eventualmente di valutare se anticipare la naturale scadenza della riunione successiva. Il *Responsabile del Progetto* deve convocare l'assemblea con almeno tre giorni di preavviso attraverso il canale di comunicazione interna (vedi sezione 2.2).

Ciascun componente del gruppo può avanzare una richiesta di riunione interna. Tale richiesta deve pervenire al *Responsabile del Progetto* che provvederà a valutare le motivazioni espresse dal richiedente e ad approvare e quindi indire la riunione.

È inoltre possibile e auspicabile che possano essere necessarie riunioni tra specifici membri del gruppo senza richiedere la presenza di persone non direttamente coinvolte nel lavoro in questione, che verranno comunque informate delle decisioni prese tramite verbale inviato mediante posta elettronica nella mail ufficiale del gruppo.

2.2.2 Riunioni Esterne

Con riunioni esterne si intende qualsiasi incontro fra Proponente/Committente e un gruppo di rappresentanza, composto da almeno la maggioranza assoluta del gruppo di progetto, altrimenti detto *numero legale*.

La richiesta di indire una riunione esterna può essere avanzata da qualsiasi componente del gruppo; è compito del *Responsabile del Progetto* approvare ed eventualmente organizzare l'evento. Una volta che è stata approvata una richiesta, il *Responsabile del Progetto* dovrà contattare, con le modalità scritte nella sezione 2.2, il Proponente/Committente.

Ad ogni incontro verrà incaricato dal *Responsabile del Progetto* un membro del gruppo con il compito di stilare un verbale della riunione, spedito in copia poi a tutti i membri del gruppo per presa visione, verifica e conferma di quanto riportato.

2.3 Metodi di Condivisione

Qui vengono elencate le piattaforme adottate per facilitare lo svolgimento collettivo del progetto.

2.3.1 GitHub

GitHub$_G$ è un servizio di hosting di repository$_G$ Git$_G$.
Una repository$_G$ è uno spazio di archiviazione da cui è possibile recuperare software o codice sorgente; il sistema di controllo di versione$_G$ Git$_G$ in particolare sfrutta la tecnica dei branch$_G$, ossia di ramificazioni del codice sorgente su cui poter lavorare senza entrare in conflitto con il lavoro di altri collaboratori. Il branch originale viene chiamato *master*.
Sono state create due repository$_G$, accessibili dall'indirizzo:
https://github.com/404notfoundunipd/

- `premi` contiene i file di codifica e di sviluppo dell'applicazione eseguibile;
- `premi-documents` contiene i file di tutti i documenti formali soggetti a versionamento compresi i template$_G$ per la loro stesura.

Una volta terminata la fase di lavorazione di un documento, verrà creato un branch di verifica. In questo modo i *Verificatori* potranno lavorare parallelamente al resto del gruppo ed effettuare il merge$_G$ delle loro modifiche, una volta terminato il lavoro di verifica. Il meccanismo di verifica e approvazione è descritto in dettaglio nella sezione 5.5.1.

2.4 Condivisione file

Per la condivisione informale di file e per il lavoro collaborativo su documenti di supporto, si usano le piattaforme di condivisione online Mega$_G$ e Google Drive$_G$. Trattandosi di strumenti informali, non si definiscono procedure rigorose d'uso e se ne lascia la descrizione alle sezioni 6.1.2.1 e 6.1.2.2.

3 Documenti

Il seguente capitolo descrive le convenzioni scelte ed adottate da 404NotFound riguardo alla stesura, verifica e approvazione della documentazione da produrre.

3.1 Template

Come punto di riferimento per della documentazione è stato creato un template$_G$ LaTeX contenente tutte le impostazioni stilistiche e grafiche menzionate in questo documento. Tale modello si trova nel repository$_G$ in documents/template.

3.2 Struttura del documento

3.2.1 Prima pagina

La prima pagina di ogni documento contiene le seguenti informazioni:

- Nome del gruppo;
- Nome del progetto;
- Logo del gruppo;
- Titolo del documento;
- Versione del documento;
- Cognome e nome dei redattori del documento;
- Cognome e nome dei verificatori del documento;
- Cognome e nome del responsabile approvatore del documento;
- Destinazione d'uso del documento;
- Stato del documento;
- Data dell'ultima modifica del documento;
- Lista di distribuzione del documento.

3.2.2 Registro delle modifiche

Nella seconda pagina di ogni documento è riportato il registro delle modifiche del documento.
Ogni riga del registro delle modifiche contiene le seguenti informazioni:

- Versione del documento dopo la modifica;
- Cognome e nome dell'autore della modifica;
- Data della modifica;
- Breve descrizione delle modifiche svolte.

La tabella è ordinata per data in ordine decrescente, in modo che la prima riga corrisponda alla versione attuale del documento.

3.2.3 Indici

La terza pagina di ogni documento contiene un indice delle sezioni, un indice delle tabelle e un indice delle figure. Nel caso non siano presenti tabelle o figure i rispettivi indici non saranno presenti.

3.2.4 Formattazione generale delle pagine

L'intestazione di ogni pagina è caratterizzata da:

- Logo del gruppo;
- Nome del gruppo.

A piè di pagina invece è indicato:

- Nome e versione del documento;
- Pagina corrente nel formato N di T, dove N indica il numero di pagina corrente e T indica il numero di pagine totali.

3.3 Norme tipografiche

In questo paragrafo sono riportate le convenzioni ortografiche, tipografiche adottate e la definizione di uno stile uniforme per tutti i documenti.

3.3.1 Punteggiatura

- **Lettere Maiuscole:** Le lettere maiuscole vanno posto solo all'inizio di ogni elemento di un elenco puntato oltre che dove richiesto dalla lingua italiana. Inoltre l'iniziale maiuscola viene utilizzata nel nome del team, del progetto, dei documenti, dei ruoli di progetto, delle fasi si lavoro e nelle parole Proponente e Committente;
- **Punteggiatura:** un carattere di punteggiatura non deve mai seguire un carattere di spaziatura;
- **Parentesi:** Il testo tra parentesi non deve mai iniziare o terminare con un carattere di spaziatura, ne chiudersi con un carattere di punteggiatura.

3.3.2 Stile del testo

- **Corsivo:** Il corsivo deve essere utilizzato solo nei seguenti casi:
  - **Nomi particolari:** il corsivo deve essere utilizzato quando si parla di figure particolari (es. *Responsabile del Progetto*);
  - **Documenti:** il corsivo deve essere utilizzato quando si parla di documenti (es. Glossario);
  - **Altri casi:** in altre situazione, il corsivo va utilizzato per evidenziare parole significative e riferimenti ai documenti interni o esterni.

