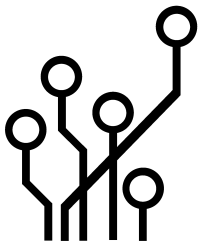


PROJECT ORIGIN

projectorigin2023@gmail.com

Norme di Progetto

Versione	0.7.1
Responsabile	Bobirica Andrei Cristian
Redattori	Andreetto Alessio
Verificatori	
Uso	Interno
Destinatari	*Project Origin* Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione
Questo documento descrive le norme di progetto del gruppo *Project Origin* nella realizzazione del progetto *Personal Identity Wallet*

Registro delle modifiche

Vers.	Data	Autore	Ruolo	Descrizione
0.7.1	2023-06-06	Andreetto Alessio	Analista	Ampliamento Capitolo 2.2.3
0.7.0	2023-05-23	Bobirica Andrei	Verificatore	Verifica$_g$Documento e Aggiornamento Capitolo 2.2.2.5
0.6.1	2023-05-18	Bobirica Andrei	Analista	Stesura Capitolo 3.1.3
0.6.0	2023-05-16	Andreetto Alessio	Verificatore	Verifica$_g$Documento
0.5.1	2023-05-15	Bobirica Andrei	Analista	Aggiornamento Capitolo 3, stesura 2.1.5
0.5.0	2023-05-10	Bobirica Andrei	Verificatore	Verifica$_g$Documento e Aggiornamento Capitolo 3.2.1
0.4.0	2023-05-08	Bobirica Andrei	Verificatore	Verifica$_g$Documento e Aggiornamento Capitolo 3.3.2 e 3.1.5
0.3.1	2023-05-04	Ibra Elton	Analista	Aggiornamento Processi Organizzativi
0.3.0	2023-05-04	Ibra Elton Corbu Teodor	Verificatore	Verifica$_g$Documento
0.2.1	2023-05-04	Ibra Elton	Verificatore	Aggiornamento Documento
0.2.0	2023-05-01	Andreetto Alessio Ibra Elton	Verificatore	Verifica$_g$Documento
0.1.1	2023-04-30	Corbu Teodor Bobirica Andrei	Analista	Stesura § Processi primari
0.1.0	2023-04-29	Andreetto Alessio	Verificatore	Verifica$_g$Documento
0.0.4	2023-04-27	Corbu Teodor	Analista	Stesura § Processi Organizzativi
0.0.3	2023-04-26	Corbu Teodor Bobirica Andrei	Analista	Stesura § Processi di Supporto
0.0.2	2023-04-26	Corbu Teodor Bobirica Andrei	Analista	Stesura § Introduzione
0.0.1	2023-04-26	Corbu Teodor Bobirica Andrei	Analista	Creazione struttura documento

Indice

1 Introduzione

1.1 Scopo del documento

Il presente documento mira a stabilire le direttive, le consuetudini e le procedure che tutti i membri di Project Origin dovranno rispettare per formalizzare un metodo di lavoro univoco per l'intero progetto. Ogni membro avrà l'obbligo di consultare periodicamente il documento e di aderire a tutte le normative ivi presenti, allo scopo di conseguire documenti uniformi e semplificare le operazioni di verifica$_g$.
La redazione sarà basata su un'ottica incrementale, quindi il documento attualmente in fase di elaborazione è incompleto e le norme saranno definite gradualmente, iniziando dalle più urgenti, con la finalità di stabilire un processo$_g$standardizzato prima dell'avvio del progetto, tenendo presente che ogni norma potrebbe essere soggetta a modifiche.
Inoltre, eventuali cambiamenti o integrazioni al documento dovranno essere ratificati dall'intero gruppo per garantire la coerenza e l'efficacia$_g$del metodo di lavoro comune adottato.

1.2 Descrizione del prodotto

1.2.1 Riferimenti

- *Presentazione del capitolato*:
  https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C3.pdf
- Standard ISO/IEC 9126$_g$:
  https://it.wikipedia.org/wiki/ISO/IEC_9126

2 Processi primari

2.1 Fornitura

2.1.1 Descrizione

Date le richieste del proponente, i processi di fornitura hanno come scopo di determinare e descrivere le attività e i compiti che il fornitore si impegna a svolgere. Le seguenti indicazioni dovranno essere seguite durante tutto il ciclo di vita del progetto didattico.

2.1.2 Obiettivi

Gli obiettivi principali che si cerca di raggiungere definendo i processi di fornitura sono:

1. Evitare la creazione di eventuali dubbi durante il progetto;
2. definire i bisogni che il prodotto deve soddisfare;
3. calcolare i costi per la realizzazione del prodotto;
4. definire le procedure necessarie per gestire il progetto;
5. ottenere il responso del committente e del proponente sulle decisioni prese e sulle attività svolte.

