NetBreak

Progetto API Market

Norme di Progetto

Informazioni sul documento

Nome del documento	NormeDiProgetto 4_0_0.pdf
Data di creazione	06 Dicembre 2016
Ultima modifica	08 Giugno 2017
Versione	4.0.0
Stato	Approvato
Redatto da	Marco Casagrande Dan Serbanoiu Davide Scarparo Alberto Nicolè
Verificato da	Andrea Scalabrin Nicolò Scapin
Approvato da	Nicolò Scapin
Uso	Interno
Distribuzione	NetBreak
Destinato a	Prof. Tullio Vardanega, Prof. Riccardo Cardin, NetBreak
Email di riferimento	netbreakswe@gmail.com

Abstract

Documento contenente le norme di progetto che il gruppo NetBreak seguirà durante tutte le fasi di realizzazione del prodotto API Market.

Changelog

Versione	Data	Autore	Ruolo	Descrizione
4.0.0	2017-06-08	Nicolò Scapin	Responsabile	Approvazione del documento
3.1.0	2017-06-06	Davide Scarparo	Verificatore	Verifica del documento
3.0.3	2017-06-04	Andrea Scalabrin	Amministratore	Ampliamento sezione #4: Consuntivazione, Gestione dei task, Gestione delle infrastrutture
3.0.2	2017-06-02	Andrea Scalabrin	Amministratore	Aggiunte alla sezione #3: Ampliate verifica e validazione
3.0.1	2017-06-01	Andrea Scalabrin	Amministratore	Aggiunte alla sezione #2: Testing. Ampliato processo di fornitura: Pianificazione, preventivazione, preparazione al collaudo.
3.0.0	2017-03-25	Davide Scarparo	Responsabile	Approvazione del documento
2.1.0	2017-03-23	Marco Casagrande	Verificatore	Verifica del documento
2.0.3	2017-03-21	Nicolò Scapin	Amministratore	Ampliata in profondità sezione #2.2.2: "Progettazione"
2.0.2	2017-03-19	Alberto Nicolè	Amministratore	Ampliata in profondità sezione #2.2.5: "Strumenti utilizzati"
2.0.1	2017-03-16	Nicolò Scapin	Amministratore	Ampliata in profondità sezione #4.3.2: "Google Drive"
2.0.0	2017-02-20	Davide Scarparo	Responsabile	Approvazione del documento
1.2.0	2017-02-16	Andrea Scalabrin	Verificatore	Verifica del documento
1.1.3	2017-02-12	Alberto Nicolè	Responsabile	Ristrutturazione del paragrafo #4.2.3.3: ""
1.1.2	2017-02-10	Dan Serbanoiu	Amministratore	Ampliata sotto-sezione #3.3: "Validazione"
1.1.1	2017-02-06	Marco Casagrande	Amministratore	Correzioni secondo verifica
1.1.0	2017-02-04	Nicolò Scapin	Verificatore	Verifica del documento

Versione	Data	Autore	Ruolo	Descrizione
1.0.2	2017-01-28	Davide Scarparo	Responsabile	Ampliata sezione #3: "Processi di supporto"
1.0.1	2017-01-25	Alberto Nicolè	Responsabile	Ampliate sezioni #2: "Processi primari"
1.0.0	2016-12-31	Andrea Scalabrin	Responsabile	Approvazione del documento
0.4.0	2016-12-30	Davide Scarparo	Verificatore	Correzioni varie secondo verifica
0.3.0	2016-12-28	Davide Scarparo	Verificatore	Verifica paragrafo #4.2.3: "Analista"
0.2.2	2016-12-26	Nicolò Scapin	Amministratore	Aggiunta strumento per diagrammi di Gantt
0.2.1	2016-12-24	Nicolò Scapin	Amministratore	Aggiunta paragrafo #4.2.3: "Analista"
0.2.0	2016-12-20	Alberto Nicolè	Verificatore	Verifica del documento
0.1.1	2016-12-17	Marco Casagrande	Amministratore	Correzioni varie secondo verifica
0.1.0	2016-12-14	Davide Scarparo	Verificatore	Prima verifica del documento
0.0.6	2016-12-12	Nicolò Scapin	Amministratore	Completamento sezione #3: "Processi di supporto"
0.0.5	2016-12-12	Nicolò Scapin	Amministratore	Stesura iniziale sezione #4: "Processi organizzativi"
0.0.4	2016-12-11	Nicolò Scapin	Amministratore	Stesura della sezione #3: "Processi di supporto"
0.0.3	2016-12-10	Marco Casagrande	Amministratore	Stesura della sezione #2: "Processi primari"
0.0.2	2016-12-06	Nicolò Scapin	Amministratore	Stesura della sezione #1: "Introduzione"
0.0.1	2016-12-06	Andrea Scalabrin	Responsabile	Creazione template

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo del presente documento è la definizione di norme e regolamenti interni che il *team*$_G$ *NetBreak* seguirà scrupolosamente al fine di produrre materiale uniforme ed omogeneo. Il documento descrive accuratamente gli strumenti e le convenzioni adottate per la stesura dei vari documenti di progetto, le tecniche di revisione degli stessi, la definizione dei ruoli e la loro suddivisione, gli ambienti utilizzati nella fase di sviluppo del prodotto, le modalità di comunicazione interne ed esterne al gruppo.

1.2 Scopo del prodotto

Lo scopo del prodotto è la realizzazione di un *API Market*$_G$ per l'acquisto e la vendita di *microservizi*$_G$. Il sistema offrirà la possibilità di registrare nuove *API*$_G$ per la vendita, permetterà la consultazione e la ricerca di API ai potenziali acquirenti, gestendo i permessi di accesso ed utilizzo tramite creazione e controllo di relative *API key*$_G$. Il sistema, oltre alla web app stessa, sarà corredato di un *API Gateway*$_G$ per la gestione delle richieste e il controllo delle chiavi, e fornirà funzionalità avanzate di statistiche per il gestore della piattaforma e per i fornitori dei microservizi.

1.3 Riferimenti normativi

- **ISO 8601** (Rappresentazione di date e orari)
  `https://it.wikipedia.org/wiki/ISO_8601`

1.4 Riferimenti informativi

- PianoDiProgetto 4_0_0.pdf
- PianoDiQualifica 4_0_0.pdf
- **Amministrazione di Progetto**
  `http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L05.pdf`

1.5 Glossario

Per semplificare la consultazione e disambiguare alcune terminologie tecniche, le voci indicate con la lettera *G* a pedice sono descritte approfonditamente nel documento Glossario 3_0_0.pdf e specificate solo alla prima occorrenza all'interno del suddetto documento.

2 Processi primari

2.1 Fornitura

2.1.1 Studio di Fattibilità

Questa fase vedrà la realizzazione di un documento redatto a partire da ciò che emergerà dalle riunioni, alle quali è richiesta la partecipazione dell'intero team. Ciò consentirà di scegliere di comune accordo il capitolato più adatto. Durante tali riunioni, verrà stilata una lista di pro e contro per ogni capitolato d'appalto proposto.
Successivamente, gli *Analisti* avranno l'incarico di effettuare la stesura degli studi per ogni singolo capitolato. Il documento prodotto sarà strutturato nel seguente modo:

- **Descrizione:** introduzione generale del capitolato, nel quale si evidenziano le funzionalità minime richieste per il prodotto finale;
- **Dominio applicativo:** descrizione del bacino di utenza e dell'ambito in cui il prodotto sarà utilizzato;
- **Tecnologie:** breve elenco delle tecnologie di interesse, fondamentali nella realizzazione del prodotto richiesto;
- **Aspetti critici:** elenco delle problematiche, dei rischi e delle difficoltà che potrebbero nascere in fase di sviluppo del prodotto;
- **Considerazioni conclusive:** valutazione finale e soggettiva del gruppo, che fornisce le motivazioni per cui è stato scelto oppure scartato il capitolato in analisi.

2.1.2 Pianificazione

La pianificazione è l'attività che gestisce le scadenze temporali del progetto e il documento dedicato a questa attività è il *Piano di Progetto*. Una corretta pianificazione rispetta i seguenti vincoli:

- La pianificazione non deve superare il budget o il numero di ore previste;
- La pianificazione deve tener conto di aventuali rischi associati allo sviluppo del progetto;
- La pianificazione non deve superare la data di scadenza concordata con il cliente/committente.

2.1.3 Preventivazione

La preventivazione è l'attività che stima i costi e si basa sull'*Analisi dei Requisiti*.
Tale attività viene eseguita prima di partecipare alla gara d'appalto e i risultati sono riportati nel *Piano di Progetto*. Una corretta preventivazione rispetta i seguenti vincoli:

- Il preventivo deve basarsi sui requisiti obbligatori e opzionali riportati nell'*Analisi dei Requisiti*;
- Il preventivo deve indicare le ore esatte di ogni ruolo;
- Il preventivo deve seguire il costo imposto dal proponente.

2.1.4 Preparazione al collaudo

La preparazione al collaudo è la fase precedente alla consegna del prodotto. Al collaudo sono presenti tutti gli stakeholders, in particolare il proponente e il committente.
Affinchè il collaudo avvenga con successo, occorre rispettare i seguenti vincoli:

- Tutti i requisiti indicati come obbligatori nel documento Analisi dei Requisiti 3_0_0.pdf devono essere soddisfatti;
- Tutti i test di validazione previsti devono essere soddisfatti;
- I requisiti indicati come desiderabili o opzionali, ma richiesti espressamente dal proponente, devono essere soddisfatti;
- Devono essere predisposti dei test specifici che dimostrino il corretto funzionamento del prodotto finale.

