SWEG GROUP

Piano di Qualifica

Versione 2.0.0
Data di rilascio 06/03/2017
Redazione Gianluca Crivellaro
Piergiorgio Danieli
Verifica Pietro Lonardi
Validazione Sebastiano Marchesini
Responsabile Gianluca Crivellaro
Uso Esterno
Destinato ItalianaSoftware S.r.l
Prof. Vardanega Tullio
Prof. Cardin Riccardo

Sommario
Questo documento ha l'obiettivo di fornire un metodo per misurare la qualità del prodotto.

1 Registro Modifiche

Ver.	Modifica	Nome	Data
2.0.0	Validazione	Danieli Piergiorgio	06/03/2017
1.2.0	Verifica	Sebastiano Marchesini	06/03/2017
1.1.1	Estensione dell'appendice	Lonardi Pietro	06/03/2017
1.1.0	Verifica	Sebastiano Marchesini	28/02/2017
1.0.3	Stesura Pianificazione Test	Lonardi Pietro	25/02/2017
1.0.2	Stesura dell'appendice	Lonardi Pietro	22/02/2017
1.0.1	Correzzione delle metriche e misure	Lonardi Pietro	21/02/2017
1.0.0	Validazione	Sebastiano Marchesini	10/01/2017
0.5.0	Verifica	Lonardi Pietro	10/01/2017
0.2.0	Stesura del documento	Crivellaro Gianluca	04/01/2017
0.1.0	Stesura introduzione	Danieli Piergiorgio	04/01/2017
0.0.1	Studio dei riferimenti	Crivellaro Gianluca	02/01/2017

Indice

1 Registro Modifiche **2**

2 Introduzione **5**
2.1 Scopo del documento . . . 5
2.2 Scopo del Prodotto . . . 5
2.3 Glossario . . . 5
2.4 Riferimenti . . . 5
2.4.1 Normativi . . . 5
2.4.2 Informativi . . . 5

3 Definizione obiettivi di qualità **6**
3.1 Funzionalità . . . 6
3.2 Affidabilità . . . 6
3.3 Usabilità . . . 6
3.4 Efficienza . . . 6
3.5 Manutenibilità . . . 7
3.6 Portabilità . . . 7
3.7 Altre qualità . . . 7

4 Visione generale delle strategie di verifica **8**
4.1 Procedure di controllo di qualità di processo . . . 8
4.2 Procedure di controllo di qualità di prodotto . . . 8
4.3 Organizzazione . . . 8
4.4 Pianificazione strategica e temporale . . . 9
4.5 Responsabilità . . . 9
4.6 Risorse . . . 9
4.7 Tecniche di analisi . . . 10
4.7.1 Analisi statica . . . 10
4.7.2 Analisi dinamica . . . 10
4.8 Misure e metriche . . . 10
4.8.1 Metriche per i processi . . . 10
4.8.2 Metriche per i documenti . . . 11

5 Gestione amministrativa della revisione **12**
5.1 Comunicazione e risoluzione delle anomalie . . . 12
5.2 Procedure di controllo di qualità del processo . . . 12

6 Resoconto delle attività di verifica **13**

7 Pianificazione dei test **14**
7.1 Test di sistema . . . 14
7.2 Test di integrazione . . . 15
7.2.1 TI.model.search.SearchDataResult . . . 16
7.2.2 TI.model.SessionToken . . . 16
7.2.3 TI.model.modelView.SysLogin . . . 16
7.2.4 TI.model.modelView.recovery . . . 16
7.2.5 TI.model.modelView.ShowUserPage . . . 16
7.2.6 TI.model.modelView.ShowBuyCredits . . . 16

7.2.7 TI.model.modelView.ShowApiPage 16
7.3 Test di integrazione back-end 17

A Appendice **19**
A.1 Risultati al termine delle consegne 19
A.2 Analisi e analisi in dettaglio 19
A.2.1 Risultati dell'indice di Gulpease 19
A.2.2 Risultati PDCA 19
A.3 Progettazzione Architetturale 19
A.3.1 Risultati dell'indice di Gulpease 20
A.3.2 Risultati PDCA 20

2 Introduzione

2.1 Scopo del documento

Lo scopo del *Piano di Qualifica* è di descrivere le strategie di verifica e validazione adottate dal gruppo SWEg al fine di perseguire obiettivi qualitativi per il progetto APIMarket$_g$.