- **Grassetto:** Il grassetto può essere utilizzato negli elenchi puntati per evidenziare il concetto che sarà poi sviluppato nella continuazione del punto;
- **Monospace$_G$:** questo carattere viene utilizzato per formattare il testo contenente indirizzi web e percorsi;
- **Maiuscolo:** L'utilizzo di parole completamente in maiuscolo è riservato solo agli acronimi o alle macro LaTeX riportate nei documenti;
- **LaTeX:** Ogni riferimento a LaTeX viene indicato utilizzando il comando `\LaTeX`.

3.3.3 Composizione del testo

- **Citazioni:** Le citazioni devono essere centrate e separate dal testo. In coda alla citazione, spostato verso destra, devono essere indicati l'autore e la fonte da cui è stata estratta la citazione;
- **Elenchi puntati:** Ogni punto dell'elenco deve terminare con un punto e virgola, tranne l'ultimo che terminerà con un punto. La prima parola di ogni punto deve iniziare con la lettera maiuscola a parte in casi particolari, come ad esempio il nome di un file;
- **Parti di codice:** Per riportare porzioni di codice deve essere utilizzato l'ambiente LaTeX *lstlisting*;
- **Note a piè di pagina:** Ogni nota dovrà avere la prima parola che inizia con una lettera maiuscola e dovrà terminare con un punto.

3.3.4 Formati

- **Percorsi:** Per gli indirizzi email e web deve essere usato il comando LaTeX `\href`, mentre per i percorsi relativi dovrà essere usato il carattere monospace$_G$;
- **Date:** Le date riportate nei documenti dovranno seguire lo standard ISO$_G$ 8601:2004:

  AAAA-MM-GG

  Dove:

  - AAAA: Indica l'anno scritto utilizzando quattro cifre;
  - MM: Indica il mese scritto utilizzando due cifre;
  - GG: Indica il giorno scritto utilizzando due cifre.

- **Ruoli di progetto:** per riferirsi ai ruoli di progetto è necessario usare il comando LaTeX `\ruoloNomeruolo`, per garantire la corretta scrittura degli stessi;
- **Nomi propri:** per utilizzare i nomi propri dei membri del team si deve seguire la forma "Cognome Nome";

- **Nome del gruppo:** ci si riferirà al gruppo solo come "404NotFound". Per la corretta scrittura è definita la macro LaTeX `\gruppo`;
- **Nome del proponente:** ci si riferirà al proponente come "Zucchetti srl" o con "Proponente". Per la corretta scrittura è definita la macro LaTeX `\Zucchetti`;
- **Nome del committente:** ci si riferirà al committente come "Prof. Vardanega Tullio" o con "Committente". Per la corretta scrittura è definita la macro LaTeX `\Vardanega`;
- **Nome del progetto:** ci si riferirà al progetto solo come "Premi"; Per la corretta scrittura è definita la macro LaTeX `\capitolato`.

3.3.5 Sigle

In questo paragrafo sono elencate le sigle utilizzabili all'interno dei documenti. Queste sigle potranno essere utilizzate esclusivamente all'interno di tabelle o diagrammi.

- **AdR:** Analisi dei Requisiti;
- **GL:** Glossario;
- **NdP:** Norme di Progetto;
- **PdP:** Piano di Progetto;
- **PdQ:** Piano di Qualifica;
- **SdF:** Studio di Fattibilità;
- **ST:** Specifica Tecnica;
- **RA:** Revisione di Accettazione;
- **RP:** Revisione di Progettazione;
- **RQ:** Revisione di Qualifica;
- **RR:** Revisione dei Requisiti.

3.4 Componenti grafiche

3.4.1 Tabelle

Ogni tabella presente all'interno dei documenti dev'essere accompagnata da una didascalia che comprende un numero identificativo incrementale utile a rintracciare la stessa all'interno del documento.

3.4.2 Immagini

Le immagini presenti all'interno dei documenti devono essere nel formato PNG_G (Portable Network Graphics). Ogni immagine deve essere accompagnata da una didascalia in cui deve comparire un numero identificativo incrementale utile a rintracciare la stessa all'interno del documento.

3.5 Classificazione dei documenti

3.5.1 Documenti informali

Un documento è considerato informale fino a quando non viene approvato dal *Responsabile del Progetto*. Di conseguenza tutti i documenti informali sono da ritenersi esclusivamente ad uso interno.

3.5.2 Documenti formali

Un documento diventa formale nel momento in cui riceve l'approvazione del *Responsabile del Progetto* ed è quindi pronto per essere condiviso con i richiedenti. Per giungere fino a questo punto il documento deve seguire un processo di verifica e validazione descritto nel paragrafo 6.7 riguardante il ciclo di vita dei documenti.

3.6 Versionamento

Tutta la documentazione prodotta deve indicare il numero di versione versione attuale, riportato nel modo seguente:

vX.Y

Dove:

- X: indica il numero crescente di uscite formali del documento;
- Y: indica il numero crescente di modifiche al documento.

Ogni qualvolta si presenti la necessità di citare una versione specifica di un documento, essa deve comprendere sia il nome che il numero di versione nel seguente formato:

Nome Documento_vX.Y

3.7 Ciclo di vita

Ogni documento, dal momento della sua creazione fino all'approvazione finale, segue un percorso ben preciso durante il quale può assumere tre diversi stati:

- **In lavorazione:** Un documento si trova in questa fase dal momento in cui viene creato e per tutto il periodo necessario alla sua realizzazione. Ci si può trovare in questa fase anche per eventuali modifiche successive;
- **Da verificare:** Una volta che il documento viene terminato, esso deve essere preso in consegna dai verificatori che hanno il compito di rilevare e correggere eventuali errori o imprecisioni;
- **Approvato:** Una volta terminata la fase di verifica il documento deve essere approvato dal *Responsabile del Progetto*. Questo rappresenta lo stato finale del documento.

Queste fasi posso essere attraversate più volte da uno stesso documento.

3.8 Glossario

Nel glossario verranno elencate, in ordine alfabetico, tutte le definizioni di termini tecnici o soggetti ad ambiguità, in modo da rendere univoco il senso di tali termini.
La stesura del glossario deve avvenire parallelamente alla stesura di tutti i documenti, inserendo un termine con relativa definizione ogni qualvolta se ne incontra uno nuovo.
Allo stesso modo andrà posizionata una "G" a pedice a fianco al vocabolo all'interno del documento utilizzando la formula LaTeX $_G$.