2.2 Sviluppo

2.2.1 Analisi dei Requisiti

L'attività di analisi dei requisiti dovrà essere successiva all'attività di Studio di Fattibilità. Essa, infatti, analizzerà, in modo quanto più accurato possibile, i requisiti necessari, per ciascun ambito del progetto scelto.
Inoltre, verranno stilati i casi d'uso del prodotto, corredati da un'opportuna descrizione ed analisi.
La struttura del documento prodotto sarà organizzata nei seguenti punti:

- **Casi d'uso**;
- **Requisiti progettuali**.

2.2.1.1 Casi d'uso

2.2.1.1.1 Nomenclatura

La scelta del nome per i casi d'uso avverrà secondo la seguente sintassi:

UC[Codice categoria].[Codice progressivo]

dove:

- **Codice categoria**: identifica il codice entro cui lo specifico caso d'uso viene raggruppato. Può essere organizzato in ulteriori sottocategorie oppure omesso;
- **Codice progressivo**: identificativo dello specifico caso d'uso.

2.2.1.1.2 Struttura

L'analisi di ciascun caso d'uso dovrà essere strutturata come segue, avendo cura di mantenere l'ordine indicato:

- **Sigla e nome**: identifica il caso d'uso. Va indicato nel titolo del paragrafo corrispondente;
- **Diagramma**: immagine del diagramma UML_G per il caso d'uso;
- **Attori**: descrizione degli attori coinvolti;
- **Descrizione**: breve descrizione del caso d'uso;
- **Pre-Condizioni**: descrive lo state iniziale per il caso d'uso;
- **Post-Condizioni**: descrive lo stato finale che deve valere al termine dello scenario descritto;
- **Scenario Principale**: analisi completa del flusso di esecuzione principale del caso d'uso;
- **Scenari Alternativi**: se presenti, elenca e descrive gli eventuali scenari alternativi per il caso d'uso.

2.2.1.2 Requisiti progettuali

Gli *Analisti* hanno il compito di produrre il documento AnalisiDeiRequisiti 3_0_0.pdf contenente un elenco di requisiti, con annesse peculiarità richieste ed informazioni sulla loro tipologia e rilevanza.

2.2.1.2.1 Nomenclatura

La scelta del nome per i requisiti progettuali avverrà secondo la seguente procedura:

R[Tipologia][Rilevanza][Codice]

Tipologia: può assumere uno dei seguenti valori:

- **V:** requisito di vincolo;
- **F:** requisito di funzionalità;
- **Q:** requisito di qualità;
- **P:** requisito prestazionale.

Rilevanza: può assumere uno dei seguenti valori, elencati in ordine di importanza:

- **O:** requisito obbligatorio;
- **D:** requisito desiderabile;
- **F:** requisito facoltativo.

Codice: assume un numero sequenziale e univoco, necessario a catalogare e riconoscere ogni requisito progettuale.

2.2.1.2.2 Struttura

L'analisi di ciascun requisito dovrà essere strutturata come segue, avendo cura di mantenere l'ordine indicato:

- **Descrizione:** descrizione sintetica e concisa del requisito;
- **Fonte:** descrive la fonte del requisito, ovvero da chi è stato sollevato e in che ambito. Può assumere i seguenti valori: capitolato, caso d'uso, interno.

2.2.2 Progettazione

La progettazione è l'attività fondamentale in cui viene realizzata un'astrazione di quella che diverrà la struttura software del prodotto. È successiva alla produzione di una completa Analisi dei Requisiti, in quanto da essa vengono tracciate le linee guida per la progettazione.
I documenti risultanti da questa attività fungeranno, poi, da percorso per la produzione del software vero e proprio. Una progettazione svolta al meglio è utile per un'attività di codifica ottimale.
I *Progettisti* sono responsabili delle attività di progettazione, e sono tenuti ad avere:

- Una profonda conoscenza di tutto ciò che riguarda il processo di sviluppo del software;
- La capacità di saper anticipare i cambiamenti;
- Una forte inventiva per riuscire a trovare una soluzione progettuale accettabile anche in mancanza di una metodologia che sia sufficientemente espressiva;
- La capacità di individuare con rapidità e sicurezza le soluzioni più opportune.

2.2.2.1 Obiettivi

Durante questa attività, il team si prefigge i seguenti obiettivi:

- Fornire una visione macroscopica del percorso da seguire nella codifica software;
- Acquisire una profonda conoscenza di tutto ciò che riguarda lo sviluppo del software;
- Realizzare un prodotto rispettando gli standard prefissati nelle attività di Studio di Fattibilità e Analisi dei Requisiti;
- Flessibilità del prodotto, per poter far fronte, in modo rapido, a repentine ed eventuali modifiche rilevanti, pur non condizionando il lavoro pregresso;
- Soddisfare le richieste del committente.

2.2.2.2 Diagrammi

Per la progettazione, il gruppo ha deciso di utilizzare quattro tipologie di diagrammi UML:

- ***Diagrammi dei package$_G$:*** raggruppano le classi in un'unità di alto livello;
- ***Diagrammi delle classi$_G$:*** descrivono le caratteristiche delle singole unità;
- ***Diagrammi di attività$_G$:*** illustrano il flusso di operazioni relativo ad un'attività che è possibile svolgere sul prodotto. Possono essere utilizzati per descrivere la logica di un algoritmo specifico;
- ***Diagrammi di sequenza$_G$:*** descrivono le sequenze di azioni dove tutte le decisioni sono già state effettuate, quindi sono assenti scelte dell'utente e flussi alternativi.

2.2.2.3 Specifica Tecnica

I *Progettisti* devono definire la struttura ad alto livello dell'architettura del sistema e delle singole componenti, raccogliendo il tutto nel documento Specifica Tecnica 3_0_0.pdf.
Inoltre, hanno il compito di definire i test di integrazione tra le varie componenti, che verranno inseriti in appendice al documento Piano di Qualifica 4_0_0.pdf.
Il documento di *Specifica Tecnica* conterrà i seguenti argomenti:

- **Diagrammi UML:** diagrammi dei packages, diagrammi delle classi, diagrammi di sequenza e diagrammi di attività.
- **Design patterns:** i *Progettisti* devono fornire una descrizione dei design patterns adottati nella definizione dell'architettura. Questa descrizione dovrà essere accompagnata da un diagramma UML, che ne esemplifichi il funzionamento, e dalle motivazioni che hanno portato all'adozione di tale pattern;
- **Tracciamento delle componenti:** ogni componente dovrà essere tracciato ed associato ad almeno un requisito. Così facendo, si potrà essere certi che tutti i requisiti vengano soddisfatti, e che ogni componente presente nell'architettura soddisfi almeno un requisito. Tale tracciamento dovrà essere effettuato tramite *NetBreakDB*, che si occupa di generare in modo automatico le relative tabelle.
- **Test d'integrazione:** i *Progettisti* devono definire delle strategie di verifica per poter dimostrare la corretta integrazione tra le varie componenti definite.

2.2.2.4 Definizione di Prodotto

I *Progettisti*, a partire dal documento di *Specifica Tecnica*, hanno il compito di produrre la *Definizione di Prodotto*, nella quale 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, non fornire alcun grado di libertà ai *Programmatori* .
Parallelamente alla progettazione di dettaglio dei componenti software, si dovranno prevedere e progettare i relativi test di unità.
Il documento di *Definizio di Prodotto* conterrà i seguenti argomenti:

- **Diagrammi UML:** diagrammi dei packages, diagrammi delle classi, diagrammi di sequenza.
- **Definizione delle classi:** ogni classe progettata viene descritta più in dettaglio, fornendo una descrizione più approfondita dello scopo, delle sue funzionalità e del suo funzionamento. Per ogni classe, dovranno essere anche definiti i vari metodi e attributi che la caratterizzano;
- **Tracciamento delle classi:** ogni classe deve essere tracciata ed associata ad almeno un requisito, così facendo è possibile avere la certezza che tutti i requisiti vengano soddisfatti, e che ogni classe presente nell'architettura soddisfi almeno un requisito.
  Questo tracciamento dev'essere effettuato tramite *DocumentsDB*, che si occupa di generare in modo automatico le tabelle di tracciamento.
- **Test di unità:** i *Progettisti* devono definire le strategie di verifica delle varie classi, in modo che durante l'attività di codifica sia possibile verificare che la classe si comporti in modo corretto.

2.2.2.5 Norme progettuali

2.2.2.5.1 Tecniche di modularizzazione

Prima di iniziare l'attività di progettazione, i *Progettisti* hanno bisogno di definire delle tecniche di modularizzazione, affinchè l'architettura software progettata sia di elevata qualità.
Essi devono necessariamente adottare delle tecniche che consentano la scomposizione del sistema in moduli e definire una descrizione precisa della struttura modulare e delle relazioni che esistono tra i singoli moduli.
La modularizzazione porterà i seguenti vantaggi:

- Semplificazione della verifica della correttezza semantica e nella correzione di errori;
- Leggibilità del codice;
- Riusabilità del software;
- Possibilità di realizzazione di prototipi;
- Semplificazione dell'attività di manutenzione.