Una volta individuate le attività da svolgere, queste, insieme alle tempistiche, saranno definite nel documento *Piano di Progetto*.

2.1.3 Inizio dei lavori

La prima attività svolta è stata un'analisi dei capitolati delle aziende proponenti, lo studio di questi capitolati è stato esposto nei documenti *Analisi dei capitolati*.
Per presentare la nostra candidatura il gruppo ha effettuato uno studio di fattibilità che prevede come informazioni principali la stima dei costi degli orari di ogni ruolo necessario, insieme alla stima dei costi complessiva e la definizione degli impegni di ogni membro del gruppo.
Una volta ricevuta l'aggiudicazione per il capitolato$_g$scelto il gruppo ha iniziato a definire il proprio *Way Of Working*.
Quest'ultimo è descritto attraverso questo documento, *Norme di Progetto*.
Come obiettivo principale il gruppo ha deciso di descrivere inizialmente i Processi di Supporto ed Organizzativi in quanto sono importanti per definire gli strumenti e le metodologie necessarie per iniziare il lavoro.

2.1.4 Repository

Le attività da effettuare comprenderanno un lavoro condiviso e il codice prodotto sarà di pubblico dominio. Pertanto il gruppo ha deciso di utilizzare una repository$_g$pubblica su GitHub$_g$, piattaforma conosciuta da tutti i membri del gruppo.
La visione dei documenti creati è pubblica e solo i membri del gruppo hanno i permessi necessari per contribuire all'interno del progetto.
I file che sono soggetti a versionamento includeranno all'interno della loro struttura la numerazione della versione corrente e un riepilogo delle versioni precedenti sotto forma di tabella. In questo modo, è possibile tenere traccia dei cambiamenti apportati nel tempo al file e fornire un'indicazione chiara della sua storia. È stata creata un'organizzazione di nome Project Origin, i riferimenti sono `https://github.com/Project-Origin-2023`. Al suo interno esiste la repo *docs* contenente tutta la documentazione di interesse al proponente e al committente.

2.1.5 Pianificazione

2.1.5.1 Piano Di Progetto

Per favorire la pianificazione è stato creato il documento *Piano Di Progetto* il quale descrive la pianificazione delle attività del gruppo.
L'obiettivo di questa attività è di fornire una stima dei costi e dei tempi di consegna del progetto, nonchè un piano dettagliato corredato di orari per ogni ruolo coinvolto.
La pianificazione sarà basata sull'analisi periodica delle attività svolte e una retrospettiva su quelle attività già completate, al fine di mantenere aggiornate e valide le stime effettuate.
Vengono inoltre identificate le possibili problematiche che il team potrebbe incontrare durante l'intero periodo di svolgimento del progetto.

2.1.5.2 Gestione della Qualità

Questa attività riguarda le strategie di verifica e validazione che il gruppo si impegna ad utilizzare.
Per garantire efficacia e qualità nei processi e nei prodotti il gruppo ha realizzato il documento *Piano Di Qualifica* il quale descrive l'insieme delle risorse atte al perseguimento della qualità.
In particolare definisce le metodologie che il gruppo intende attuare per garantire la qualità del prodotto per tutta la durata del progetto.
La scelta di queste metodologie è a carico dei progettisti.

2.2 Sviluppo

2.2.1 Scopo

Il processo$_g$di sviluppo ha come obiettivo di descrivere le attività di analisi, di progettazione e di codifica per il prodotto software da sviluppare.
L'obiettivo più importante è la realizzazione del prodotto finale, attraverso l'individuazione dei requisiti del prodotto, la descrizione e l'individuazione degli eventuali vincoli da rispettare.

2.2.2 Analisi dei requisiti

2.2.2.1 Scopo

L'attività di analisi dei requisiti si pone come obiettivo la creazione di un documento contenente tutti i requisiti richiesti dal proponente. Lo scopo è di identificare i requisiti obbligatori e facilitare l'attività di pianificazione della mole di lavoro.
Gli Analisti hanno il compito di creare il documento di Analisi dei Requisiti per:

- Definire lo scopo del lavoro;
- Fornire ai Progettisti indicazioni precise e affidabili;
- Stabilire le funzioni e le richieste concordate con il cliente;
- Offrire una base per miglioramenti successivi per garantire un costante avanzamento del prodotto e del processo di sviluppo;
- Agevolare la revisione del codice;
- Fornire ai Verificatori$_g$indicazioni sui principali e alternativi casi d'uso per l'attività di test;
- Calcolare i costi del progetto.