4 Risorse

La realizzazione del progetto richiede l'utilizzo di alcune risorse suddivisibili in diverse categorie.

4.1 Risorse Necessarie

4.1.1 Risorse Umane

- ***Responsabile del Progetto***: E' responsabile nei confronti del committente della corretta realizzazione del prodotto;
- ***Analista***: E' colui che si occupa dell'individuazione dei requisiti, impliciti ed espliciti del progetto.
- ***Progettista***: Ha il compito di trovare una soluzione possibile per lo sviluppo.
- ***Verificatore***: Coordina e svolge le attività di verifica vere e proprie;
- ***Programmatore***: Codifica ed esegue attività di debugging sul codice.
- ***Amministratore***: Ha il compito di gestire e garantire le risorse e l'ambiente di sviluppo e di lavoro all'interno del gruppo.

Per una descrizione dettagliata e più completa delle figure elencate si rimanda al documento allegato *PianoDiPrgetto_v2.0.pdf*.

Rotazione dei ruoli

Da regolamento[**?**] ogni componente del gruppo può occupare più di un ruolo nelle varie fasi del progetto, garantendo però l'assenza di conflitto di interessi tra i ruoli assunti. Questo richiede che le seguenti norme vengano attuate nell'assegnazione dei ruoli:

- rotazione del ruolo di *Responsabile del Progetto* ad ogni macro-fase per non lasciare in mano allo stesso componente il coordinamento delle attività per tutto l'arco temporale del progetto;
- la verifica di un documento o di parte del codice dev'essere sempre eseguita da componenti che non hanno scritto quel documento o codificato quella parte di software.

Per facilitare lo sviluppo del software vengono inoltre suggerite le seguenti raccomandazioni:

- separazione del ruolo di *Responsabile del Progetto* nei suoi incarichi di redattore di documenti (che potranno essere scritti da più componenti) e di coordinatore di attività (insieme di incarichi che dovranno essere svolti dal Responsabile assegnato per la macro-fase corrente);
- specializzazione di componenti del gruppo, quando ritenuto opportuno, nella stesura di determinati documenti per una maggiore efficienza nella loro elaborazione e verifiche incrociate su tali documenti per assicurare che ognuno li conosca in dettaglio.

4.1.2 Risorse Software

Durante la fase di realizzazione del progetto saranno necessari:

- Software per la gestione di documenti in LaTeX;
- Piattaforma di testing sui vari browser$_G$ dell'applicazione da sviluppare;
- Piattaforma di versionamento per la creazione e la gestione di ticket$_G$;
- Software per la creazione dei diagrammi in UML$_G$;
- Ambiente per lo sviluppo del codice nel linguaggio di programmazione scelto;
- Strumenti di validazione del codice prodotto.

4.1.3 Risorse Hardware

- Computer dotati di tutti gli strumenti software descritti nel Piano di Qualifica e nelle Norme di Progetto;
- Luogo fisico in cui incontrarsi per lo sviluppo del progetto, possibilmente con una connessione ad Internet.

4.2 Risorse Disponibili

4.2.1 Risorse Software

Vengono di seguito elencate le risorse software disponibili. Per una descrizione più dettagliata si rimanda alla sottosezione Strumenti.

- TeXMaker per l'editing dei documenti in LaTeX;
- BrowserStack per il testing sui vari browser$_G$;
- GitHub per il versionamento e la gestione dei ticket$_G$;
- Astah per i diagrammi UML$_G$;
- WebStorm come ambiente di sviluppo;
- Strumenti di validazione online del W3C$_G$.

4.2.2 Risorse Hardware

- Computer personali (portatili o fissi) dei membri del gruppo;
- Computer messi a disposizione nei laboratori informatici del Dipartimento di Matematica Pura ed Applicata dell'Università di Padova;
- Aule studio del Dipartimento di Matematica Pura ed Applicata dell'Università di Padova.

5 Procedure a supporto dei processi

Le seguenti norme regolano i processi principali del ciclo di vita del software. Ogni ruolo di progetto è tenuto a seguirli per il raggiungimento degli obiettivi fissati nel *Piano di Qualifica.*

5.1 Gestione di Progetto

La gestione del progetto, dalla nascita alla conclusione, spetta al *Responsabile del Progetto.* Egli ha il compito di:

5.1.1 Pianificare le attività

(Pianificare in anticipo le attività e tenere traccia del loro corretto svolgimento)

Questa attività è eseguita attraverso lo studio dei processi e la stima delle ore richieste per portare a termine le attività da cui sono composti. Servirsi dei dati ricavati dalle attività compatibili già portate a termine dal gruppo o, quando non presenti, utilizzare dati provenienti da fonti esterne attendibili.
Una volta creato un piano di lavoro per una macro-fase, il *Responsabile del Progetto* deve realizzare un diagramma di Gantt$_G$ con il software ProjectLibre$_G$, il cui compito è quello di mostrare la distribuzione temporale delle attività. Il calcolo delle ore viene invece eseguito con l'aiuto di un foglio elettronico; per questo scopo è stato realizzato un template$_G$, disponibile nella Repository$_G$ del progetto. Tutte informazioni ricavate devono essere disponibili all'interno del *Piano di Progetto*, che dovrà essere sempre aggiornato e riportare ogni modifica effettuata sulla pianificazione delle attività.

5.1.2 Assegnare attività a risorse

(Collocare e gestire le risorse necessarie e sufficienti allo svolgimento delle attività)

Per l'assegnazione delle attività alle risorse, e per il controllo periodico del loro progresso, il *Responsabile del Progetto* ha a disposizione il sistema di ticketing$_G$ di GitHub, descritto nella sezione 5.7.

5.1.3 Studiare i rischi

(Studiare i possibili rischi a cui il progetto puo' andare incontro, controllare periodicamente la possibilità che si verifichino e gestire quelli che si sono verificati)

I rischi, il loro grado di pericolosità, la loro probabilità di avverarsi e le contromisure da adottare per prevenirli o correggerli vengono descritti in dettaglio nel *Piano di Progetto.* Tutte queste attività devono essere il più possibile automatizzate o supportate da strumenti in grado di tenere traccia dei compiti portati a termine. Tali strumenti vengono descritti nella sezione 6.