2.2.3 Codifica

L'attività di codifica si prefigge la realizzazione pratica del codice dell'applicativo richiesto. I *Programmatori* sono i principali responsabili di questa attività, i quali, tenendo conto degli stadi precedenti e seguendo nel dettaglio quanto precedentemente stilato, realizzano un prodotto efficace.
Ogni *Programmatore* deve essere, come requisito prioritario, strettamente ancorato alle linee guida stabilite.

2.2.3.1 Best practices

L'attività di codifica richiede di essere molto scrupolosi, attenendosi alle linee guida definite all'interno del presente documento.
Esse compongono gli standard di codifica per la realizzazione del progetto. I seguenti standard hanno il compito di:

- Fornire uno strumento per garantire una più agevole cooperazione tra i diversi sviluppatori;
- Favorire la creazione di un codice valido e ben formato;
- Favorire la realizzazione di un software di qualità;
- Rendere più agevole la lettura e la modifica del codice a tutti i componenti del team.

Le regole stilistiche del codice riguardano le parti non prettamente implementative, ma di ausilio ai *Programmatori*, al fine di permettere una miglior cooperazione e comprensione del codice stesso.
Le tecniche di scrittura del codice, descritte nel presente documento, sono suddivise in tre categorie:

- **Impostazione**;
- **Nomenclatura**;
- **Commenti**.

2.2.3.1.1 Impostazione

L'impostazione stilistica del documento consente una più semplice comprensione e lettura del codice, e pertanto è importante fornire una logica comune. Di seguito sono elencate le principali convenzioni relative alla formattazione del codice:

- I blocchi di codice vanno indentati tramite i rientri standard, evitando spaziature singole. Ogni porzione di codice interna ad un altra dev'essere rigorosamente indentata a sua volta;
- Utilizzare gli spazi per separare gli operatori, ove possibile. Aggiungere spazi che aumentino la leggibilità del codice in tutte le situazioni dove ciò sia permesso, a patto che il funzionamento non venga compromesso;
- Dividere il codice in più moduli e non utilizzare un unico file di ingenti dimensioni e di difficile modifica. Ciò è un requisito fondamentale per un corretto e miglior utilizzo dei sistemi di versionamento.

2.2.3.1.2 Nomenclatura

Di seguito, sono elencati gli standard di nomenclatura utilizzati per il codice che verrà prodotto. Tale aspetto è fondamentale per una miglior comprensione del codice:

- Assegnare nomi univoci ai costrutti, evitando possibili ambiguità dovute a nomi similari. I costrutti devono avere nomi quanto più significativi possibili, anche se utilizzati in porzioni marginali. Le variabili formate da un solo carattere sono, invece, consentite qualora si tratti di iterazioni circoscritte;
- Scegliere nomi descrittivi per file e cartelle. È importante che indichino, nel modo più accurato possibile, il proprio contenuto;

- Accertarsi che ogni parola utilizzata sia scritta nella corretta forma della lingua a cui appartiene;
- Nella scrittura di nomi composti utilizzare la forma *Camel Case*$_G$, ottenuta scrivendo con iniziale minuscola la prima parola di un nome composto, seguito dalle successive parole, ognuna con iniziale sempre maiuscola;
- Nella descrizione di funzioni, utilizzare come prima voce il verbo dell'azione svolta, seguito dal nome dell'azione o dell'oggetto su cui viene eseguita;
- Accertarsi che non vi sia ambiguità nelle abbreviazioni utilizzate, avendo cura di verificare che non possano venir impiegate due abbreviazioni simili per funzionalità differenti.

2.2.3.1.3 Commenti

All'interno del codice sorgente è di fondamentale importanza la presenza di commenti, al fine di facilitare la comprensione del flusso logico.
Di seguito, vengono elencati alcuni metodi di inserimento dei commenti:

- Produrre e/o mantenere aggiornata un'intestazione per ciascun file. Essa deve sempre avere una breve descrizione di ciò che è incluso nello specifico file;
- Indicare pre e post-condizione di ogni funzionalità di rilievo, in modo da favorire un riutilizzo di eventuale codice già scritto. È, inoltre, consigliato introdurre una breve descrizione concisa riguardante la funzionalità;
- Ogni qualvolta si modifica il codice, occorre mantenere sempre aggiornati i relativi commenti;
- All'inizio di ogni funzione o procedura, è utile fornire commenti che ne indichino le limitazioni, i presupposti e lo scopo;
- Evitare l'aggiunta di commenti alla fine di una riga di codice, poichè può rendere più difficoltosa la lettura del codice. Tuttavia, questo tipo di commento è accettabile in caso di annotazione di dichiarazioni di variabili;
- Evitare commenti confusi come righe intere di asterischi, bensì si raccomanda di utilizzare spazi vuoti per separare i commenti dal codice;
- Quando si scrivono commenti, utilizzare frasi di senso compiuto. La funzione dei commenti consiste nel chiarire il significato del codice senza aggiungere alcun tipo di ambiguità;
- Inserire commenti in fase di scrittura del codice, in quanto ciò potrebbe non essere possibile in un secondo momento;
- Evitare commenti superflui o inappropriati, come annotazioni umoristiche;
- Commentare tutto ciò che si ritiene poco chiaro all'interno del codice;
- Utilizzare sempre i commenti nel caso di codice relativo a correzioni di errori e a potenziali soluzioni, al fine di evitare problemi ricorrenti e reintrodurre errori già corretti in precedenza;
- Aggiungere commenti al codice costituito da cicli e diramazioni logiche, per facilitarne la comprensione;
- Creare commenti adottando uno stile uniforme e una struttura e una punteggiatura coerenti.

2.2.4 Testing

2.2.4.1 Descrizione

L'attività di testing ha lo scopo di verificare che ogni funzione codificata rispetti la progettazione e che i risultati in output, per ciascuna componente, soddisfino le aspettative.

2.2.4.2 Strategie generali

Le strategie principali che verranno intraprese in fase di testing sono le seguenti:

- Ridurre tempi e costi dell'attività di testing aumentando il livello di automatizzazione quanto più possibile;
- Seguire le linee guida dei *Progettisti* riguardo i risultati attesi e lo svolgersi dei test.

2.2.4.3 Obiettivi attesi

Il team si impegna per soddisfare i seguenti obiettivi:

- I test di integrazione devono essere quanto più possibile concordi alle componenti integrate all'interno del software prodotto;
- I requisiti obbligatori pattuiti in fase di definizione contrattuale verranno testati in modo approfondito.

2.2.5 Strumenti utilizzati

2.2.5.1 Astah

Per la modellazione dei diagrammi dei casi d'uso in linguaggio UML, la scelta è ricaduta su ***Astah*$_G$**, editor multi-piattaforma e gratuito. Il suo punto forte rispetto ad altre soluzioni è la semplicità di utilizzo.
Tra le funzionalità interessanti ai fini del team, c'è la possibilità di esportare i diagrammi creati direttamente in un file immagine con estensione *.png*.
Il software Astah sarà utilizzato per la creazione di tutti i diagrammi necessari richiesti durante le attività di progettazione ed analisi dei requisiti.

2.2.5.2 Smartsheet

Per la realizzazione dei *diagrammi di Gantt*$_G$ richiesti dall'attività di pianificazione, si è scelto di utilizzare lo strumento ***Smartsheet*$_G$**, il quale consente di creare diagrammi secondo le attività pianificate per uno specifico periodo di tempo, indicato dall'utente.
Smartsheet è uno strumento gratuito e intuitivo, fruibile tramite interfaccia web. Le funzioni offerte sono ampie e immediate, ed è possibile personalizzare i diagrammi con una formattazione condizionale per evidenziare attività, punti critici o *milestone*$_G$.
Smartsheet semplifica notevolmente la collaborazione in tempo reale e permette a diverse persone di rimanere aggiornate in tempo reale sugli sviluppi del progetto. È risultato molto utile per la creazione dei grafici presenti nel documento PianoDiProgetto 4_0_0.pdf.

2.2.5.3 NetBreakDB

Per il tracciamento dei documenti, preponderante specialmente nei periodi di Analisi dei Requisiti e Progettazione, la scelta del team è ricaduta sull'utilizzo di uno strumento dedicato, per permettere una più agevole gestione.
Il software utilizzato è un fork di un progetto esistente, *PragmaDB*, creato da un gruppo di

studenti degli scorsi anni e modificato secondo le esigenze del team *NetBreak*.
NetBreakDB permette la gestione di casi d'uso, attori, componenti, metriche, fonti e il calcolo automatizzato dell'indice Gulpease, effettuato direttamente sui documenti contenuti in una repository di *GitHub*$_G$. I risultati dell'indice vengono visualizzati in una tabella secondo questo ordine:

Nome Documento | Valore Indice Gulpease | Esito

L'applicativo è in grado di generare codice LaTeX a seguito di un'indicizzazione dei requisiti all'interno del database, e permette di esportare le voci necessarie al tracciamento in maniera automatizzata, rendendo il lavoro più uniforme, ordinato e semplificato.