2.2.2.2 Casi d'uso

I casi d'uso descrivono l'interazione tra il sistema e uno o più attori. Ogni caso d'uso è univoco, e si distingue con un codice che segue la seguente sintassi: "UC[Codice] - [Titolo]", dove "UC" sta per "Use Case", "[Codice]" indica un numero intero univoco per ogni caso d'uso, e "[Titolo]" è il titolo del caso d'uso. Ogni caso d'uso può includere informazioni aggiuntive come:

- Scenario principale
- Scenari alternativi
- Pre-condizioni
- Post-condizioni
- Trigger
- Attori principali
- Attori secondari
- Diagramma UML

2.2.2.3 Requisiti

I requisiti sono ricavati attraverso l'analisi del capitolato$_g$insieme a successive discussioni con il proponente e attraverso l'analisi dei casi d'uso. Possono essere individuati sia tramite discussioni approfondite all'interno del gruppo che attraverso discussioni con il proponente. I requisiti possono essere categorizzati in diverse tipologie :

- **Funzionali**: Requisiti dati dalle funzionalità che il prodotto deve presentare;
- **Qualità**: Requisiti dati per descrivere la qualità che il prodotto deve avere;
- **Vincolo**: Requisiti dati da vincoli tecnologici e strettamente legati all'implementazione del prodotto.

I requisiti possono essere categorizzati in base a diversi gradi d'importanza:

- **Desiderabile**: Rappresenta un requisito che apporta valore e miglioramenti al sistema, ma non è strettamente necessario per il suo funzionamento base;
- **Opzionale**: Se relativamente vantaggioso, sarà preso in considerazione in un momento successivo;
- **Obbligatorio**: Denota un requisito essenziale per il cliente che non può essere trascurato.

2.2.2.4 Codifica dei requisiti

R[Tipologia][Numero]-[Importanza]
Di seguito sono elencate le sigle per la codifica dei requisiti:

- **F**: Funzionale; **N**: Non Funzionale; **Q**: Qualità; (**Tipologia**)
- **UC**[codice]; (vedi paragrafo 2.2.2.2) (**Casi d'uso**)
- **O**: Obbligatorio; **D**: Desiderabile; **P**: Opzionale. (**Importanza**)

2.2.2.5 Codifica dei rischi

R[Tipologia][Numero] - Rischio [Tipologia] [Numero]
Di seguito sono elencate le sigle per la Tipologia:

- **T**: Tecnologico;
- **O**: Organizzativo;
- **R**: Requisito;
- **I**: Interpersonale

Il Numero è un contatore inizializzato a 1 e che incrementa ad ogni rischio individuato.

2.2.3 Progettazione

I progettisti sono responsabili dell'attività di progettazione, la quale consiste nell'ideazione dell'architettura del sistema sulla base dei requisiti identificati nell'Analisi dei Requisiti.
Per raggiungere questo obiettivo, il processo di progettazione prevede la realizzazione di un Proof of Concept$_g$per la **R**equirements and **T**echnology **B**aseline, seguito da una fase di approfondimento dei dettagli per la Product Baseline.

2.2.3.1 Introduzione a Docker e container

Docker è un programma che permette di creare sistemi virtualizzati e che useremo per creare ambienti di sviluppo isolati, minimali e facilmente distribuibili. Questi singoli ambienti vengono chiamati container e hanno l'obiettivo di semplificare il deploy delle applicazioni. A differenza dei classici modi di virtualizzare i sistemi operativi , docker si basa su "OS-Level virtualization" una funzionalità fornita dal kernel linux che si occupa di orchestrate l'isolamento e le limitazioni delle risorse. Risulta pertanto l'ambiente perfetto per lo sviluppo dei servizi di nostro interesse legati all'uso di api e comunicazione tra di essi.

2.2.3.2 Lavoro in locale con le componenti

Nel nostro progetto abbiamo deciso di sviluppare le applicazioni web in react; in particolare issuerapp e verifierapp saranno implementate in react mentre wallet sarà implementato in react native in maniera tale da avere un applicazione per i dispositivi mobile oltre che una webpp. Per queste componenti abbiamo deciso di lavorare in locale:

- Issuerapp,
- Verifierapp,
- Wallet.