5.2 Analisi dei Requisiti

5.2.1 Studio di Fattibilità

In seguito alla scelta del capitolato da parte del gruppo è necessario che gli *Analisti* redigano uno *Studio di Fattibilità*. Questo documento dovrà focalizzarsi su:

- **Dominio applicativo e tecnologico:** la conoscenza, da parte dei componenti del gruppo, del dominio applicativo e delle tecnologie richieste nel capitolato scelto;
- **Guadagno:** rapporto tra costi per la realizzazione e dei benefici ricavati dal prodotto completo;
- **Analisi dei rischi:** individuazione dei rischi derivati da lacune di conoscenza del dominio e dalle tecnologie richieste;
- **Confronto con gli altri capitolati**: elenco delle ragioni che hanno portato a scartare gli altri capitolati.

5.2.2 Requisiti

In seguito allo studio di fattibilità gli *Analisti* dovranno identificare e classificare i Requisiti.
I Requisiti devono essere individuati attraverso lo studio delle seguenti fonti:

- il capitolato d'appalto;
- il dominio applicativo;
- i documenti ufficiali del committente;
- i resoconti dell'interazione col fornitore (vedi 2.2.2);
- i resoconti delle riunioni con il committente;
- i resoconti delle riunioni di gruppo (brainstorming$_G$);
- la prototipazione di alcune parti del prodotto;

La classificazione verrà operata sui seguenti attributi:

- **Tipo del Requisito** i cui valori possono essere:

  F Funzionale;
  Q di Qualità;
  V di Vincolo;
  P Prestazionale.

- **Valore del Requisito** i cui valori possono essere:

  Ob Obbligatorio;
  De Desiderabile;

Op Opzionale.

- **Codice univoco** è il codice espresso in modo gerarchico come segue.

  [codice del padre].[numero unico rispetto ai fratelli]

Oltre a questa classificazione è necessario aggiungere una breve descrizione ad ogni requisito

5.2.3 Casi d'Uso

In seguito alla classificazione dei requisiti gli *Analisti* dovranno identificare i *casi d'uso* o *use case*. Di ogni caso d'uso interessa:

- **Titolo** breve titolo che identifica lo UC;
- **Pre e Post Condizione**;
- **Attori** principali e secondari;
- **Scenari** principali e alternativi definendo per ognuno titolo, attori coinvolti, una breve descrizione e il caso d'uso a cui si riferiscono se presente;
- **Descrizione** breve descrizione dello UC;
- **Requisiti Dedotti**.

5.2.4 Tracciamento

I requisiti, i casi d'uso e le fonti dovranno essere inseriti, dagli *Analisti*, in un database$_G$ appositamente creato. Per facilitare le operazioni necessarie è stata creata un'applicazione web il cui nome è 404TrackerDB. Compito dell'applicazione è anche quella di creare il codice Latex, sia delle tabelle per il tracciamento sia delle generalità di fonti e requisiti, in modo automatico per velocizzare la stesura del documento *Analisi dei Requisiti*.

5.3 Progettazione Architetturale

È compito dei *Progettisti* studiare la struttura dei componenti più adatta a soddisfare i requisiti tracciati nella fase di Analisi.
La progettazione dovrebbe essere il più possibile indipendente dai linguaggi di programmazione e dagli ambienti di sviluppo che verranno adottati nella fase di codifica del software, a meno che non vengano richiesti esplicitamente dal proponente.

5.3.1 Specifica Tecnica

Il documento che raccoglie le informazioni ricavate durante la progettazione architetturale è la *Specifica Tecnica*, e deve essere composto dalle seguenti sezioni:

- **Tecnologie Utilizzate:** Devono essere esposte le tecnologie ed i design pattern$_G$ su cui si basa la progettazione del software.

- **Descrizione dell'Architettura:** l'architettura generale del sistema deve essere introdotta da una breve descrizione dei metodi e formalismi di specifica adottati durante la fase di progettazione, con particolare attenzione ad eventuali dipendenze dall'ambiente di sviluppo adottato.
- **Diagrammi dei Package:** I componenti devono essere rappresentati da package$_G$ che definiscono i namespace$_G$ per le classi, le librerie o gli oggetti da cui saranno composti. Ogni componente deve essere rappresentata seguendo lo standard UML$_G$ 2.0, e seguita da una descrizione che indichi la propria funzione all'interno del software, ed eventuali dipendenze con altri componenti. Package$_G$ e classi interne devono essere citate con la seguente notazione:
  `PackageEsterno.PackageInterno.Classe`
- **Descrizione dei singoli componenti:** Ogni elemento proprio del software presente all'interno di un componente deve essere descritto nel modo seguente:
  - **Nome:** della classe o dell'elemento
  - **Tipo:** classe, classe anonima, etc.
  - **Package:** di appartenenza, con la notazione scritta sopra
  - **Descrizione:** della classe e delle funzionalità che mette a disposizione
  - **Relazioni con altri componenti:** Dipendenze, associazioni, relazioni di ereditarietà o implementazione secondo la notazione UML2.0$_G$.

  Suddividere le classi per package e accompagnare ogni package da un Diagramma delle Classi utilizzando la notazione UML$_G$ 2.0. Evidenziare in particolar modo le relazioni di dipendenza, sia interne che esterne.
- **Diagrammi delle attività:** Le azioni che l'utente puo' compiere attraverso i componenti, e la logica procedurale del software, devono essere descritti tramite i Diagrammi delle Attività secondo la notazione UML$_G$ 2.0. Anche qui ogni diagramma dovrà essere accompagnato da una breve descrizione degli obiettivi che l'attività si propone di raggiungere.
- **Stime di Fattibilità:** In questa sezione viene appurata la conformità e l'efficienza delle tecnologie utilizzate per lo sviluppo del software.
- **Tracciamento delle relazioni classi-requisiti:** Ogni classe, ed eventualmente ogni componente, deve corrispondere alla soddisfazione di uno o più requisiti, o di parte di essi. Allo stesso tempo ogni requisito deve poter essere soddisfatto da una o più classi o componenti. L'applicazione 404TrackerDB si occuperà di creare il codice Latex delle relazioni inserite nella sua base di dati, e di verificare che tali norme vengano rispettate.
- **Design Pattern:** Ogni design pattern$_G$ utilizzato per costruire l'architettura del software deve essere descritto in dettaglio in questa sezione apposita, in modo da fornire ai membri del gruppo e ai lettori del documento una fonte sempre disponibile delle informazioni necessarie a capire i componenti ed il loro modo di interagire con il resto del software.