2.2.5.4 Docker

***Docker*$_G$** è un progetto open source per gli sviluppatori e gli amministratori di sistema, che permette di automatizzare la distribuzione di applicazioni all'interno di contenitori software su macchine virtuali e nel cloud.
Il principio su cui si basa, promuove lo sviluppo agile, il quale garantisce maggiore efficienza e flessibilità alle iniziative cloud e di modernizzazione delle applicazioni. Docker è un contenitore che permette di integrare applicazioni o parti di esse, e fornisce tutto il necessario per l'esecuzione ed installazione su un server.
Ciò garantisce che il software funzionerà sempre allo stesso modo, indipendentemente dall'ambiente in cui verrà eseguito.
Questo strumento permette di creare software più performanti, accelerare lo sviluppo ed eliminare possibili problemi dovuti all'esecuzione su macchine diverse. Il contenitore isola l'applicazione dai livelli sottostanti della struttura, in modo da garantire protezione all'applicazione.

2.2.5.5 SwaggerHub

***SwaggerHub*$_G$** è uno strumento software online, sviluppato da SmartBear Software, per la produzione di API condivise tra un massimo di 25 sviluppatori. Esso si prefigge di accompagnare lo sviluppo durante l'intero ciclo di vita del software, in modo incrementale.
SwaggerHub supporta numerosi differenti linguaggi di programmazione e permette l'integrazione con servizi di controllo della versione come GitHub.
Una caratteristica rilevante è la creazione in tempo reale della documentazione relativa all'API che si sta sviluppando.
La manutenzione è facile ed è possibile distribuire l'API prodotta direttamente su piattaforme come *AWS*$_G$ e *Microsoft Azure*$_G$.

2.2.5.6 Sublime Text

***Sublime Text*$_G$** è l'editor scelto dal team per la scrittura del codice. Esso è caratterizzato da un'interfaccia intuitiva, un'ampia gamma di features innovative e ottime prestazioni.
Questo editor è compatibile con diversi sistemi operativi ed è personalizzabile in molti aspetti.
Una caratteristica peculiare è il *simultaneous editing*, ovvero la possibilità di modificare più porzioni di codice contemporaneamente, premendo il tasto *Shift* e posizionando il cursore nel punto esatto in cui si intende effettuare delle modifiche.
Le caratteristiche di personalizzazione riguardano i temi, l'interfaccia e le modalità di visione del codice.
Inoltre, l'auto-completamento del codice in base al linguaggio scelto e il salvataggio automatico sono altre funzionalità che hanno portato il gruppo a scegliere questo strumento.

3 Processi di supporto

3.1 Documentazione

In questa sezione del documento verrà spiegata la struttura, la classificazione e il metodo di revisione dei documenti del gruppo *NetBreak*. L'attività di documentazione è di supporto a tutte le altre attività, difatti ogni cosa deve essere tracciata e documentata. L'obiettivo prefissato è:

- Produrre documenti corretti grammaticalmente e sintatticamente;
- Leggibili dal più largo numero di persone e nel rispetto dell'indice di Gulpease;
- Avere un glossario comune per evitare ambiguità di nomenclatura tecnica;
- Produrre documenti di struttura uniforme.

3.1.1 Nomenclatura e versione del documento

Ogni documento deve rispettare una denominazione comune, il più possibile chiara, affinché venga individuato facilmente il nome e la versione del documento. Il team ha scelto di utilizzare la seguente nomenclatura:

NomeDelDocumento X_ Y_ Z.pdf

All'interno del nome, *X_ Y_ Z* indica la versione del documento nel seguente modo:

- ***X***: indica il numero di pubblicazioni del documento. Questo indice è incrementato esclusivamente dal *Responsabile di Progetto* in seguito alla sua approvazione finale. L'incremento di tale indice, azzera automaticamente gli indici Y e Z;
- ***Y***: indica il numero di verifiche e viene incrementato esclusivamente dai *Verificatori*. L'incremento di tale indice, azzera automaticamente l'indice Z;
- ***Z***: indica il numero di aggiornamenti minori effettuati prima o in seguito a una verifica o approvazione. Viene incrementato progressivamente e viene azzerato solo in seguito a una modifica degli indici X e Y.

Ogni modifica della versione del documento deve riflettersi nel changelog, nel nome del documento e nel frontespizio in corrispondenza della voce *Versione*.

3.1.2 Classificazione del documento

Per tutti i documenti redatti dal gruppo occorre sempre specificare una classe, che può essere:

- **Interna**: la consultazione è limitata al solo gruppo e comprende tutti i documenti riguardandi l'organizzazione del gruppo e le regole imposte ai membri e al team;
- **Esterna**: la consultazione è destinata ad enti esterni al gruppo, quindi a proponente e committente.

3.1.3 Ciclo di vita del documento

Un documento può trovarsi in uno dei seguenti stati:

- **Bozza**:documento in fase di stesura dal relativo redattore;
- **In attesa di verifica**: la stesura del documento è terminata e in attesa di verifica da parte del verificatore;
- **Approvato**: il documento è nella sua versione finale.

3.1.4 Struttura del documento

Ogni documento redatto dal gruppo avrà una struttura chiara e di facile comprensione per chiunque sia il lettore.
Una struttura semplice aiuta nella lettura e nell'individuare immediatamente l'argomento trattato nello specifico documento e l'identità di chi ha redatto, verificato e approvato il documento.
Di seguito, viene riportata la struttura utilizzata nei documenti:

3.1.4.1 Frontespizio

- Nome del gruppo;
- Nome del progetto;
- Logo del gruppo;
- Nome del documento;
- Data di creazione;
- Ultima modifica;
- Versione;
- Stato del documento;
- Nome e cognome del/i redattore/i;
- Nome e cognome del/dei *Verificatore/i*;
- Nome e cognome del *Responsabile di Progetto* che ha approvato il documento;
- Classificazione del documento;
- Distribuzione;
- Destinatari del documento;
- Email di riferimento;
- Abstract del documento.

3.1.4.2 Changelog

La seconda sezione del documento è una lista che raccoglie e tiene traccia di tutte le modifiche effettuate al documento, dalla sua creazione alla sua approvazione finale. Le modifiche sono indicate secondo questo schema:

- **Versione**: indica la versione del documento;
- **Data**: indica la data di modifica del documento;
- **Autore**: indica l'autore della modifica effettuata;
- **Ruolo**: indica il ruolo dell'autore che ha effettuato la modifica;
- **Descrizione**: indica le aggiunte e/o modifiche effettuate al documento.

3.1.4.3 Indice

Ogni documento, ad eccezione dei verbali, deve possedere un indice contenente tutte le sezioni presenti nel documento.
L'indice è ordinato nella sequenza in cui appaiono i capitoli, le sezioni e sottosezioni, e contiene il titolo di ognuna.
Se nel documento sono presenti immagini o tabelle, occorre indicarle con il relativo indice.

3.1.4.4 Intestazione e piè di pagina

Tutte le pagine del documento, ad eccezione della prima, hanno un'intestazione e un piè di pagina. L'intestazione è suddivisa in due parti:

- Nell'angolo sinistro, sono indicati il nome del progetto e del documento in questione;
- Nell'angolo destro, è presente il nome del capitolo contente la sezione che si sta consultando, ad esclusione del changelog.

Il piè di pagina, invece, è strutturato nel seguente modo:

- Nell'angolo sinistro, vengono indicati nome e indirizzo email ufficiale del gruppo;
- Nell'angolo destro, è presente una numerazione in numeri romani per le pagine relative a changelog e indice. Per tutte le altre pagine, invece, viene utilizzata una numerazione in numeri arabi, che indica il numero di pagina attuale e il numero totale di pagine del documento.

3.1.4.5 Norme tipografiche

Questa sezione del documento contiene i criteri riguardanti l'ortografia e la tipografia, utilizzate nel corso dello sviluppo della documentazione del progetto.

3.1.4.5.1 Stili di testo e punteggiatura

Stili:

- **Grassetto**: deve essere utilizzato per parole importanti, all'interno di frasi o elenchi puntati;
- *Corsivo*: deve essere utilizzato per:
  - Citazioni;
  - Parole del glossario (unitamente ad una G maiuscola in pedice) alla loro prima occorrenza nel documento di riferimento;
  - Riferimenti ai ruoli;
  - Riferimento all'intero gruppo.
- MAIUSCOLO: viene utilizzato unicamente per scrivere acronimi e macro LaTeX presenti nei documenti;
- MAIUSCOLETTO: viene utilizzato per i riferimenti ad altri documenti;
- LaTeX: viene usato il comando \LaTeX per ogni occorrenza del termine LaTeX.

Punteggiatura:

- Punteggiatura: non deve seguire spazi, ad eccezione del punto, del punto interrogativo e del punto esclamativo, che sono seguiti da uno spazio e lettera maiuscola.
- elenchi puntati: Ogni voce di un elenco puntato deve terminare con punto e virgola, ad eccezione dell'ultima che termina con un punto.

3.1.4.5.2 Formato data

All'interno di ogni documento, tutte le date seguiranno lo standard ***ISO 8601:2004***$_G$:

YYYY-MM-DD

Dove:

- **YYYY**: indica l'anno;
- **MM**: indica il mese;
- **DD**: indica il giorno.

3.1.4.5.3 Formato ora

All'interno di ogni documento, tutti gli orari seguiranno lo standard ***ISO 8601:2004***$_G$:

hh-mm

Dove:

- **hh**: l'ora;
- **MM**: indica il minuto.

3.1.5 Documenti da consegnare

3.1.5.1 Studio di Fattibilità

- **Classificazione**: interno;
- **Destinazione**: gruppo e committente;
- **Contenuto**: questo documento contiene lo studio effettuato dal gruppo *NetBreak* su tutti i capitolati e, per ognuno di essi, le motivazioni che hanno portato alla relativa scelta o rifiuto.