Questa scelta è stata attuata al fine di garantire maggiori performance durante lo sviluppo, semplificare la modalità di modifica ed avere una visualizzazione in tempo reale delle modifiche fatte sul browser. Queste applicazioni se ospitate su docker infatti necessitavano ad ogni modifica del codice un deploy e un ricaricamento della pagina web per visualizzare l'effettivo avvenimento; andando così ad impattare a lungo termine le tempistiche di sviluppo. Queste componenti sono avverabili singolarmente o tutte assieme attraverso i seguenti comandi:

- Avvio singolo di wallet:
  - Aprire il terminale e posizionarsi sulla cartella tramite il seguente comando:

    ```
    cd /Personal-Identity-Wallet
    ```

  - Avviare il deploy richiamando lo script:

    ```
    sh deployWallet.sh
    ```

  - Lo script si posizionerà all'interno della cartella ./wallet, scaricherà le dipendenze necessarie alla webapp attraverso il commando

    ```
    npm install
    ```

    e avvierà l'ambiente di sviluppo react native attraverso il commando

    ```
    npm start.
    ```

  - L'app sarà disponibile al endpoint http://localhost:19000

- Avviso singolo di issuer:
  - Aprire il terminale e posizionarsi sulla cartella tramite il seguente comando:

    ```
    cd /Personal-Identity-Wallet
    ```

  - Avviare il deploy richiamando lo script:

    ```
    sh deployIssuerapp.sh
    ```

  - Lo script si posizionerà all'interno della cartella ./issuerapp, scaricherà le dipendenze necessarie alla webapp attraverso

    ```
    npm install
    ```

    e successivamente sarà quindi avviato l'ambiente di sviluppo del framework react attraverso il commando

    ```
    npm start
    ```

    sarà visibile al endpoint http/localhsot:19001 il frontend della applicazione.
  - Lo script avvierà anche l'istanza di node.js che esegue funzionalità serverside e sarà disponibile al endpoint http://localhost:19101

- Avvio singolo di verifier:
  - Aprire il terminale e posizionarsi sulla cartella tramite il seguente comando:

    ```
    cd /Personal-Identity-Wallet
    ```

  - Avviare il deploy richiamando lo script:

    ```
    sh deployVerifier.sh
    ```

  - Lo script si posizionerà all'interno della cartella ./verifierapp, scaricherà le dipendenze necessarie attraverso il commando

    ```
    npm install
    ```

    e avvierà l'ambiente di sviluppo del framework react con

    ```
    npm start.
    ```

  - Sarà quindi visualizzabile al endpoint http/localhsot:19002 il frontend della applicazione.

- Avvio delle 3 componenti simultaneamente:
  - Aprire il terminale e posizionarsi sulla cartella tramite il seguente comando:

    ```
    cd /Personal-Identity-Wallet
    ```

  - Avviare il deploy richiamando lo script:

    ```
    sh deployAll.sh
    ```

2.2.3.3 Considerazioni sulle componenti e sui processi

La parte dell'*Issuer* è composto oltre che da un interfaccia web frontend di react anche da una parte serverside in node.js. *Wallet*, essendo come sopra citato realizzata con react native, ad ogni modifica del codice la pagina per la visualizzazione web deve essere aggiornata, richiedendo quindi al ambiente di sviluppo di rifare la build della webapp.
Issuerapp e *Verifierapp* sono realizzate con react e, tramite un sistema di watcher, le modifiche al codice vengono riconosciute automaticamente, la webapp viene ricompilata e la pagina del browser viene aggiornata in automatico con le modifiche apportate.
Avviando le webapp utilizzando gli script, in base al caso utilizzato, implica che si avrà un singolo processo in foregroung e gli altri saranno in background.
Può essere necessario avviarli manualmente e non utilizzando gli script. Per esempio l'*Issuer* ha la parte serverside in node.js ed essa ad ogni modifica deve essere fatto il deploy nuovamente, per fare questo si ricorda l'utilizzo per il frontend del commando

```
npm start
```

e per la parte serverside del commando

```
node index.js
```

Per riavviare il nodo serverside si preme Ctrl+C sul terminale e si riavvia con il commando

```
node index.js
```

Si ricorda che avendo dei processi in background essi devono essere spenti se si vogliono riaprire delle nuove istanze di essi, questo perchè occupano già le porte delle webapp.
Per spegnere tutte le istanze delle webapp si utilizza il commando

```
killall node
```

2.2.3.4 Lavoro su Docker

Per quanto riguarda i container docker nel nostro progetto utilizzeremo le seguenti componenti:

- 3 istanziazioni di Walt id:
  - verifierapi con endpoint http://localhost:19012
  - walletapi con endpoint http://localhost:19010
  - issuerapi con endpoint http://localhost:19011
- Un database per l'issuer, dbissuerapp, il DBMS è prostgresql
- Un tool di amministrazione del database che serve allo sviluppo detto adminer