5.4 Progettazione di Dettaglio

5.4.1 Definizione di Prodotto

È compito dei *Progettisti* utilizzare quanto scritto nella *Specifica Tecnica* per produrre la *Definizione di Prodotto* dove viene descritta la progettazione di dettaglio del sistema. Lo scopo di questo documento è quello di definire dettagliatamente ogni singola unità di cui è composto il sistema in modo da semplificare l'attività di codifica e allo stesso tempo di non fornire alcun grado di libertà al *Programmatore*.

5.5.1.1 Diagrammi UML

Devono essere redatti i seguenti diagrammi:

- diagrammi delle classi;
- diagrammi di attività;
- diagrammi di sequenza.

5.5.1.2 Definizione di classe

Ogni classe progettata deve essere descritta all'interno della "Definizione di Prodotto". Tale descrizione deve comprendere una spiegazione sullo scopo della classe e deve specificare quale funzionalità essa modella. Nella descrizione devono inoltre essere presenti l'elenco di metodi e attributi della classe. La descrizione di ogni classe deve essere composta da:

- **Diagramma della Classe:** secondo il formalismo UML_G 2.0;
- **Classi ereditate:** la gerarchia a cui appartiene la classe;
- **Attributi:** propri della classe, con una lista formata da:
  Nome scritto come `accesso nome : tipo`, *Accesso*, *Tipo*, *Descrizione*;
- **Metodi:** propri della classe o ridefiniti, con una lista formata da:
  Nome scritto come `accesso nome(tipoAtt1 attributo1) : tipoRitorno`, *Accesso*, *Tipo di ritorno*, *Parametri*, *Descrizione*

5.5.1.3 Tracciamento requisiti - classi

Ogni classe deve soddisfare un requisito o parte di esso. Allo stesso tempo ogni requisito deve essere soddisfatto da una o più classi. Le classi vanno inserite nel database gestito dal software 404TrackerDB, il quale si occuperà di verificare la soddisfacibilità dei requisiti e di generare il codice Latex appropriato.

5.5.1.4 Test di unità

Durante la *Progettazione di Dettaglio* i *Progettisti* hanno inoltre il compito di definire i test di unità per assicurarsi che i componenti realizzati siano privi di difetti.

5.5 Verifica

La verifica di processi, documenti e prodotti è un'attività da eseguire continuamente durante lo sviluppo del Progetto. Di conseguenza, servono modalità operative chiare e dettagliate, in modo da uniformare le attività di verifica svolte ed ottenere il miglior risultato possibile. Si descrivono ora le modalità di verifica di processi, documenti, attività e codice alle quali ci si riferirà in questo documento e alle quali i Verificatori dovranno attenersi.

5.5.1 Verifica dei documenti

Il processo di verifica viene istanziato nel momento in cui l'output di un processo passa dalla versione X.Y alla versione X+1.0. È compito del *Responsabile del Progetto* notificare i Verificatori dell'inizio dell'attività di verifica. Attraverso il diario delle modifiche è possibile focalizzare l'attenzione maggiormente sulle sezioni che hanno subito dei cambiamenti dall'ultima verifica, riducendo il tempo necessario al controllo. Per eseguire un'accurata verifica dei documenti redatti è necessario seguire il seguente protocollo:

1. **Controllo sintattico del periodo:** Utilizzando TeXMaker$_G$ vengono evidenziati gli errori ortografici. Gli errori di sintassi, di sostituzione di lettere che provocano la creazione di parole grammaticalmente corrette ma sbagliate nel contesto ed i periodi di difficile comprensione necessitano dell'intervento di un verificatore umano. Per questa ragione ciascun documento dovrà essere sottoposto ad un walkthrough da parte dei verificatori per individuare tali errori;
2. **Rispetto delle norme di progetto:** In questo documento sono state definite norme tipografiche di carattere generale. Queste regole impongono una struttura dei documenti che non può essere verificata in maniera automatica, di conseguenza è necessario che i *Verificatore* esegua inspection sul rispetto di quelle norme in ciascun documento;
3. **Verifica delle proprietà di glossario:** Il *Verificatore* dovrà controllare che per ogni termine contrassegnato come definito nella sezione 3.8 sia presente la definizione corrispondente all'interno del Glossario;
4. **Miglioramento del processo di verifica:** Per avere un miglioramento del processo di verifica, quando il *Verificatore* utilizza la tecnica walkthrough su un documento, dovrà riportare gli errori più frequenti, per consentire l'utilizzo di inspection su tali errori nelle verifiche future;

5.5.2 Verifica dei diagrammi UML

Al *Verificatore* è richiesto il controllo dei diagrammi UML$_G$ prodotti:

- **Diagrammi dei casi d'uso:** Il *Verificatore* deve controllare il rispetto delle specifiche UML$_G$ 2.4 e il corretto uso delle relazioni di inclusione ed estensione. Il diagramma di caso d'uso deve rappresentare fedelmente quanto descritto dal caso d'uso;

- **Diagrammi delle classi:** Il *Verificatore* ha il compito di controllare il rispetto del formalismo UML$_G$ 2.0 e la corrispondenza tra progettazione e diagrammi delle classi.

5.5.3 Verifica del codice

Per la verifica del codice al *Verificatore* è richiesto l'avvio dei test statici e dinamici riportati nella sezione 2.5.2 del *PianoDiQualifica_v2.0.pdf* e l'analisi dei risultati ottenuti.

5.5.4 Gestione delle anomalie

Lo strumento scelto per la gestione delle anomalie è la sezione "Issue" messa a disposizione da Github$_G$. Coerentemente con l'organizzazione generale delle strategie di verifica, nuove anomalie potranno essere scoperte in due modi:

- In ogni fase di verifica, il *Verificatore* avrà il compito di cercare eventuali anomalie;
- Grazie all'approccio "Broken Window Theory" (*PianoDiQualifica_v2.0.pdf* sezione 2.1), chiunque in qualunque momento è incentivato alla ricerca di possibili anomalie.

Nel caso in cui un *Verificatore* o un membro del gruppo individui un anomalia dovrà segnalarlo aprendo un ticket$_G$ (vedi sezione 5.2). Un *Verificatore* ha il compito di controllare le pull request quindi nel caso trovasse un anomalia deve impedire la pull request con le modalità descritte nella sezione 5.4.