3.1.5.2 Norme di Progetto

- **Classificazione**: interno;
- **Destinazione**: gruppo e committente;
- **Contenuto**: lo scopo del documento è raccogliere tutte le convenzioni, gli strumenti e le regole che il gruppo *NetBreak* adotterà durante l'intera realizzazione del progetto.

3.1.5.3 Analisi dei Requisiti

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento si prefigge lo scopo di dare una visione generale dei requisiti essenziali del progetto e dei relativi casi d'uso.

3.1.5.4 Piano di Progetto

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento descrive come il gruppo *NetBreak* ha impiegato tempo e risorse umane, ma anche la pianificazione delle stesse, per le attività future previste per la realizzazione del prodotto richiesto dal progetto.

3.1.5.5 Piano di Qualifica

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento descrive come il gruppo *NetBreak* intende raggiungere gli obiettivi di qualità prefissati.

3.1.5.6 Glossario

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento ha lo scopo di fornire una definizione di tutti i termini tecnici e acronimi, al fine di rendere la lettura comprensibile a tutti i destinatari della documentazione fornita.

3.1.5.7 Specifica Tecnica

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: lo scopo di questo documento è fornire una visione generale del prodotto e dei suoi requisiti. In particolare, viene mostrata una progettazione ad alto livello, basata su diagrammi dei package, per la descrizione delle componenti, diagrammi di sequenza e di attività, per la descrizione degli scenari d'uso.
  Inoltre, vengono elencate le tecnologie che si intendono utilizzare per la realizzazione del progetto.

3.1.5.8 Definizione di Prodotto

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento ha lo scopo di fornire una progettazione dettagliata del prodotto. In esso vengono forniti tutti i dettagli implementativi del prodotto, fondamentali in fase di codifica, che comprendono i diagrammi delle classi e i relativi metodi.

3.1.5.9 Manuale Utente

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento ha lo scopo di fornire all'utente una guida dettagliata riguardo tutte le funzionalità offerte dal prodotto realizzato.

3.1.5.10 Manuale Manutentore

- **Classificazione**: esterno;
- **Destinazione**: gruppo, committente e proponente;
- **Contenuto**: questo documento ha lo scopo di fornire agli sviluppatori e manutentori dell'applicazione, una guida dettagliata riguardo tutte le specifiche tecniche e tecnologiche per il mantenimento del prodotto.

3.1.6 Strumenti utilizzati

Per la stesura dei documenti è stato scelto di utilizzare LaTeX, un linguaggio di markup utilizzato per la stesura di documenti scientifici.
LaTeXè un linguaggio semplice e modulare, in grado di evitare possibili conflitti provenienti dall'utilizzo di software e piattaforme differenti.

3.1.6.1 TeXstudio

In seguito a un breve periodo di test, la scelta del team per l'editor di documenti *LaTeX*$_G$ è ricaduta su ***TeXstudio*$_G$**. Esso è un noto fork di un altrettanto famoso editor, *TexMaker*$_G$, ma con la peculiarità di essere completamente gratuito, open source e disponibile per i più diffusi sistemi operativi.
L'editor consente un semplice utilizzo dei file *.tex* e una comoda gestione di inclusione e accorpamento di documenti. Tale caratteristica viene incontro alle esigenze del gruppo per via della predisposizione alla modularità dei documenti richiesti dal progetto.
L'editor scelto, inoltre, possiede numerose features rilevanti, quali l'autocompletamento e il controllo della sintassi e dell'ortografia, e l'anteprima in PDF incorporata. Per questi motivi, TeXstudio sarà il principale strumento tramite cui verranno redatti tutti i documenti.

3.2 Verifica

Il processo di verifica ha lo scopo di controllare ed assicurare che documentazione e software prodotti rispecchino quanto previsto dai requisiti.
In particolare, la verifica serve a stabilire che il prodotto rispetti le specifiche, mentre la validazione accerta che le specifiche siano rispettate nel modo più consono.

3.2.1 Verifica documenti

3.2.1.1 Descrizione

L'attività di verifica dei documenti è svolta dai *Verificatori*. Essi hanno il compito di segnalare ogni incongruenza o errore, facendo presente eventuali punti non chiari, in modo tale da permetterne la correzione.
Ogni problema deve essere segnalato tramite gli strumenti di comunicazione e issue tracking internamente utilizzati, con una breve descrizione del problema e un riferimento alla posizione del documento da controllare.
In alternativa, il *Verificatore* può effettuare precise annotazioni nei documenti PDF resi disponibili per il controllo, da consegnare in seguito alla persona responsabile di effettuare le modifiche nel documento.
Qualsiasi dubbio aggiuntivo può essere chiarito tramite gli strumenti di comunicazione abitualmente utilizzati dal team.

3.2.1.2 Strategia generale

Si considerano, in aggiunta alle descrizioni e alle strategie descritte nella sottosezione corrente, le metodologie sottoriportate:

- Si effettua un controllo preliminare della documentazione correlata ed eventuali aggiornamenti che possano essere rilevanti in attività di verifica;
- Si valutano le *Norme di Progetto* in vigore;
- Si procede ad una correzione preliminare tramite controllo ortografico standard.

3.2.1.3 Obiettivi attesi

Gli obiettivi di qualità attesi dall'attività di verifica dei documenti sono:

- L'ottemperanza delle *Norme di Progetto* per l'ultima versione del documento durante il processo di verifica;
- L'eliminazione degli errori sintattici e ortografici;
- L'eliminazione di porzioni di difficile leggibilità e comprensione;
- La corrispondenza con la restante documentazione.

3.2.1.4 Indice Gulpease

Per verificare l'indice di leggibilità dei documenti, così come definito nel documento PianoDiQualifica 4_0_0.pdf, si è deciso di utilizzare uno strumento realizzato da un gruppo degli anni precedenti.
Il calcolo dell'indice Gulpease è effettuato direttamente attraverso la piattaforma *NetBreakDB*, la quale è in grado di analizzare i vari file *.tex* all'interno di una cartella e restituire il nome del documento, il valore dell'indice e un'indicazione se il valore è superiore al minimo prefissato.

Questo script è un ottimo strumento per automatizzare il lavoro, in quanto, dopo alcune modifiche, è in grado di esportare i risultati in codice LaTeX e semplificarne l'integrazione, indicando in quali documenti è necessario intervenire per migliorarne la leggibilità del testo.

3.2.1.5 Modalità di analisi

3.2.1.5.1 Analisi statica

L'analisi statica è applicata ai documenti di testo, e consiste nel trovare errori sintattici ed ortografici. Una prima verifica viene effettuata dai *Verificatori* e, successivamente, dal *Responsabile di Progetto*. Le due tecniche scelte sono la ***Formal Walkthrough*$_G$** e la ***Fagan Inspection*$_G$**.

- **Formal Walkthrough**: consiste nell'individuare quanti più errori di sintassi e di ortografia possibili, senza concentrarsi su particolari e specifici errori. Questa tecnica sarà utilizzata dai *Verificatori*, i quali annoteranno gli errori più frequenti in un'apposita lista, necessaria per la successiva tecnica di Fagan Inspection;
- **Fagan Inspection**: si basa su una lettura attenta dei documenti, basandosi sulla lista degli errori stilata durante la Formal Walkthrough. Questo processo acquisirà rilevanza in concomitanza con l'incremento della lista di possibili errori, stilata dai *Verificatori*.

3.2.1.6 Analisi dinamica

L'analisi dinamica è applicata esclusivamente sul software prodotto, in quanto consiste nell'effettuare dei test per verificare il corretto funzionamento dell'applicativo prodotto.

3.2.2 Verifica del codice

3.2.2.1 Descrizione

L'attività di verifica del codice è effettuata tramite analisi statica e dinamica. In particolare, l'analisi dinamica viene applicata al codice solamente, poichè richiede la compilazione dello stesso e la sua esecuzione.
Lo scopo di questa attività è che il software prodotto fornisca i risultati attesi.

3.2.2.2 Strategia generale

Si considerano, in aggiunta alle descrizioni e alle strategie descritte nella sottosezione corrente, le metodologie sottoriportate:

- Il software viene analizzato secondo le metriche stabilite e ogni discrepanza dai risultati attesi viene segnalata tempestivamente ai responsabili della porzione di codifica considerata.

3.2.2.3 Obiettivi attesi

Gli obiettivi di qualità attesi dall'attività di verifica del codice sono:

- I test effettuati devono essere quanto più estesi possibili, per coprire ogni caso d'uso del prodotto finale;
- I test devono avere l'apporto minimo da parte del *Verificatore*, ed essere quindi, quanto più automatizzati possibile.

3.2.2.4 Test

Il sistema adottato per testare l'operato prodotto, prende atto della struttura comune a tutto il software, decomponendo l'attività, a partire da elementi semplici fino a giungere ai test dell'intero sistema, seguendo opportune strategie di integrazione.