Il database è configurato sulla porta standard 5432, all'interno della rete di docker ha come ip fisso 10.5.0.5 ed è disponibile anche da localhost.
La porta in entrambi i casi è la 5432. Le credenziali di accesso al DB dbissuerapp sono:

- username: admin
- password: admin
- db: issuerapp
- server: localhost:5432

2.2.3.5 Comandi componenti Docker

Per fare il deploy di queste componenti docker:

- Aprire il terminale e posizionarsi sulla cartella tramite il seguente comando:

```
cd /Personal-Identity-Wallet
```

- Avviare le componenti tramite il comando:

```
docker-compose up -d
```

2.2.3.6 Considerazioni componenti Docker

Il comando

```
docker-compose up -d
```

prende le configurazioni all'interno del file docker-compose.yaml e crea le istanze dei container con le nuove configurazioni.
Se il container non subisce modifiche esso non viene intaccato dal commando, se non esiste viene creato, se presenta modifiche viene ricreato.
Il funzionamento dei container si basa su un'immagine e una istanza di essa detto container, può essere necessario in alcuni momenti legati al mantenimento di eliminare anche le immagini per ricreare un ambiente di sviluppo con la giusta configurazione.
La memoria di archiviazione nei container può avvenire in due maniere:

- se il container presenta volumi, i file saranno memorizzati in locale,
- se non presenta volumi saranno memorizzati nella istanza del container.

Una volta ricreato il container si perdono anche i file memorizzati al suo interno, a meno che essi non siano memorizzati in un volume oppure all'interno di una copia dell'immagine. Una volta terminato lo sviluppo in vista del POC il progetto (composto dai file htm,css e js) verrà trasportato in un container docker adibito a server http (es. Apache), la parte serverside del Issuerapp sarà trasportata su un container di Node.js

2.2.4 Codifica

- **Indentazione**: I blocchi di codice innestati e le varie linee di codice dovranno rispettare una determinata tabulazione al fine di rendere il codice più leggibile.
- **Blocco di codice**: La delimitazione dei vari blocchi di codice tra parentesi graffe dovranno avere una struttura determinata dove le parentesi graffe saranno poste nella seguente maniera:
  - **Parentesi di apertura**: Posta a distanza di 1 spazio rispetto alla dichiarazione del costrutto ma sempre nella stessa linea.
  - **Parentesi di chiusura**: Posta nella linea successiva dell'ultima riga di codice del blocco a cui si fa riferimento e indentata secondo il blocco.
- **Commenti**: I commenti saranno inseriti in lingua italiana, saranno indispensabili nelle porzioni di codice dove non è immediata la comprensibile. Ogni qualvolta il codice venga aggiornato va ricontrollata la validità dei commenti ad esso associato.
- **Nomi univoci**: È necessario utilizzare nomi univoci e significativi per ogni costruttore al fine di garantire la chiarezza e la facilità di identificazione delle diverse entità.

3 Processi di Supporto

3.1 Documentazione

3.1.1 Scopo

Sarà necessario documentare ogni fase e attività coinvolta nello sviluppo del progetto.
Questa sezione illustrerà le regole e gli standard da seguire durante la produzione dei documenti per l'intero ciclo di vita del software.
Qui esamineremo approfonditamente i processi adottati per la stesura, la verifica$_g$, il mantenimento e l'approvazione di tutta la documentazione prodotta dal nostro team.
Lo scopo è di fornire una descrizione accurata di tutte le regole, le convenzioni e i limiti che ci impegniamo a rispettare per ottenere una documentazione efficace, coerente e formale.

3.1.2 Documenti prodotti

- Norme di progetto;
- Glossario;
- Piano di progetto;
- Piano di Qualifica;
- Analisi dei requisiti;
- Verbali.

3.1.3 Documenti tecnici

Durante il progetto sarà necessario fare diversi lavori di ricerca e di studio su standard, tecnologie e protocolli da utilizzare.
Per facilitare questa azione gli analisti incaricati dovranno redarre dei documenti tecnici, categorizzati come documenti interni.
Questi documenti hanno come obiettivo il riassumere brevemente e spiegare i concetti chiave di un detterminato argomento citando le fonti e preferendo un approccio schematico.
Questi documenti dovranno anch'essi essere verificati, successivamente dovranno essere letti da tutti i membri del gruppo in quanto le informazioni al loro interno sono destinate a tutti i membri.