5.6 Codifica

5.6.1 Codifica e convenzioni

- Tutti i file contenenti codice o documentazione devono essere conformi alla codifica UTF$_G$-8 senza BOM$_G$ (poiché potrebbe causare problemi nella verifica). Per andare a capo viene usato il carattere LF$_G$ (U+000A).
- È ammessa la possibilità di effettuare modifiche alle convenzioni stabilite, in seguito ad una decisione del *Responsabile del Progetto*.

5.6.2 Nomi e norme stilistiche

- I nomi di variabili, classi, funzioni e metodi devono essere in inglese e privi di underscore;
- I commenti devono essere scritti in lingua italiana;
- I nomi delle classi devono avere la prima lettera maiuscola;
- 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.

5.6.3 Ricorsione

La ricorsione va evitata quando possibile. Per ogni funzione ricorsiva è necessario fornire una prova di terminazione. È inoltre necessario valutare il costo in termini di occupazione della memoria. Nel caso in cui l'utilizzo di memoria risulti troppo elevato, la ricorsione verrà rimossa.

5.6.4 Intestazione File

Tutti i file di codice sorgente devono iniziare con la seguente intestazione:

```
/**
  * Name:      [nome del file]
  * Package:     [directory completa]
  * Author:      [creatore del file]
  * Date:      AAAA-MM-GG [data di creazione del file]

  * Use:
  [cosa fa questo file??]

  * Changes:
  Version    Date     Who        Changes        Reason
   ------------------------------------------------------------------------

  [x].[y]    AAAA-MM-GG          [cambiamento]    [motivazione]
   ------------------------------------------------------------------------

  [x].[y]    AAAA-MM-GG          [cambiamento]    [motivazione]
   ------------------------------------------------------------------------


  * Created by 404Notfound for Premi - Better than Prezi!

  * Premi is a free software: you can redistribute it and/or modify
  * it under the terms of the GNU General Public License as published by
  * the Free Software Foundation, either version 3 of the License, or
  * (at your option) any later version.

  * This program is distributed in the hope that it will be useful,
  * but WITHOUT ANY WARRANTY; without even the implied warranty of
  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
  * GNU General Public License for more details.

  * You should have received a copy of the GNU General Public License
  * along with this program. If not, see <http://www.gnu.org/licenses/>
*/
```

I campi andranno aggiornati ad ogni modifica del codice. Cercare di adeguarsi il più possibile alla tabulazione basilare dell'intestazione.

- **Name:** il nome e l'estensione del file;

- **Package:** la directory in cui è stato inserito il file, partendo dalla cartella premi. Esempio: `premi/server/`;
- **Author:** il creatore del file.;
- **Date:** la data di creazione del file;
- **Use:** il perché dell'esistenza del file;
- **Changes:** il registro delle modifiche del file. La modifica più recente si trova in cima alla lista, mentre quella meno recente alla fine:
  - *Version:* versione del file;
  - *Date:* data di modifica del file;
  - *Who:* chi ha modificato il file;
  - *Changes:* dove sono state effettuate le modifiche;
  - *Reason:* le ragioni della modifica.

5.6.5 Commenti nel codice

Lo scopo di ogni metodo e ogni attributo di classi, view e controller dovrebbe essere comprensibile anche ad un programmatore esterno al progetto. Quando metodi e attributi non sono di facile comprensione, inserire dei commenti che ne descrivano il funzionamento, l'utilizzo, e le ragioni della loro creazione. Servirsi di commenti inline e blocchi di commenti come da esempio:

```
angular.module("esempio.commenti")
  .factory('Classe', [
    function() {
            return klass(function () {
                this.info = {
                    campo1 : "stringa", // scopo dell'attributo
                    campo2 : 0, // scopo dell'attributo
                };
            })
            .methods({
                /*  metodoDifficile
                    perche' questo metodo e' stato creato?
                    cosa fa? come lo fa?*/
                metodoDifficile : function(field,value){
                    if(field == 'id'){
                        this.info._id = value;
                    }
                    if(this.info.hasOwnProperty(field)){
                        this.info[field] = value;
                    }
                    return this; // restituisce un riferimento all'
  oggetto
                },
                /* metodoSemplice
                    questo metodo e' di facile comprensione e
                    non andrebbe commentato */
                metodoSemplice : function(){
```

```
27                  return campo1;
                },
```

5.6.6 Formattazione del Codice

Per la formattazione del codice utilizzare la formattazione automatica del software WebStorm$_G$, ad ogni modifica apportata e ad ogni nuovo file aggiunto.

5.7 Procedure per l'utilizzo del repository

Vengono qui elencate le norme per l'utilizzo degli strumenti di sviluppo del sito web GitHub$_G$.
Per una guida completa all'uso di GitHub$_G$ si consiglia il sito:
https://guides.github.com/
Per una guida completa sull'uso di git$_G$ da terminale si consiglia il sito:
http://git-scm.com/doc/

5.7.1 Creare una milestone

La creazione di una milestone$_G$ viene affidata esclusivamente al *Responsabile del Progetto* e il suo scopo è quello di suddividere il lavoro in elenchi di attività correlate tra loro per un maggiore controllo sullo stato di avanzamento del progetto. Le attività possono essere a loro volta suddivise in compiti che vengono spartiti tra i componenti del gruppo attraverso la creazione di ticket$_G$.
Milestone e ticket vengono creati seguendo i calendari delle attività inseriti nel *Piano di Progetto*.
Dalla pagina della repository$_G$ accedere al menu *issues*, andare sulla sezione *Milestones* e quindi su *New milestone*$_G$.
Inserire un breve titolo che la distingua dalle milestone$_G$ precedenti, una descrizione approfondita che identifichi le parti del progetto su cui si sta lavorando e infine la data di chiusura.

5.7.2 Creare un ticket

Un ticket$_G$ corrisponde ad un compito da portare a termine all'interno di una milestone$_G$.
Vengono creati dal *Responsabile del Progetto* e assegnati a sua discrezione ad un componente del gruppo.
Dalla pagina della repository$_G$ accedere al menu *issues* e premere su *New Issue*.
È richiesta la compilazione dei seguenti campi:

- *Title:* descrive sinteticamente il compito assegnato;
- *Assigned to:* è il membro del gruppo che si occuperà del compito;
- *Comment:* descrizione approfondita del compito. È possibile includere immagini esplicative se ritenuto necessario;
- *Milestone$_G$:* per includere il ticket$_G$ in un contesto temporale;

- *Labels:* aiutano a suddividere i compiti per tipo.