2.2 Scopo del Prodotto

Lo scopo del progetto è la realizzazione di un applicazione web che gestisca e monitori microservizi, muniti di un'interfaccia Jolie. In particolare APIMarket$_g$ deve permettere di vendere, acquistare e monitorare microservizi tramite la piattaforma.

2.3 Glossario

Al fine di evitare ambiguità e ottimizzare la comprensione dei documenti, viene incluso un Glossario, nel quale saranno inseriti i termini tecnici, acronimi e parole che necessitano di essere chiarite.
Un glossario è una raccolta di termini di un ambito specifico e circoscritto. In questo caso per raccogliere termini desueti e specialistici inerenti al progetto.

2.4 Riferimenti

2.4.1 Normativi

- **Norme di Progetto**:
  Norme di Progetto v2.0.0;
- **Capitolato d'appalto C1**: APIM: An API$_g$ Market Platform
  http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/Capitolati.html

2.4.2 Informativi

- **Piano di Progetto**:
  Piano di Progetto v2.0.0;
- **Slide di Ingegneria del Software**:
  http://www.math.unipd.it/~tullio/IS-1/2016/;
- **SWEBOK - Version 3 (2004)**: capitolo 11 - Software Quality
  https://www.computer.org/web/swebok;
- **ISO$_g$/IEC$_g$ TR 15504**: Software process assessment:
  https://en.wikipedia.org/wiki/ISO/IEC_15504;
- **ISO$_g$/IEC$_g$ 9126: Product quality**:
  https://en.wikipedia.org/wiki/ISO/IEC_9126;
- **Nielsen$_g$ e le sue 10 euristiche**:
  http://www.far.unito.it/usabilita/Cap5.htm

3 Definizione obiettivi di qualità

Il gruppo SWEg si impegna a rispettare il modello di standard definito in ISO/IEC 9126, il quale ha le seguenti qualità:

3.1 Funzionalità

Il sistema prodotto deve garantire tutte le funzionalità indicate nel documento *"Analisi dei Requisiti v2.0.0"*. L'implementazione dei requisiti deve essere più completa ed economica possibile.

- **Misura**: viene usata come unità di misura la quantità di requisiti mappati in componenti del sistema create e funzionanti;
- **Metrica**: la sufficienza è stabilita nel soddisfacimento di tutti i requisiti obbligatori;
- **Strumenti**: perchè questa qualità sia soddisfatta il sistema deve superare tutti i test previsti.

3.2 Affidabilità

Il sistema si deve dimostrare robusto e di facile ripristino in caso di errori.

- **Misura**: l'unità di misura sarà la quantità di esecuzioni del sistema andati a buon fine;
- **Metrica**: le esecuzioni dovranno soddisfare la più ampia gamma di funzionalità previste;
- **Strumenti**: da definire.

3.3 Usabilità

Il sistema prodotto deve risultare di semplice utilizzo per l'utente. Questo sistema deve allo stesso tempo soddisfare tutte le necessità dell'utente.

- **Misura**: i dieci punti dell'euristica di Nielsen;
- **Metrica**: la sufficienza è stabilita dal soddisfacimento del numero minimo di punti;
- **Strumenti**: perchè questa qualità sia soddisfatta il sistema deve superare tutti i test previsti.

3.4 Efficienza

Il sistema deve fornire tutte le funzionalità nel più breve tempo possibile, con il minimo utilizzo di risorse.

- **Misura**: tempi di latenza per ottenere una risposta dall'applicazione web in condizioni normali o in caso di sovraccarico della rete;
- **Metrica**: la sufficienza viene stimata come un tempo di latenza minore del 20% su rete via cavo o Wi-Fi e del 30% su reti cellulari rispetto alla media dei tempi di latenza del server web;
- **Strumenti**: si veda il documento *"Norme di Progetto v2.0.0"*.

3.5 Manutenibilità

Il sistema prodotto deve essere comprensibile ed estensibile in modo facile e verificabile.