3.2.2.4.1 Test di unità

Questa tipologia di test è atta ad analizzare ogni piccola unità software singolarmente. Così facendo, si tende a minimizzare il numero di errori, a partire da ogni singola componente. Testando i singoli moduli, si rende necessaria la presenza di componenti fittizie per simulare parti di sistema incomplete e altrimenti non testabili singolarmente. Il codice utilizzato per denotare questa fase di test sarà:

TU[Identificativo test]

3.2.2.4.2 Test di integrazione

I test denominati di integrazione riguardano insiemi di moduli, che vengono testati una volta assemblati. Alternando passi di integrazione e di controllo, come in questo caso, significa necessariamente predisporre test su componenti ancora incomplete. Il codice utilizzato per denotare questa fase di test sarà:

TI[Identificativo test]

3.2.2.4.3 Test di sistema

I test di sistema valutano ogni caratteristica di qualità di prodotto nella sua completezza. Il prodotto, sottoposto a questa tipologia di test, si ritiene giunto ad una realizzazione definitiva. I test riguardano numerosi aspetti cruciali, e si possono riassumere nelle seguenti categorie:

- **Facility test**: vengono testate le funzionalità del prodotto, affinché svolgano correttamente le attività preposte;
- **Compatibility test**: si testa la compatibilità del prodotto con software di terze parti che interagirà con il sistema. Nel nostro caso specifico, trattasi, ad esempio, dei web browser che verranno utilizzati dagli utilizzatori finali;
- **Security test**: viene valutata la robustezza del sistema in materia di sicurezza;
- **Performance test**: si analizzano la reattività e le prestazioni del sistema, anche in situazioni di particolare carico.

Il codice utilizzato per denotare questa fase di test sarà:

TS[Identificativo test]

3.2.2.4.4 Test di non regressione

Durante lo sviluppo, qualora si renda necessaria la modifica di alcune componenti, i test di non regressione consistono nel rieseguire tutti i test riguardanti le componenti aggiornate, in modo tale da verificare che il funzionamento non sia stato compromesso. Il codice utilizzato per denotare questa fase di test sarà:

TNR[Identificativo test]

3.2.2.4.5 Test di validazione

Esso è il controllo ultimo e definitivo che il prodotto realizzato dal fornitore sia conforme alla richiesta del proponente. Il test è effettuato in presenza del proponente, ed in caso di esito positivo, il software può essere rilasciato. Il codice utilizzato per denotare questa fase di test sarà:

TV[Identificativo test]

3.2.2.5 Strumenti

3.2.2.5.1 Strumenti per l'analisi statica

- ***W3C Markup Validator Service$_G$***: validatore online di codice *HTML$_G$*, utile per trovare eventuali errori nel codice. L'indirizzo web di riferimento è: `https://validator.w3.org/`;
- ***CSSLint$_G$***: validatore online di codice *CSS$_G$*, utile per trovare eventuali errori nel codice. L'indirizzo web di riferimento è: `http://csslint.net/`;
- ***JSHint$_G$***: è uno strumento che aiuta a rilevare possibili errori nel codice *JavaScript$_G$*. L'indirizzo web di riferimento è: `http://jshint.com/`;
- ***SQLFiddle$_G$***: validatore online di codice *SQL$_G$*, utile per verificare la consistenza del codice del database, e compatibile con *MySQL$_G$* e *PostgreSQL$_G$*. L'indirizzo web di riferimento è: `http://sqlfiddle.com/`.

3.2.2.5.2 Strumenti per l'analisi dinamica

- ***Google Chrome DevTools$_G$***: è uno strumento atto a controllare l'utilizzo di risorse del prodotto, l'usabilità su differenti dispositivi, nonchè simulare situazioni di carico della rete. L'indirizzo web di riferimento è: `https://www.google.it/chrome/`;
- ***Jasmine$_G$***: framework utile ad eseguire test di unità su codice JavaScript. L'indirizzo di riferimento è: `https://jasmine.github.io/`;
- ***Protractor$_G$***: framework per effettuare test su applicazioni sviluppate in AngularJS. L'indirizzo di riferimento è: `http://www.protractortest.org/#/`.

3.3 Validazione

3.3.1 Validazione dei documenti

3.3.1.1 Descrizione

L'attività di validazione dei documenti riguarda la fase preliminare al rilascio di una nuova versione di documento. Il responsabile di tale attività è il *Responsabile di Progetto*, e prevede una lettura e veloce analisi del documento sottoposto ad approvazione.

3.3.1.2 Strategia generale

Durante l'attività di approvazione, il *Responsabile* deve procedere ad una lettura complessiva del documento, per approvare le modifiche e correzioni stabilite.
Viene controllato, prima dell'approvazione e distribuzione, l'indice Gulpease affinchè rispetti gli standard prefissati.

3.3.1.3 Obiettivi attesi

I documenti devono essere coerenti con la strategia del team e la restante documentazione.

3.3.2 Validazione del codice

3.3.2.1 Descrizione

L'attività di validazione del codice riguarda un controllo manuale per quanto riguarda i requisiti del software, in modo che le funzioni siano effettivamente coerenti con i casi d'uso stabiliti.

3.3.2.2 Strategia generale

Un test generale del prodotto verrà svolto seguendo le procedure come indicate nel *Manuale Manutentore*, per verificare che il risultato atteso sia fruibile completamente per l'utilizzatore finale.
Inoltre, vengono verificati tutti i casi d'uso, in modo che l'applicazione soddisfi in pieno i requisiti obbligatori pattuiti con il proponente.
Qualora alcuni requisiti non vengano soddisfatti, il software viene sottoposto ad una nuova fase di incremento per integrare le modifiche necessarie alla sua approvazione e accettazione.

3.3.2.3 Obiettivi attesi

L'obiettivo principale è che il software rilasciato rispetti tutti i requisiti attesi e pattuiti in fase di definizione contrattuale.

4 Processi organizzativi

4.1 Pianificazione di progetto

La pianificazione riguarda tutto ciò che è inerente alla gestione del personale per conseguire gli scopi prefissati nei tempi pattuiti.
La pianificazione dell'attività è svolta dal *Responsabile di Progetto* che compila il piano di progetto e si occupa della divisione dei tasks, per garantire una corretta ed efficiente ripartizione del lavoro.

4.1.1 Organizzazione

Il *Responsabile di Progetto* verifica i vincoli di ore disponibili nei vari ruoli e per ogni candidato, e procede ad una corretta organizzazione.
In particolare, ripartisce equamente le ore tra i membri del team, con lo scopo ultimo di ottenere un prodotto conforme nei minori tempi possibili.
Il *Responsabile* deve, inoltre, garantire l'assegnazione di ruoli differenti alla stessa persona, per permettere una corretta conoscenza di tutti gli aspetti del progetto.

4.1.1.1 Metodologia

Per quanto concerne le operazioni elencate astrattamente nella sezione soprastante, esse corrispondono alle seguenti attività. Il *Responsabile di Progetto* ha il compito di:

- Assegnare equamente ai membri del team i differenti ruoli che dovranno intraprendere nel corso del progetto o di una data fase progettuale;
- Calcolare i tempi effettivi e assicurarsi che il carico di lavoro venga ripartito correttamente;
- Procedere allo scheduling delle attività, coadiuvata dagli strumenti organizzativi in uso.

4.1.2 Consuntivazione

La consuntivazione è un'attività che viene svolta alla fine di ogni periodo prestabilito, nella quale vengono presi in esame i dati dalla pianificazione e si valutano le performance e i relativi aggiustamenti da effettuare per il periodo successivo.
I dati in uscita dall'attività di consuntivazione devono essere dettagliati e coerenti, presentando un resoconto atto a mitigare eventuali situazioni di pianificazione errata.

4.1.2.1 Metodologia

Per conseguire un efficiente consuntivazione, vengono considerate e svolte le seguenti attività:

- Al termine di ogni periodo, si deve analizzare tempestivamente la performance in modo preliminare alla scadenza del periodo stesso, come ultima attività da svolgere;
- Si deve confrontare quanto effettivamente svolto dalle ore pianificate durante l'inizio del periodo;
- Si aggiorna il *Piano di Progetto* con la consuntivazione dei rischi e l'eventuale mitigazione effettuata;
- Si analizzano le motivazioni che han portato ad eventuali discrepanze in negativo, procedendo a svolgere degli audit interni al team, dove si discutono le possibili soluzioni da attuare.

4.2 Gestione Organizzativa

4.2.1 Gestione delle attività

La gestione delle attività regolamenta come verranno organizzate le attività del progetto. Ogni attività dovrà essere divisa in task di minore entità, in modo tale che essi possano essere presi in carico e svolti da un solo membro.
I tasks così divisi devono essere assegnati al membro del team che ricopre il ruolo di competenza per tali attività.
Ogni attività dovrà essere svolta dal membro del team alla quale è stata assegnata, con un attenzione particolare al rispetto delle tempistiche stabilite dal *Responsabile di Progetto*.

4.2.1.1 Gestione dei task

La lista dei singoli task, creata e assegnata dal *Responsabile di Progetto*, sarà organizzata tramite i software indicati nel presente documento.
Qualora un task fosse assegnato a un gruppo di persone, essi hanno facoltà di suddividerli in sub-task e gestire autonomamente la programmazione purchè rispetti le tempistiche stabilite.
Qualora non risulti unanime la decisione per la divisione di sub-task, sarà il *Responsabile di Progetto* stesso a procedere a una ripartizione corretta.
Al completamento di un task, all'assegnatario viene affidato un *Verificatore* che avrà il compito di segnalare eventuali problematiche o discrepanze in quanto prodotto in output per il task (documentazione o codice).
In seguito a quanto segnalato dal *Verificatore*, vengono effettuate le opportune correzioni e il task può venir considerato chiuso.
Il *Responsabile di Progetto* deve essere messo al corrente del reale apporto di tempo impiegato per la conclusione di una determinata attività.