È possibile creare dei collegamenti ai ticket$_G$ all'interno dei commenti in GitHub$_G$ scrivendo l'id del ticket preceduto dal carattere #.

5.7.3 Eseguire il compito

Per eseguire il compito assegnato, l'incaricato deve creare un branch$_G$ su cui lavorare. Dalla pagina della repository$_G$ premere sul pulsante della lista dei branch$_G$ e digitare il nome del nuovo branch$_G$, che deve essere formato dal nome dell'incaricato seguito dall'id del ticket$_G$ nella forma:

`nome-id`

Se quello su cui si sta lavorando non dovesse essere collegato a nessun ticket$_G$ inserire, invece dell'id, un nome che identifichi il compito.
Viene lasciato a discrezione dell'incaricato il numero di salvataggi (commit$_G$) da effettuare. È però consigliabile salvare i files su su cui si sta lavorando nella repository$_G$ remota almeno una volta al giorno per non perdere il lavoro in caso di incidenti e in modo che il resto del gruppo possa vedere lo stato di avanzamento del compito.
Ogni commit$_G$ deve essere accompagnato da un breve titolo e una descrizione che elenchi quello che è stato aggiunto al branch$_G$.
Quando il lavoro è concluso inviare una *pull request*$_G$, ossia la richiesta di riunire il branch$_G$ con l'originale. Dalla pagina della repository$_G$ accedere al menu *Pull Requests*$_G$ e premere sul pulsante *New pull request*; selezionare il branch$_G$ base e quello su cui si ha eseguito il compito e premere *Create pull request*.
È richiesta la compilazione dei seguenti campi:

- *titolo:* deve essere il nome del branch$_G$ da riunire;
- *commento:* deve contenere almeno l'id del ticket$_G$(*issue*) del compito eseguito nella forma `#id` e preceduto dalla parola chiave `Fixes` , per rendere automatica la chiusura del ticket una volta accettata la richiesta.

5.7.4 Verifica del compito

Il *Verificatore* ha il compito di esaminare le *pull request*$_G$ proposte dai membri del gruppo.
Dopo aver verificato il lavoro svolto, dal menu *Pull Requests*$_G$ ha tre possibilità:

- *Accettare la richiesta:* Il lavoro è corretto e soddisfa i requisiti del compito; il branch$_G$ viene unito al *master* e il ticket$_G$ viene chiuso. Questa azione viene chiamata *merge*$_G$;
- *Comunicare con il richiedente:* Se il lavoro presenta degli errori o non soddisfa i requisiti del compito, il *Verificatore* può lasciare dei commenti all'interno della *pull request*$_G$ in cui descrive i problemi riscontrati. L'incaricato è tenuto a risolvere i problemi e ad aggiornare il branch$_G$;
- *Rifiutare la richiesta:* La richiesta non è pertinente al lavoro in corso e viene chiusa senza effettuare il *merge*$_G$. Il *Verificatore* è tenuto comunque a lasciare un commento che giustifichi tale azione.

5.7.5 Chiudere una milestone

Quando tutti i ticket$_G$ sono stati chiusi il *Responsabile del Progetto* può chiudere la milestone$_G$ in corso.
Dalla pagina della repository$_G$ accedere al menu *issues*, andare sulla sezione *Milestones* e selezionare quella che si intende chiudere; infine premere su *close*.

6 Ambiente di Lavoro

6.1 Coordinamento

6.1.1 Software di versionamento

E' stato scelto di utilizzare GitHub$_G$ come software di versionamento. I motivi principali della scelta sono:

- **Flessibilità:** Git$_G$ è un repository$_G$ distribuito con la possibilità di commit$_G$ e revert locali;
- **Esperienza del team:** Git$_G$ è già stato usato da alcuni componenti del gruppo.

Inoltre Git$_G$ mette a disposizione un'interfaccia grafica (SmartGit$_G$) e un sistema di integrazione con WebStorm. Nella sezione 5.7 del documento corrente sono specificate alcune procedure di utilizzo del repository$_G$.

6.1.2 Condivisione dei file

6.1.2.1 Mega

Per la condivisione delle esportazioni delle Virtual Machine$_G$ con Ubuntu Desktop$_G$ è stato scelto il servizio cloud$_G$ Mega, che dà a disposizione 50GB di spazio gratuito.

6.1.2.2 Google Drive

Google Drive$_G$ permette una facile condivisione di documenti informali e di tutte quelle risorse utili al progetto che non sono soggette a versionamento.
Per sfruttare nel modo migliore le potenzialità della piattaforma si raccomanda di attenersi alla suddivisione intuitiva in cartelle dei file e di utilizzare, quando possibile, formati compatibili con gli editor$_G$ testuali del sito.

6.1.3 Google Calendar

Google Calendar$_G$ viene utilizzato come promemoria per le date degli incontri e per segnalare le fasce orarie in cui un membro non è disponibile in caso di impegni settimanali inderogabili.

6.1.4 Mozilla Thunderbird

E' stato scelto Mozilla Thunderbird$_G$ come client di posta, in quanto è uno strumento utilizzato quotidianamente da molti componenti del gruppo, e inoltre è già presente nell'installazione base di Ubuntu desktop. Questo client, permette l'installazione di un plugin a noi necessario: Lightning$_G$. Quest'ultimo verrà utilizzato per sincronizzare i calendari di Google Calendar$_G$.

6.2 Pianificazione

Per pianificare le attività legate allo sviluppo del progetto e la gestione delle risorse si è scelto di utilizzare **ProjectLibre**. ProjectLibre è un programma open source per il project management basato su Java$_G$ e quindi eseguibile su ogni sistema operativo. Le principali caratteristiche di tale software sono:

- Portabilità, essendo basto su Java$_G$;
- Open-source;
- Genera diagrammi Gantt$_G$;
- Genera automaticamente diagrammi Program Evaluation and Review Technique(PERT$_G$), a partire dal Gantt$_G$;
- Salvataggio su file XML$_G$: essendo un file testuale è possibile effettuare il merge$_G$ di più file in caso di conflitti sul repository$_G$.