3.1.4 Ciclo di vita di un documento

- **Creazione Struttura**: A partire da un template comune viene creata la struttura del documento inserendo il registro delle modifiche e l'indice dei contenuti;
- **Stesura**: Durante questa fase i redattori producono le sezioni assegnate a loro dal Responsabile$_g$di Progetto aggiornando progressivamente il documento;
- **Approvazione**: Quando un redattore ha terminato le sue modifiche il Responsabile del progetto assegna ad un Verificatore$_g$il compito di esaminare ed approvare le modifiche. Il documento viene pubblicato se tutte le modifiche hanno ricevuto l'approvazione da parte dei Verificatori e successivamente dal Responsabile del Progetto.

3.1.5 Verbali

I verbali sono suddivisi in Interni ed Esterni.
Dovranno contenere le seguenti informazioni:

- Motivo della riunione;
- Luogo della riunione (se in presenza o da remoto);
- Data, ora e durata della riunione;
- Partecipanti della riunione;
- Resoconto della riunione con le decisioni intraprese.

3.1.6 Strumenti utilizzati

Gli strumenti utilizzati per la stesura dei documenti sono:

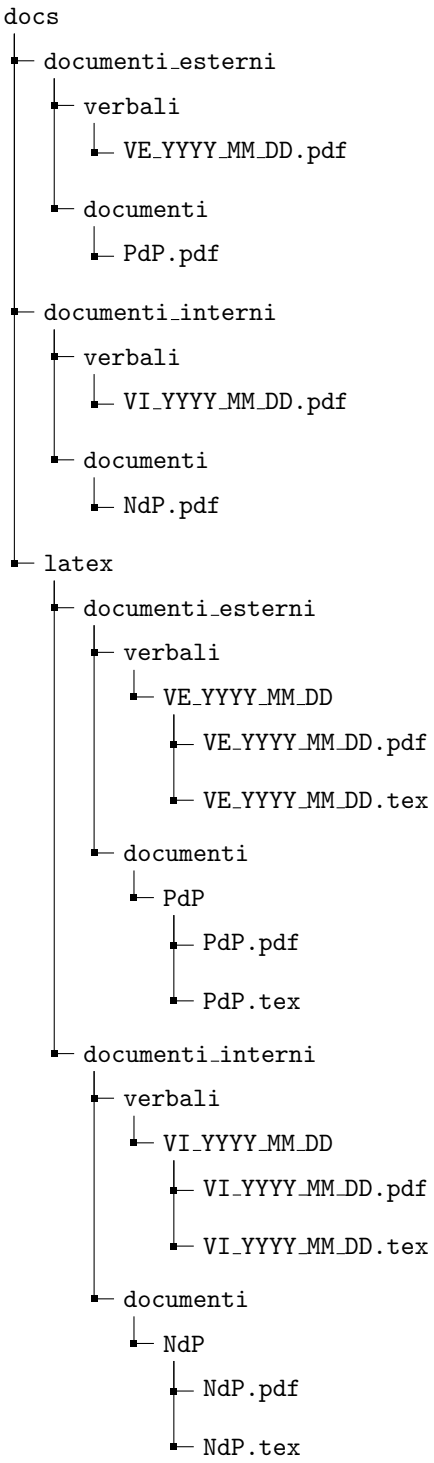
- **LATEX**: Tutti i documenti prodotti dal gruppo verranno redatti usando il linguaggio di markup LateX;
- **Visual Studio Code con l'estensione Latex Workshop**: Il gruppo adotterà questo strumento per scrivere i documenti e per creare i file;
- **Draw.io**: Sito online per la creazione di grafici UML;
- **GanttProject**: Programma usato per la realizzazione di diagrammi di Gantt.

Per semplificare le operazioni di verifica$_g$e di lettura della documentazione dovrà essere reso disponibile il documento completo in formato PDF.

3.2 Directory Documenti

3.2.1 Struttura delle directory e dei file

I documenti saranno archiviati con la seguente struttura della directory:



I documenti saranno redatti in formato Latex e la loro stesura averrà all'interno della cartella `docs/latex`. Saranno suddivisi nelle rispettive cartelle `documenti_esterni` e `documenti_interni` e per ciascuna

categoria ripartiti nelle cartelle `verbali` e `documenti`.
Un documento è archiviato nella sua rispettiva cartella `nomeDocumento` dove al suo interno si può trovare il file Latex `nomeDocumento.tex`, `nomeDocumento.pdf` con il file compilato, la cartella `res` contenente file di risorse del documento come allegati e le sezioni utilizzate, ed infine la cartella `config` contenente la configurazione specifica del documento che instanzia il template.
La cartella `docs/template` contiene i file creati appositamente per avere un template utilizzabile per tutti i documenti.
Per facilitare la lettura e separare i file Latex e le loro rispettive dipendenze si è deciso di creare una struttura directory parallela contenente solo i file PDF.
Le rispettive cartelle sono `docs/documenti_esterni` e `docs/documenti_interni` anch'esse individualmente ripartite in documenti e verbali.
Un documento sarà quindi archiviato per esempio come
`docs/documenti_interni/documenti/nomeDocumento.pdf`.
Si è deciso di mantenere i documenti in formato PDF sia nella cartella latex che nelle cartelle
`docs/documenti_esterni` e `docs/documenti_interni`.
Per ovviare al ovvio problema della inconsistenza dei file PDF ed evitare che qualcuno legga un documento non aggiornato si è attuato una azione che mantiene la consistenza e la coerenza dei file PDF spiegata di seguito.