4.2.1.2 Metodologie

Durante il periodo considerato, la pianificazione effettuata viene aggiornata costantemente per tenere conto delle situazioni di rischio verificatesi o potenziali, e per tenere conto dei bisogni specifici di ciascun membro del team.

4.2.2 Comunicazione e coordinamento

Al fine di ottenere comunicazioni chiare e concise il *Responsabile di Progetto* ha l'obbligo di gestirle in modo strutturato, utilizzando la forma che meglio si adatta alla situazione. Le comunicazioni possono essere interne o esterne al team.

4.2.2.1 Comunicazioni interne

4.2.2.1.1 Descrizione

Le comunicazioni interne sono uno strumento ad uso esclusivo dei membri del gruppo e possono essere in stile informale o formale, in forma scritta o orale.

- **Comunicazione formale scritta**: è un'informazione rilevante per i membri del gruppo, deve essere utilizzata quando è di fondamentale importanza per la corretta gestione del progetto;
- **Comunicazione formale orale**: questa modalità sarà utilizzata durante le presentazioni o in occasione di incontri con il proponente e/o il committente. Se la comunicazione con questi ultimi ha particolare importanza per lo svolgimento del progetto, deve essere scritta e diventare un'informazione rilevante per i membri del team;

- **Comunicazione informale scritta**: scambio di messaggi tra i membri del team mediante l'utilizzo dell'applicazione di instant messaging *Telegram*$_G$, sulla quale è presente la chat del gruppo, utile per chiarimenti o precisazioni;
- **Comunicazione informale orale**: colloquio che avviene tra i membri del team e possono riguardare opinioni e informazioni sulle attività del progetto.

4.2.2.2 Comunicazioni esterne

Per tutte le comunicazioni esterne, il gruppo *NetBreak* si è dotato di una email ufficiale, necessaria per gestire le comunicazioni con committente e proponente. La gestione di tale casella di posta elettronica è affidata al *Responsabile di Progetto*, il quale ha il compito di tenere informati i membri del gruppo riguardo le comunicazioni importanti. La casella di posta elettronica è:

`netbreakswe@gmail.com`

Le email ufficiali devono rispettare alcune linee guida, esposte di seguito:

- **Destinatario:** indica il destinatario dell'email e se non è già presente nei contatti, è necessario salvarlo, tramite la funzione automatica di *Gmail*$_G$;
- **Oggetto:** l'oggetto dell'email deve esprimere in modo chiaro e breve il contenuto dell'email;
- **Corpo:** il corpo del messaggio è composto dal contenuto informativo che il mittente vuol comunicare ai destinatari;
- **Allegati:** un allegato è un file di testo, audio o video inviato insieme a una email. La dimensione del file influisce sulla dimensione totale dell'email e non è possibile inviare allegati sopra determinate dimensioni espresse in megabyte.

4.2.2.3 Coordinamento con secondo gruppo

Il proponente richiede una collaborazione tra i gruppi intenzionati a sviluppare il progetto proposto.
In particolare, ItalianaSoftware S.r.l. richiede un'intesa sull'architettura di massima del progetto: al fine di agevolare la comunicazione tra i gruppi è stato creato un gruppo su Telegram per il coordinamento del capitolato C1.

4.2.3 Riunioni

Le riunioni sono un elemento essenziale per il corretto svolgimento del progetto. Il loro compito è quello di permettere a tutti i membri del gruppo di confrontarsi tra loro, nel caso di riunioni interne, e con il proponente e il committente, nel caso di riunione esterne.
La corretta gestione dell'andamento di una riunione permette di risparmiare tempo e sfruttare al meglio le risorse umane.

4.2.3.1 Riunioni interne

Il *Responsabile di Progetto* ha la facoltà di decidere e convocare una riunione del gruppo. Tale convocazione deve essere inviata tramite posta elettronica a tutti i membri del gruppo, con almeno due giorni di anticipo, e deve contenere l'ordine del giorno.
Tutti i membri del gruppo hanno il dovere di confermare la loro presenza entro ventiquattro ore dalla ricezione della comunicazione. Se impossibilitati a partecipare, devono fornire al *Responsabile di Progetto* una valida motivazione alla loro assenza.
Il *Responsabile di Progetto*, inoltre, ha il compito di dirigere la riunione, tenendo fede all'ordine

del giorno descritto nella mail di invito ed evitare perdite di tempo. Tutti i partecipanti devono presentarsi alla riunione preparati, aver rispettato le scadenze decise nella precedente riunione e caricato il materiale assegnato sulla piattaforma di web storage indicata. Le scadenze sono visibili sull'applicazione multi piattaforma *Asana*$_G$. All'inizio di ogni riunione il *Responsabile di Progetto* nominerà un segretario, che avrà il compito di tenere la minuta dell'incontro, stilare un verbale informale a disposizione esclusivamente del gruppo e creare le attività su Asana con relative scadenze concordate.
Le riunioni devono avere un preciso scopo e in base ad esso possono avvenire tramite *Skype*$_G$ oppure devono essere svolte di persona. Le riunioni si dividono in quattro tipi:

- **Informative:** il *Responsabile di Progetto* informa il gruppo riguardo informazioni di carattere generale, comunicazioni ricevute da committente e/o proponente, prossime scadenze e piano di lavoro per il periodo attuale. Queste riunioni possono anche essere tenute su Skype, oltre che fisicamente;
- **Valutative:** il *Responsabile di Progetto* convoca la riunione per valutare insieme ai membri del team alcuni aspetti dello svolgimento del progetto in seguito a eventi particolari, quali nuove scadenze da rispettare oppure comunicazioni rilevanti dal proponente. È possibile svolgerle su Skype solo in caso di problemi logistici che ne impediscono un incontro fisico;
- **Decisionali:** queste riunioni non posso avvenire tramite Skype, ma richiedono un incontro fisico tra i componenti. In queste riunioni, si prendono decisioni rilevanti per lo svolgimento del progetto e, quindi, la presenza fisica dei membri è necessaria per evitare incomprensioni e confrontarsi di persona;
- **Di progettazione:** come per le riunioni di tipo decisionale, questa tipologia di riunioni richiede un incontro fisico del gruppo poichè riguardano l'importante e fondamentale attività di progettazione del prodotto API Market.

Affinchè una riunione risulti utile e produttiva, occorrerà prendere delle decisioni in merito a:

- Definire gli obiettivi da raggiungere;
- Designare i responsabili;
- Scegliere gli strumenti;
- Determinare i tempi di esecuzione che si intendono rispettare;
- Fissare la data di una nuova ed eventuale riunione, sia essa interna o esterna.

Importante è la gestione del dopo riunione, attraverso la compilazione di un verbale che metta a conoscenza tutti i membri del team delle decisioni prese.

4.2.3.2 Riunioni esterne

Questo tipo di riunioni è fondamentale per il corretto svolgimento e progessivo andamento del progetto. Il *Responsabile di Progetto* è una figura chiave in questi incontri, in quanto deve svolgere un numero di compiti notevole.
Nei giorni precedenti alla riunione, sarà compito del *Responsabile di Progetto* stilare un documento condiviso con l'ordine del giorno, le domande e/o i chiarimenti da esporre. Se la riunione è con il proponente e richiede di incontrare entrambi i gruppi al lavoro sul capitolato scelto, il *Responsabile di Progetto* deve contattare il responsabile del secondo gruppo per accordarsi sullo svolgimento dell'incontro.
Prima di ogni riunione, il *Responsabile di Progetto* nominerà un segretario, incaricato di prendere nota dei temi discussi e redigere un verbale, evidenziando gli aspetti più rilevanti e/o le modifiche

da apportare. Tale documento è riservato ad un uso esclusivo del team.
A questi incontri, parteciperanno tutti i membri del gruppo, salvo impedimenti di rilievo, e sarà compito del *Responsabile di Progetto* coordinare e gestire l'interazione tra il gruppo e l'interlocutore.
Durante la riunione, i membri del team potranno effettuare domande e/o chiedere chiarimenti.

4.2.3.3 Verbali

Il gruppo ha deciso di redigere dei verbali per ogni riunione che viene effettuata. I verbali sono un documento utile al gruppo, utile per tenere traccia degli argomenti discussi e delle decisioni prese.
Il team ha deciso di dividere i verbali in due categorie (interni ed esterni) e di indicare all'inizio di ogni verbale la tipologia della riunione.
All'inizio della riunione, il *Responsabile di Progetto* nomina un segretario, il quale ha il compito di procedere alla stesura del documento.
Il documento contiene un abstract che indica il motivo della convocazione della riunione, un elenco puntato con informazioni sulla riunione, un riassunto dei temi e contenuti trattati e infine, una tabella di tracciamento per le decisioni significative prese.
L'elenco contiene le seguenti informazioni:

- **Tipologia:** indica la tipologia della riunione tra informative, valutative, decisionali e di progettazione. È possibile che una riunione abbia più tipologie;
- **Generalità del segretario:** indica il nome dell'incaricato a redigere il verbale e farlo avere al Responsabile per l'approvazione;
- **Data:** indica la data di svolgimento della riunione, nel formato DD MM YYYY;
- **Luogo:** indica il luogo di svolgimento della riunione. Le riunioni possono avvenire sia fisicamente che virtualmente, tramite Skype. Quest'ultimo metodo sarà utilizzato per brevi comunicazioni e aggiornamenti sullo svolgimento del progetto, o qualora dovessero presentarsi problemi per un incontro fisico;
- **Orario inizio:** indica l'orario di inizio della riunione nel formato HH:MM;
- **Orario fine:** indica l'orario di fine della riunione nel formato HH:MM;
- **Membri presenti:** indica i componenti del gruppo presenti alla riunione. Nel caso di riunioni con il proponente, congiuntamente al secondo gruppo, devono essere indicati anche i nomi di quest'ultimi;
- **Membri assenti:** indica i componenti del gruppo assenti alla riunione.