- **Misura**: l'unità di misura utilizzata saranno le metriche sul codice descritte nel documento *"Norme di Progetto v2.0.0"* ;
- **Metrica**: le metriche che verranno rispettate sono descritte nel documento *"Norme di Progetto v2.0.0"*;
- **Strumenti**: si veda il documento *"Norme di Progetto v2.0.0"*.

3.6 Portabilità

Il sistema deve essere più portabile possibile. Il sito deve essere visitabile da più browser possibili. Il sito deve essere sviluppato con sistemi che abbiano le varie componenti tecnologiche di tipo standard. Con il termine "standard" si intente che i contenuti devono poter essere utilizzati su più sistemi operativi e ambienti di lavoro possibili.

- **Misura**: il front end deve aderire agli standard W3C;
- **Metrica**: se il software avrà la sufficienza in tutte le metriche descritte nel documento *"Norme di Progetto v2.0.0"*, allora il sito avrà le caratteristiche di portabilità descritte;
- **Strumenti**: si veda il documento *"Norme di Progetto v2.0.0"*.

3.7 Altre qualità

Saranno inoltre importanti per il prodotto le seguenti qualità:

- **Semplicità**: si ottiene tramite la realizzazione del prodotto nel modo più semplice, senza tralasciare la qualità;
- **Incapsulamento**: applicare le tecniche di incapsulamento per aumentare la manutenibilità e la possibilità di riuso del codice. L'interazione sarà possibile tramite interfacce;
- **Coesione**: riguarda le funzionalità che collaborano tra di loro per ottenere uno stesso obiettivo. Devono risiedere nello stesso componente, e come scopo hanno quello di ridurre l'indice di dipendenza, favorire la semplicità e la manutenibilità.

4 Visione generale delle strategie di verifica

4.1 Procedure di controllo di qualità di processo

La qualità dei processi verrà garantita dall'applicazione del principio PDCA. Grazie a questo principio sarà possibile garantire non solo il controllo e la correttezza, ma anche il miglioramento costante della qualità di tutti i processi. Come conseguenza diretta si otterrà il miglioramento del prodotto.
Per avere controllo dei processi, e di conseguenza qualità, è necessario che:

- vi sia controllo sull'operato dei membri del team e sui processi;
- i processi siano pianificati in modo dettagliato;
- le risorse siano ripartite in modo chiaro.

L'attuazione di questi punti è descritta nel documento *"Piano di Progetto v2.0.0"*. Inoltre l'analisi costante della qualità del prodotto permette di monitorare in modo indiretto la qualità dei processi: se il prodotto è di bassa qualità sicuramente il processo è migliorabile.
Per valutare la qualità di un processo è fondamentale che questa sia quantificata, le metriche per fare ciò sono descritte nel documento *"Norme di Progetto v2.0.0"*.

Per monitorare costantemente lo stato del processo il gruppo fa affidamento alle direttive fornite dallo standard ISO/IEC 15504 che tratta esattamente di come identificare lo stato di maturazione di un processo in modo da poter agire per migliorarlo nel caso sia necessario. Per approfondimenti si riporta alla documentazione presenti di link ai riferimenti all'inizio di questo documento.

4.2 Procedure di controllo di qualità di prodotto

Il controllo di qualità dei prodotti verrà garantito dai seguenti processi:

- **Software Quality Assurance (SQA)**: assicura che i processi siano appropriati per il progetto, che siano correttamente implementati. Prevede l'attuazione di tecniche di analisi statica e dinamica, descritte nel documento *"Norme di Progetto v2.0.0"*;
- **Verifica**: è il procedimento che controlla la coerenza e la correttezza dei prodotti dei processi. La verifica verrà eseguita costantemente durante tutta la durata del progetto;
- **Validazione**: ovvero la conferma che il sistema soddisfi i requisiti e sia conforme alle attese.