3.2.2 Github Actions copyONPush.yaml

Si è utilizzata lo strumento GitHub Actions, in particolare esiste uno script contenuto nella cartella
`.github/workflows/copyOnPush.yaml` che ad ogni push copia e sovrascrive i file PDF dalla cartella
`docs/latex` nelle rispettive cartelle parallele.
Utilizzando questo strumento si ha sempre i file aggiornati e non è necessario copiarli manualmente, i file vengono spostati automaticamente dalla cartella latex nelle cartelle dedicate solo ai file PDF.
Per questo motivo è essenziale spiegare che è dovere solo del script copiare i file PDF, i membri del gruppo non devono spostare file PDF nelle cartelle dedicate, essi vengono spostati solo e solamente attraverso lo script.
Tutti i file Latex all'interno della cartella `docs/latex` devono stare alla stessa altezza della gerarchia, questo per non creare problemi legati alla incompatibilità con i percorsi URL nel file di template. Nelle cartelle parallele dedicate solo ai PDF tuttavia si è deciso che è possibile creare delle cartelle contenitori per ripartire diversi documenti. Essi sono spostati automaticamente dallo script. I file legati alla candidatura iniziale sono stati ragruppati dallo script dentro la sottocartella
`docs/documenti_esterni/documenti/candidatura`.
I file legati a questioni strettaente tecniche sono stati raggruppati dallo script dentro la sottocartella
`docs/documenti_interni/documenti/documenti_tecnici`.

3.3 Forma Documenti

3.3.1 Struttura Documenti

La prima pagina è composta dal logo del gruppo, email di riferimento, titolo del documento. Sotto si può trovare una struttura indicante la versione attuale, il responsabile che ha approvato il documento, i redattori e i verificatori che hanno contribuito al documento, l'uso che ne si fa e i destinatari del documento. In basso si trova una descrizione breve del documento. Tutte queste informazioni sono istanziabili per ogni documento attraverso il file di configurazione `titlePageInput.tex`. In qualsiasi pagina del documento si può trovare il logo del gruppo in alto a sinistra, il nome del documento in alto a destra ed in fine a piè di pagina si può trovare la pagina corrente con il numero totali di pagine.

3.3.2 Registro delle modifiche

All'inizio del documento deve essere presente una tabella riassuntiva della cronologia delle versioni del documento dove sono specificate le modifiche apportate. Nella tabella ogni riga corrisponde ad una modifica apportata, mentre le colonne sono le seguenti:

- **Versione**: Versione del documento dopo la modifica;
- **Data**: Data della modifica;
- **Autore**: Nome dell'autore della modifica o della azione apportata;
- **Ruolo**: Redattore nel caso di modifiche del documento, Verificatore$_g$per chi verifica$_g$le modifiche apportate e Responsabile$_g$per l'approvazione finale da parte del Responsabile del Progetto;
- **Descrizione**: Breve descrizione della modifica apportata.

3.3.3 Forma Tipografica

- **Nomi dei documenti**: I nomi dei documenti iniziano sempre con la lettera minuscola. Se presenti più parole, queste saranno attaccate ma distinguibili dalla lettera maiuscola (convenzione "CamelCase");
- **Nomi dei verbali**: I verbali avranno la seguente struttura `V[Tipologia]_[YYYY]_[MM]_[DD]` con la tipologia che può essere Interno `I` oppure Esterno `E`.

3.4 Versionamento

I documenti dovranno supportare il versionamento, in modo da permettere l'accesso ad ogni singola versione prodotta durante il loro ciclo di vita.
Il formato identificativo del versionamento dovrà seguire questa forma:
v[X].[Y].[Z]
Il valore "X" rappresenta la versione pubblicamente rilasciata e approvata dal responsabile del progetto. Questa numerazione inizia da zero e viene incrementata ad ogni nuova versione approvata.
Il valore "Y" indica una revisione da parte del Verificatore effettuata per assicurarsi che, dopo l'implementazione di una modifica, il prodotto sia ancora integro e coerente. La numerazione inizia anch'essa da zero e viene azzerrata ogni volta che viene incrementato il valore "X".
Il valore "Z" viene incrementato ad ogni modifica del prodotto. Anche questa numerazione inizia da zero e viene azzerata ogni volta che viene incrementato il valore "X" o "Y".