6.2.1 Modalità di utilizzo

Il *Responsabile del Progetto* ha il compito di creare un progetto per ogni macro-fase indicata nella sezione Pianificazione delle attività del Piano di Progetto v2.0 e procedere nel modo seguente:

1. Creare un calendario lavorativo per il progetto;
2. Inserire le attività da svolgere e le corrispondenti sotto-attività;
3. Inserire le dipendenze temporali tra le attività;
4. Inserire i periodi di slack dove previsto;
5. Inserire la milestoneG per indicare il termine previsto delle attività;
6. Creare le risorse e suddividerle adeguatamente tra le attività definite.

6.3 Strumenti per la documentazione

6.3.1 Stesura documenti

Per la stesura dei documenti si è scelto di usare il linguaggio di markup$_G$ LaTeX. Il motivo principale che ha portato a questa scelta è la facilità di separazione tra contenuto e formattazione: con LaTeXè possibile definire l'aspetto delle pagine in un file template condiviso da tutti i documenti, cosa che non sarebbe stata possibile optando per altre soluzioni come Microsoft Office o LibreOffice. Inoltre permette la definizione di funzioni e variabili globali garantendo una formattazione del testo uniforme a tutti i documenti e la scrittura del contenuto più corretta da un punto di vista semantico. Il team 404NotFound si avvarrà del software *TexMaker* versione 4.4.1 per la redazione, la compilazione e l'esportazione dei file in LaTeX;

6.3.2 Verifica

Per il processo di verifica dei documenti si è scelto di utilizzare i seguenti strumenti:

- **Correttore automatico di TeXMaker$_G$:** l'ambiente grafi TeXMaker$_G$, già utilizzato per la stesura dei documenti, integra i dizionari di OpenOffice.org e segnala i potenziali errori ortografici presenti nel testo;

- **404TrackerDB**: Strumento software realizzato dal gruppo 404NotFound che contiene ed associa:
  - Requisiti individuati durante l'analisi;
  - Fonti di requisiti individuate, inclusi anche i casi d'uso.

  Permette inoltre di esportare automaticamente:
  - Codice LaTeX per la descrizione dei casi d'uso;
  - Tabella in LaTeX per il tracciamento fonti-requisiti.

6.3.3 Diagrammi UML

Per la creazione dei diagrammi UML$_G$ è stato scelto di usare il software *Astah* versione community 6.9.0, disponibile per il sistema operativo Linux installato nella Virtual Machine$_G$.

6.4 Strumenti di sviluppo

6.4.1 Ambiente

- **Virtual Machine$_G$**:

  Come ambiente di sviluppo si è scelto di creare una macchina virtuale$_G$ (VM) con Ubuntu Desktop 14.04, per avere a disposizione un ambiente uniforme per tutti i componenti del gruppo 404NotFound. La VM sarà disponibile per il download ai componenti del team nell'account Mega$_G$

- **Virtual Box versione 4.3.20**: è il software utilizzato per far girare la macchina virtuale sui personal computer de vari componenti del gruppo di lavoro;
- **Server dedicato**: 404NotFound si avvale dell'uso di un server dedicato gestito da un applicazione creata dai membri del team di sviluppo denominata 404TrackerDB. Questo strumento servirà a contenere ed associare i requisiti individuati in fase di analisi;
- **Inkscape versione 0.48**: Software scelto su suggerimento del *Proponente* per la creazione dei file SVG$_G$.

6.4.2 Framework

Come emerso dalla riunione con il proponente (vedi documento allegato *Verbale18-12-2014_v2.0.pdf*), è stato scelto di utilizzare *MeteorJS* come framework open-source$_G$ per JavaScript$_G$. All'interno questo framework integra il database MongoDB$_G$.

6.4.3 Linee Guida

La linea guida grafica che si è scelto di utilizzare nella realizzazione del progetto Premi è quella ispirata da Google del Material Design$_G$. Questo in accordo con i desideri del Proponente Zucchetti S.p.a..

6.5 Strmenti di codifica

6.5.1 Stesura

Per lo sviluppo del progetto si è scelto di utilizzare L'IDE JetBrains WebStorm 7.0.3. Questo strumento nasce come IDE per applicazioni web, quindi basate su HTML$_G$, CSS$_G$ e JavaScript$_G$. E' stato scelto principalmente perchè supporta nativamente lo sviluppo su MeteorJS$_G$ e integra git$_G$ per semplificare i commit$_G$ sul repository$_G$. Ogni membro del gruppo ha installato sul suo computer la versione per studenti dell'IDE che è disponibile in licenza gratuita per un anno associando il proprio account alla e-mail universitaria personale;

6.5.2 Verifica

Per lo svolgimento del processo di verifica faremo uso dei seguenti strumenti:

- Strumenti W3C$_G$ (www.w3.org) per la validazione:
  - **validatore HTML5**$_G$ (http://validator.w3.org);
  - **validatore CSS**$_G$ (http://jigsaw.w3.org/css-validator/).
- Strumenti per debugging$_G$ HTML$_G$, CSS$_G$ e JavaScript$_G$ messi a disposizione dai vari browser$_G$:
  - **Chrome Developer Tools** (https://developers.google.com/chrome-developer-tools);
  - **Firebug** (http://getfirebug.com/).
- **JSLint** Ambiente di test (http://www.junit.org): tool per la validazione di codice JavaScript$_G$;
- **JUnit** (http://www.junit.org): semplice framework per eseguire test ripetibili;
- **BrowserStack** (http://www.browserstack.com/): per eseguire il test comparato sui vari browser$_G$;
- **WebStorm** (https://www.jetbrains.com/webstorm/): IDE JavaScript$_G$ scelto come ambiente di sviluppo;
- **Jasmine** (http://jasmine.github.io/): framework per testare codice Javascript;
- **Velocity** (http://www.meteortesting.com/chapter/velocity): framework ufficiale per i test su MeteorJS.

Il gruppo dovrà inoltre avere a disposizione i seguenti sistemi operativi:

6.5.3 Ambiente Desktop

I seguenti sistemi operativi:

- Microsoft Windows XP
- Microsoft Windows 8
- Linux Ubuntu 14.04
- Apple MacOs

I seguenti browser:

- Microsoft Internet Explorer 8
- Microsoft Internet Explorer 11
- Apple Safari
- Mozilla Firefox
- Google Chrome

Le seguenti risoluzioni di monitor:

- 1024x764
- 1440x960
- 1920x1080

6.5.4 Ambiente Mobile

I seguenti sistemi operativi:

- Android
- iOS
- Windows Phone 8.1

6.5.5 Ambiente Server

I seguenti sistemi operativi:

- Ubuntu Server 14.04