4.3 Organizzazione

Il processo di verifica inizia nel momento in cui la versione del prodotto di un processo cambia. Il diario delle modifiche aiuta a monitorare solo le sezioni che sono state modificate, ottimizzando così i tempi di verifica.
Viene verificata sia la qualità del processo che del prodotto da esso compiuto.
Ogni fase del progetto, descritta nel documento *"Piano di progetto v2.0.0"* produce un prodotto diverso, quindi necessita di diverse attività di verifica:

- **Analisi e Analisi di Dettaglio**: in questa fase si verifica che i processi e la documentazione prodotta rispettino le norme definite nel documento *"Norme di Progetto v2.0.0"* e i tempi prestabiliti nel *"Piano di Progetto v2.0.0"*;

- **Progettazzione Achitetturale**; anche in questa fase sono presenti documenti perciò verranno effettuate le stesse verifiche della precedente fase, ovviamente sulle modifiche riportate nei documenti già esistenti e inoltre sui nuovi che vorranno prodotti. In questa fase non viene ancora steso codice perciò non verrà applicata un'analisi dinamica;
- **Programmazzione in Dettaglio e Codifica**: in questa fase si effettuerà una verifica sempre sui documenti che verranno stesi, sui vecchi documenti che subiranno modifiche e infine anche al codice che verrà prodotto attravero gli strumenti definiti nel documento *"Norme di Progetto v2.0.0"*.

4.4 Pianificazione strategica e temporale

È essenziale che l'attività di verifica sia sistematica ed organizzata, al fine di evitare la propagazione di errori ed ottimizzare i tempi di sviluppo, per rispettare le scadenze elencate nel documento *Piano di Progetto v2.0.0*.
Ogni fase di redazione di documenti e di codifica deve essere preceduta da una fase di studio preliminare, al fine di ridurre la possibilità di errori di natura concettuale o tecnica, e favorendo l'attività dei verificatori.

4.5 Responsabilità

Al fine di garantire un processo di verifica efficace ed efficiente vengono attribuite delle responsabilità all'interno del gruppo di progetto.
I ruoli che intervengono nel processo di verifica sono il *Project Manager* e il *Verificatore*, inoltre l'**Ammistratore** ha il compito di assicurarsi che i processi di verifica vengano fatti in modo ottimale. La suddivisione dei compiti è descritta nel documento *"Norme di Progetto v2.0.0"*.

4.6 Risorse

Per realizzare il prodotto sono necessarie le seguenti risorse:

- **Risorse umane**: vengono descritte dettagliatamente nel documento *"Piano di Progetto v2.0.0"*. E sono:
  - Project Manager;
  - Amministratore;
  - Analista;
  - Progettista;
  - Programmatore;
  - Verificatore.
- **Risorse software**: sono necessari strumenti software. In particolare per:
  - scrivere la documentazione in formato LaTeX;
  - creare diagrammi *UML*;
  - automatizzare le verifiche;
  - sviluppare nei linguaggi di programmazione scelti;
  - analizzare il codice scritto;
  - gestire i test che effettueremo sul codice.
- **Risorse hardware**: sono necessari computer aventi applicativi descritti nel documento *"Norme di Progetto v2.0.0"*.

4.7 Tecniche di analisi

4.7.1 Analisi statica

L'analisi statica è l'analisi eseguita senza una effettiva esecuzione dei programmi. Viene eseguita quindi sia sul codice che sui documenti durante tutta la fase di sviluppo.
Può essere applicata nei seguenti modi:

- **Walkthrough**: è una lettura del documento da verificare cercando errori senza un'idea precisa di cosa si sta cercando. È utile nelle prime fasi di sviluppo per stilare una lista degli errori più frequenti da condividere con il gruppo,
- **Inspection**: è una ricerca mirata degli errori che sono stati evidenziati nella lista di controllo, che può essere espansa qualora se ne palesi la necessità.

4.7.2 Analisi dinamica

L'analisi dinamica viene eseguita sulle componenti software, testando il comportamento dell'applicativo. I test, per essere attendibili, devono essere ripetibili nello stesso ambiente e con gli stessi input. I tipi di test che verranno eseguiti saranno:

- **Test di unità**: consiste nel controllo dei moduli che costituiscono l'applicativo;
- **Test di integrazione**: consiste nella verifica che i moduli, una volta collegati tra loro, diano il risultato previsto. Verifica inoltre il corretto funzionamento di librerie e framework esterni;
- **Test di sistema**: consiste nella validazione del prodotto software, una volta giunto ad uno stadio definitivo;
- **Test di regressione**: consiste nell'eseguire a cascata tutti i test collegati ad una componente appena modificata;
- **Test di accettazione**: consiste nel collaudo del software con il proponente prima del rilascio.