La tabella di tracciamento delle decisioni è un utile strumento per il team, ma soprattutto per persone esterne che vogliono informarsi in modo rapido sulle riunioni fatte.
La tabella consiste di due colonne, una per l'identificativo della decisione presa o del significativo evento accaduto, e l'altra per una breve e riassuntiva descrizione testuale riguardo la decisione associata.
L'identificativo seguirà la seguente forma sintattica:

$$V[I/E]\_yy$$

dove I = Interno, E = Esterno, yy = numero progressivo

4.3 Ruoli e mansioni

Nel corso del progetto, ogni membro del team ricoprirà diverse mansioni. I ruoli sono standard, stabiliti con criterio e con mansioni mutuamente esclusive.
Lo scopo di questa sezione è descrivere tutti i possibili ruoli che un membro del team può assumere durante lo svolgimento del progetto. Essi verranno organizzati in modo tale da:

- Evitare conflitti di interesse, quali concomitanza dei ruoli di stesura e verifica di uno stesso documento;
- Ruotare i ruoli per permettere a ciascun membro di assumere qualsiasi posizione durante il corso del progetto;
- Permettere ad una persona di assumere più mansioni contemporaneamente.

4.3.1 Responsabile di Progetto

Il *Responsabile di Progetto* organizza il lavoro interno del team, occupandosi della pianificazione delle attività. Egli mantiene i contatti con tutti gli enti esterni, gestisce le risorse, i rischi e la documentazione. Di quest'ultima, è l'unico incaricato di apporre l'approvazione definitiva.
Le ulteriori mansioni organizzative del *Responsabile di Progetto* sono quelle di garantire il rispetto dei ruoli ed assicurarsi che le attività assegnate vengano svolte scrupolosamente nei tempi stabiliti secondo i canoni stabiliti nel documento NormeDiProgetto 4_0_0.pdf.
I documenti di sua diretta responsabilità sono PianoDiProgetto 4_0_0.pdf e PianoDiQualifica 4_0_0.pdf.

4.3.2 Amministratore

L'*Amministratore* svolge una attività di controllo sull'ambiente di lavoro come mansione principale. Altre dirette responsabilità dell'*Amministratore* sono relative al versionamento di prodotto e alla documentazione.
Il documento di sua diretta responsabilità è NormeDiProgetto 4_0_0.pdf, ma collabora anche alla redazione del documento PianoDiProgetto 4_0_0.pdf.

4.3.3 Analista

La figura di *Analista* ha il ruolo di esaminare caratteristiche e problematiche del prodotto. Si occupa, dunque, dell'attività di analisi per ogni aspetto del progetto.
Si occupa direttamente dei documenti StudioDiFattibiltà 1_0_0.pdf e AnalisiDeiRequisiti 3_0_0.pdf, mentre collabora alla stesura del documento PianoDiQualifica 4_0_0.pdf.

4.3.4 Progettista

Il ruolo di *Progettista* è una mansione che ha il compito primario di realizzare una produzione astratta del progetto, effettuando le scelte necessarie a mantenere il prodotto modulare, di facile manutenzione e ampliamento.
Il *Progettista* si occupa direttamente della stesura dei documenti SpecificaTecnica 3_0_0.pdf, DefinizioneDiProdotto 2_0_0.pdf e della parte di metriche del documento PianoDiQualifica 4_0_0.pdf.

4.3.5 Programmatore

Il *Programmatore* svolge l'attività di produzione di codice che implementa la soluzione finale progettata.
Egli ha il compito di seguire in maniera attenta la progettazione pregressa, effettuando la codifica

seguendo le convenzioni descritte nel documento NormeDiProgetto 4_0_0.pdf.
Si occupa, inoltre, di predisporre i test per le successive attività di verifica, e di documentare quanto viene prodotto.
Il *Programmatore* si occupa direttamente della stesura del documento Manuale Utente 2_0_0.pdf e della documentazione relativa al codice.

4.3.6 Verificatore

Il *Verificatore* svolge l'attività di Verifica di tutta la documentazione prodotta. Si attiene scrupolosamente a NormeDiProgetto 4_0_0.pdf, al fine di assicurarsi che i documenti prodotti siano conformi agli standard e alle convenzioni prefissate.
Inoltre, si occupa di redigere la sezione del documento PianoDiQualifica 4_0_0.pdf, il quale illustra il resoconto delle verifiche effettuate sui vari documenti prodotti.

4.4 Gestione delle infrastrutture

Le infrastrutture utilizzate per la realizzazione e l'eventuale deployment del prodotto, o per qualsiasi attività di sviluppo che richieda l'impiego di particolari strumenti, deve essere strettamente regolamentato secondo i seguenti obbiettivi in termini di qualità:

- Deve essere garantita la massima stabilità alla piattaforma utilizzata, evitando azioni di carattere sperimentale non approvate che possano minare l'affidabilità della piattaforma;
- Deve essere garantito un uptime quanto più elevato possibile, per permettere un lavoro continuativo e senza interruzioni;
- Il team deve sensibilizzarsi all'utilizzo con parsimonia ed efficienza delle risorse, onde evitare sprechi e picchi di utilizzo che vanno ad inficiare sui punti soprastanti;
- Devono essere valutati, se presenti, i costi associati al servizio necessario e valutato a preventivo l'impatto degli ultimi.

4.4.1 Metodologie

Per perseguire gli obiettivi prefissati, si dovranno prestare le seguenti attenzioni supplementari:

- Le infrastrutture utilizzate devono avere un accesso in comune per tutti i membri del team, in modo che ciascuno possa avere accesso o possibilità di manutenzione sull'infrastruttura;
- Le azioni svolte su strumenti e infrastrutture dovranno essere opportunamente annotate, qualora esse non dispongano di meccanismi di logging interno;
- Sono strettamente vietate le attività e gli utilizzi che esulano dallo scopo del progetto, quali l'utilizzo di file e applicazioni di carattere personale;
- Deve essere utilizzato software stabile, e la versione utilizzata deve essere concordata e uniformata tra i membri del team;
- Il software e le librerie aggiuntive, così come ogni ausilio esterno, devono provenire da fonti affidabili e verificate.

4.5 Strumenti

4.5.1 GitHub

Per il controllo di versione e la gestione semplificata dei file di progetto e della documentazione, si è deciso di utilizzare **GitHub**. La scelta è ricaduta su questo strumento, poiché altre piattaforme prese in considerazione non consentono la collaborazione di sei membri o più in modo gratuito.
Il team lavorerà, dunque, su un repository pubblico. I membri del team potranno, tramite questa soluzione, collaborare simultaneamente, sincronizzando le versioni in tempo reale o in un secondo momento, qualora mancasse la connessione.
Infatti, è possibile lavorare sulla copia locale della repository e caricare le modifiche una volta ristabilita la connessione con il server centrale.
Durante la produzione dei documenti tramite TeXstudio, vengono generati diversi file, non utili ai fini del progetto, e il file *.gitignore* presente sulla repository evita il caricamento di file locali con determinate estensioni, al fine di aiutare a mantenere la repository ordinata e di facile fruizione.
GitHub offre, inoltre, una sezione wiki per la gestione di eventuale documentazione, e un sistema di issue tracking sufficientemente performante ai fini del progetto. La struttura base della repository contenente le cartelle con i relativi documenti, sarà la seguente:

- RR;
- RP;
- RQ;
- RA;
- template.

Ogni cartella che contiene la documentazione relativa alla specifica revisione di progetto, a sua volta racchiude altre cartelle, una per ogni documento.

4.5.2 Google Drive

Per lo storage di ulteriori file, documenti e per tutto ciò che non è strettamente correlato alla produzione e consegna del progetto, sarà utilizzato il sistema di cloud storage ***Google Drive***$_G$.
Ciò permette l'accesso dei file a tutti i membri e la possibilità di modificarli, pur non avendo una gestione capillare del versionamento, che, per l'appunto, è affidata a GitHub.

4.5.3 Asana

Per permettere una più semplice suddivisione del lavoro e una conseguente assegnazione delle mansioni, si è scelto di utilizzare il software **Asana**. Esso permette la creazione, modifica e assegnazione di task e subtask.
Per ognuno di essi, va indicato un assegnatario e una data di scadenza. La creazione di task è limitata alla sola figura di *Responsabile di Progetto*, che coordina il lavoro tra i membri del gruppo, evitando squilibri e conflitti di interessi.
I task devono essere quanto più specifici possibile, e devono descrivere nel dettaglio i ruoli che l'assegnatario andrà a svolgere. Ogni componente del gruppo è tenuto a verificare le mansioni assegnate con frequenza giornaliera, ad aggiornare lo stato dei propri task e notificare l'eventuale completamento, e a esporre o rispondere a dubbi e interrogativi sollevati inerenti all'attività assegnata, tramite l'apposita conversazione disponibile per ogni singolo task.