4.8 Misure e metriche

Qui di seguito vengono elencate le misure e le metriche che verranno adottate nel processo di verifica. Il processo di verifica deve essere verificabile e quantificato secondo metriche stabilite a priori, che posso essere consultate nelle *Norme di progetto v.2.0.0*.
Vi possono essere due tipologie di range nell'accettazione dei risultati di un processo di verifica:

- **Accettazione**: è il livello minimo che il prodotto deve raggiungere per superare il test;
- **Ottimale**: è il livello in cui dovrebbe posizionarsi la misurazione. In caso questo valore non venga raggiunto sono necessarie verifiche aggiuntive.

4.8.1 Metriche per i processi

Schedule Variance (SV) **Parametri utilizzati**:

- **Range di accettazione**: $\geq$ -(preventivo fase*5%);
- **Range ottimale**: ≥ 0.

Budget Variance (BV) **Parametri utilizzati**:

- **Range di accettazione**:≥-(preventivo fase*10%);
- **Range ottimale**:≥ 0.

4.8.2 Metriche per i documenti

Gulpease **Parametri utilizzati**:

- **Range di accettazione**: [40 - 100];
- **Range ottimale**: [50 - 100].

Metriche per il software

Complessità ciclomatica **Parametri utilizzati**:

- **Range di accettazione**: [1 - 15];
- **Range ottimale**: [1 - 10].

Rapporto linee di commento su linee di codice **Parametri utilizzati**:

- **Range di accettazione**: [>0.2];
- **Range ottimale**: [>0.3].

Numero di livelli di annidamento **Parametri utilizzati**:

- **Range di accettazione**: [0-6];
- **Range ottimale**: [0-4].

Numero di campi dati per classe **Parametri utilizzati**:

- **Range di accettazione**: [0-15];
- **Range ottimale**: [1-8].

Validazione W3C **Parametri utilizzati:**

- **Range di accettazione**: [0-5];
- **Range ottimale**: [0-0].

Euristica di Nielsen **Parametri utilizzati:**

- **Range di accettazione**: 6/10;
- **Range ottimale**: 10/10.

5 Gestione amministrativa della revisione

5.1 Comunicazione e risoluzione delle anomalie

Quando un verificatore trova un'anomalia dovrà aprire un nuovo ticket e avvisare il responsabile del documento. Un'anomalia corrisponde a:

- Un errore ortografico;
- Una violazione delle norme tipografiche definite nel documento *"Norme di Progetto v2.0.0"*;
- Un'incongruenza del prodotto rispetto a determinate funzionalità stabilite nel documento *"Analisi dei Requisiti v2.0.0"*;
- Un'incongruenza del codice con il design prodotto;
- L'uscita dal range di accettazione degli indici di misurazione.

5.2 Procedure di controllo di qualità del processo

La pianificazione dei processi, durante lo svolgimento delle varie fasi, si basa sul ciclo PDCA_{g}. Questo ciclio prende il nome dalle iniziali delle 4 fasi che ne fanno parte:

- **Plan**: fase di pianificazione, ovvero dove vengono fissati i periodi di tempi per raggiungere determinate $\text{milestone}_{\text{g}}$ utilizzando le risorse che sono state concesse;
- **Do**: si eseguono le attività secondo il piano prefissato;
- **Check**: viene effettuata una verifica tra i risultati che si aspettava di ragiungere secondo la fase di Plan e i dati che sono stati effettivamente raggiunti dalla fase di Do, in questo modo si ha un facile confronto tra dati per capire quale processo ha presentato dei problemi;
- **Act**: in base ai risultati emersi dalla fase check vengono apportate mofiche per cercare di migliorare i processi che hanno presentato problemi.

Attraverso l'utilizzo di questo ciclo è possibile avere un continuo miglioramento dei processi che si esprimere direttamente ad un miglioramento del prodotto finale.

6 Resoconto delle attività di verifica

Nel periodo antecedente la consegna di ogni revisione i documenti ed i processi vengono verificati e validati. I documenti vengono verificati tramite la procedura del walkthrough; in seguito sono state avviate le procedure per la notifica descritta nel documento *"Norme di Progetto v2.0.0"*.
Una volta individuati gli errori, vengono trattati nel seguente modo:

- Correzione degli errori individuati;
- Inserimento nella lista di controllo degli errori comuni;
- Applicazione del ciclo PDCA.

Successivamente viene applicato il metodo inspection ai diagrammi dei casi d'uso.

7 Pianificazione dei test

In questa sezione vengono elencati i test di sistema e di integrazione che verranno applicati nella fase successiva alla progettazzione quando si inizierà a sviluppare il codice effettivo; i test di unità verranno aggiunti in seguito. Si vuole utilizzare questa strategia di test perchè grazie a loro è possibile effettuare una verifica accurata sulle funzionalità che dovrà avere il prodotto una volta terminato, rispetto ai vincoli identificati nell'*Analisi dei Requisiti v.2.0.0*. Nella seguenti tabelle i test che riportano lo stato **N.A.** corrispondono a test non ancora appricati e che verranno applicati successivamente.

7.1 Test di sistema

Questi test si focalizzano sui requisiti principali del progetto, per i requisiti più specifici verranno svolti i test di unità e di integrazione.

Test	Descrizione	Stato	Requisito
TS 1	Viene verificato che il sistema sia sviluppato a microservizi	N.A.	R0V1
TS 2.1	Viene verificato che il sistema permetta di creare nuovi profili univoci che possano caricare nuove API e acquistare APIKey	N.A.	R0F1
TS 2.2	Viene verificato che il sistema permetta di autenticarsi come utente già presente nell'APIMarket	N.A.	R0F2
TS 3.1	Viene verificato che il sistema permetta la registrazione di un' API di un microservizio attraverso la registrazione e pubblicazione della sua interfaccia	N.A.	R0F3
TS 3.2	Viene verificato che il sistema segnali un errore nel caso in cui l'utente tenti di dare un nome già presente nell'APIMarket ad un suo microservizio	N.A.	R0F3.2.1
TS 4.1	Viene verificato che il sistema permetta all'utente di modificare i suoi dati personali, i vari messaggi e le interfacce dei microservizi che ha caricato	N.A.	R0F4
TS 4.2	Viene verificato che il sistema permetta di monitorare l'andamento dei microservizi che l'utente ha caricato	N.A.	R0F4.3.2
TS 5	Viene verificato che il sistema permetta di effettuare una ricerca di microservizi da parte sia di un utente autenticato che non	N.A.	R0F5
TS 6.1	Viene verificato che la ricerca ritorni il nome del microservizio, dell'utente che l'ha caricato, una breve descrizione dello stesso e il prezzo sia ad un utente autenticato che non	N.A.	R0F6.1
TS 6.2	Viene verificato che se l'utente è autenticato la ricerca ritorna più informazioni riguardanti il microservizio	N.A.	R0F6.5
TS 6.3	Viene verificato che il sistema permetta ad un utente autenticato di acquistare un'API utilizzando il saldo di APICredits in suo possesso	N.A.	R0F6.6
TS 7.1	Viene verificato che il sistema associ ad ogni API un'APIKey ad un utente autenticato che decide di acquistarla	N.A.	R0F7
TS 7.2	Viene verificato che il sistema permetta di far visualizzare all'utente tutte le APIKey che possiede in quel momento	N.A.	R0F7.3
TS 8	Viene verificato che il sistema permetta di visitare le pagine utente a chiunque	N.A.	R1F8

7.2 Test di integrazione

I test di integrazione vengono fatti per verificare che le varie componenti all'interno del sistema comunichino correttamente tra di loro e che i risultati che il sistema ritorna siano i risultati attesi. Si è deciso di utilizzare una strategia di integrazione incrementale in modo che sia possibile, nel caso in cui vengano ritornati errori, ripercorrere a ritroso i test per ritornare ad uno stato corretto. Per la formulazione di questi test è stato utilizzato il metodo bottom-up per poter integrare prima con le parti con minore dipendenza funzionale e maggiore funzionalità. Ci siamo concentrati per prima cosa suoi requisiti obbligatori, in questo modo i vincoli obbligatori saranno i primi ad essere soddisfatti, successivamente si passerà a testare i vincoli desiderabili ed infine quelli opzionali.

7.2.1 TI.model.search.SearchDataResult

- Descrizione: Verifica che la ricerca visualizzi correttamente i risultati cercati;
- Componente: com.apim.client.model.search.SearchDataResult;
- Stato: N.A.

7.2.2 TI.model.SessionToken

- Descrizione: verifica che il token di sessione esista e scada quando previsto;
- Componente: com.apim.client.model.SessionToken.
- Stato: N.A.

7.2.3 TI.model.modelView.SysLogin

- Descrizione: verifica che la login sia sicura e che restituisca gli errori opportuni;
- Componente: com.apim..client.modelView.navigation.SysLogin;
- Stato N.A.

7.2.4 TI.model.modelView.recovery

- Descrizione: verifica che il processo di recupero della password avvenga in modo corretto;
- Componente: com.apim..client.modelView.navigation.SysRecovery;
- Stato N.A.

7.2.5 TI.model.modelView.ShowUserPage

- Descrizione: verifica che la pagina utente venga visualizzata in modo corretto;
- Componente: com.apim..client.modelView.user.ShowUserPage;
- Stato N.A.

7.2.6 TI.model.modelView.ShowBuyCredits

- Descrizione: verifica che l'acquisto di APICredits avvenga in modo corretto;
- Componente: com.apim..client.modelView.user.ShowBuyCredits;
- Stato N.A.

7.2.7 TI.model.modelView.ShowApiPage

- Descrizione: verifica che la pagina amministrativa dell'API sia visualizzata in modo corretto;
- Componente: com.apim..client.modelView.user.AhowApiPage;
- Stato N.A.

7.3 Test di integrazione back-end

Per i test di integrazione della parte back-end$_g$ si è deciso di raggrupparli per microservizio per rendere anche la descrizione più comprensibile.

Test	Descrizione	Stato
TI 1.1	Viene verificato che le classi che saranno incorporate dal microservizio Controller funzionino correttamente quando utilizzate da un Driver	N.A.
TI 1.2	Viene verificato che il microservizio Controller funzioni correttamente quando incorpora delle classi Stub	N.A.
TI 1.3	Viene verificato che le classi incorporate dal microservizio Controller ed il microservizio stesso funzionino correttamente una volta integrate	N.A.
TI 2.1	Viene verificato che le classi che saranno incorporate dal microservizio UserManager funzionino correttamente quando utilizzate da un Driver	N.A.
TI 2.2	Viene verificato che il microservizio UserManager funzioni correttamente quando incorpora delle classi Stub	N.A.
TI 2.3	Viene verificato che le classi incorporate dal microservizio UserManager ed il microservizio stesso funzionino correttamente una volta integrate	N.A.
TI 3.1	Viene verificato che le classi che saranno incorporate dal microservizio KeyManager funzionino correttamente quando utilizzate da un Driver	N.A.
TI 3.2	Viene verificato che il microservizio KeyManager funzioni correttamente quando incorpora delle classi Stub	N.A.
TI 3.3	Viene verificato che le classi incorporate dal microservizio KeyManager ed il microservizio stesso funzionino correttamente una volta integrate	N.A.
TI 4.1	Viene verificato che le classi che saranno incorporate dal microservizio UserManager funzionino correttamente quando utilizzate da un Drive	N.A.
TI 4.2	Viene verificato che il microservizio UserManager funzioni correttamente quando incorpora delle classi Stub	N.A.
TI 4.3	Viene verificato che le classi incorporate dal microservizio UserManager ed il microservizio stesso funzionino correttamente una volta integrate	N.A.
TI 5.1	Viene verificato che le classi che saranno incorporate dal microservizio Gateway funzionino correttamente quando utilizzate da un Driver	N.A.
TI 5.2	Viene verificato che il microservizio Gateway funzioni correttamente quando incorpora delle classi Stub	N.A.
TI 5.3	Viene verificato che le classi incorporate dal microservizio Gateway ed il microservizio stesso funzionino correttamente una volta integrate	N.A.
TI 6.1	Viene verificato che le classi che saranno incorporate dal microservizio Analyzer funzionino correttamente quando utilizzate da un Driver	N.A.
TI 6.2	Viene verificato che il microservizio Analyzer funzioni correttamente quando incorpora delle classi Stub	N.A.
TI 6.3	Viene verificato che le classi incorporate dal microservizio Analyzer ed il microservizio stesso funzionino correttamente una volta integrate	N.A.

A Appendice

A.1 Risultati al termine delle consegne

In questa ultima sezione verranno riportati i risultati che vengono misurati dalle metriche definite nel documento *"Norme di Progetto v2.0.0"* inoltre verranno contestualizzati secondo i range ottimali e di accettazione definiti in questo documento.

A.2 Analisi e analisi in dettaglio

In questa fase è stata prodotta solo documentazione riguardante il progetto perciò saranno solo presenti esclusivamente metriche e misurazioni riguardanti i documenti.

A.2.1 Risultati dell'indice di Gulpease

Di seguito è riportata una tabella che contiene i documenti con il rispettivo valore che è risultato dall'indice di Gulpease:

Nome documento	**Valore Indice**
Analisi dei Requisiti	55
Norme di Progetto	48
Piano di Progetto	45
Piano di Qualifica	49
Studio di Fattibilità	43

Ogni documento rientra nel range di accettazione, ma solo l'AdR rientra in quello ottimale. Una possibile spiegazione a questo fatto potrebbe essere che i termini utilizzati nei documenti sono molto specifici all'ambiente informatico e quindi il documento in se risulta difficilmente comprensibile per un lettore che non sia abituato a questi termini.
Ciò nonostante il gruppo si ritiene soddisfatto dei risultati ottenuti dall'indice anche se non hanno portato ad un range ottimale, considerando che alcuni dei documenti verranno modificati durante il procedere del progetto, i dati ottenuti possono migliorare e questo è ciò che il gruppo si impone di migliorare.

A.2.2 Risultati PDCA

Analizzando il Registro delle modifiche dei documenti rispetto ai grafici di Gantt riguardante la pianificazione di questa fase, presenti nel *Piano di Progetto v.1.0.0*, è possibile notare che si è riusciti a rientrare nelle tempistiche per le consegne previste, questo è stato possibile grazie al fatto che durante la progettazione sono state fatte delle approssimazioni in eccesso delle tempistiche per raggiungere le milestone$_g$, in questo modo se fossero emersi dei problemi, ci sarebbe stato del tempo in più da utilizzare come cuscinetto per non far ritardare i processi successivi.

A.3 Progettazzione Architetturale

Anche in questa fase non è stato prodotto codice, ma sono solo stati stesi documenti, ciononostante sono state gettate le basi per il controllo del codice attraverso la pianificazione dei test.

A.3.1 Risultati dell'indice di Gulpease

Di seguito è stata riportata la tabella dei documenti che sono stati modificati in questa fase:

Nome documento	Valore Indice
Analisi dei Requisiti	60
Norme di Progetto	60
Piano di Progetto	59
Piano di Qualifica	54
Specifica di progetto	56

Rispetto alla precedente fase si può notare come tutti i valori siano tutti superiori ai loro valori precedenti, inoltre ora tutti i documenti sono oltre la soglia di ottimalità. Questo è stato possibile grazie anche ad una attenta cura dei termini utilizzati ed inoltre anche ad un controllo più scupoloso dei documenti

A.3.2 Risultati PDCA

a differenza dell'indice precedente purtroppo in questa fase non si ha avuto lo stesso risultato, questo però non è stato dovuto ad una erronea fase di Plan, bensì a problemi dei singoli membri che hanno avuto degli impegni imprevisti durante il corso di questa fase che hanno portato a slittare tutto il lavoro alla fine. per il futuro si cercherà di marginare gli imprevisti che potranno accadere sfruttando anche l'esperienza guadagnata in questa fase.