4 Processi Organizzativi

4.1 Gestione di processo

Secondo lo standard ISO-12207:1995, questa gestione contiene attività e compiti generici, tra cui la definizione dell'obiettivo del processo$_g$, la pianificazione e la stima dei tempi, delle risorse e dei costi, l'assegnazione di compiti e responsabilità, l'esecuzione e il controllo delle attività, la revisione e valutazione delle attività svolte e la determinazione della fine del processo.
Gli obiettivi della gestione di processo includono la semplificazione e la gestione della comunicazione tra i membri del gruppo e con l'esterno, la coordinazione dell'assegnazione dei ruoli e dei compiti, il monitoraggio del lavoro del gruppo e la pianificazione delle attività da svolgere, nonché la definizione delle linee guida generali per la formazione dei membri.
Lo scopo della gestione di processo è l'istanziazione dei processi di progetto, la stima dei costi e delle risorse necessarie per la loro esecuzione, la pianificazione delle attività e dei task ad essi associati, l'assegnazione del personale e il controllo e la verifica$_g$delle attività dei processi di progetto durante la loro esecuzione. Tali attività mirano a garantire la coerenza e la coesione del prodotto, verificando che sia sempre integro e conforme agli obiettivi iniziali.

4.2 Comunicazione

Le comunicazioni all'interno del progetto avvengono su due livelli distinti: tra i membri del gruppo e tra i membri del gruppo e soggetti esterni. I soggetti esterni identificati sono l'azienda Infocert in qualità di proponente e i professori Tullio Vardanega e Riccardo Cardin come committenti. Per quanto riguarda le comunicazioni interne, il principale mezzo utilizzato è Telegram$_g$. Le videochiamate interne, come le riunioni di progetto, si svolgono principalmente su Discord$_g$, scelto per la sua semplicità e versatilità multi-piattaforma. In alternativa, può essere utilizzato Zoom$_g$. Tramite un accordo con l'azienda si è deciso di comunicare nella piattaforma Teams(Microsoft)$_g$. Per quanto riguarda le comunicazioni esterne, è stato creato un indirizzo e-mail apposito, projectorigin2023@gmail.com, a cui tutti i membri del gruppo hanno accesso. Ogni membro è tenuto a controllare regolarmente la casella di posta elettronica e a notificare il gruppo in caso di nuovi messaggi. La stesura e l'invio dei messaggi è compito del Responsabile$_g$di Progetto, previa breve verifica$_g$e approvazione del gruppo. Tutte le e-mail verranno firmate con "Project Origin".

4.3 Pianificazione

4.3.1 Scopo

L'obiettivo della pianificazione è quello di creare un piano di progetto che sia un documento esterno, il cui contenuto descrive i seguenti aspetti:

- le risorse che sono disponibili;
- come queste risorse sono state assegnate alle varie attività dei processi di progetto;
- la sequenza temporale delle attività che verranno svolte.

4.3.2 Obiettivi

L'obiettivo principale della pianificazione del progetto è quello di redigere un Piano di progetto che permetta di:

- organizzare le attività in modo efficiente, in modo da raggiungere risultati efficaci;
- definire gli obiettivi dei processi e semplificare il monitoraggio del progresso attraverso l'individuazione di milestone temporali;

La struttura del Piano di progetto è così composta:

- introduzione e finalità;
- organizzazione generale del progetto;
- analisi dei rischi;
- risorse a disposizione;
- struttura di scomposizione del lavoro (WBS) delle attività dei processi istanziati;
- pianificazione del progetto (project schedule);
- strumenti di controllo e di reportistica.

4.3.3 Procedura

- **Identificazione delle attività sulla base dei requisiti da soddisfare**: È opportuno organizzare le attività individuate in un elenco. Le attività dei processi saranno decomposte mediante una struttura di scomposizione del lavoro, in una struttura gerarchica. I task ottenuti possono così essere identificati in modo univoco;
- **Identificazione ed analisi dei rischi**: I rischi considerati sono lo sforamento di costi, sforamento dei tempi e risultati insoddisfacenti. La gestione dei rischi è costituita da tre fasi in fase di pianificazione, ed un'ulteriore fase da iterare durante tutta l'esecuzione di un processo.
  Le tre fasi in fase di pianificazione sono:
  - **Individuazione dei rischi:** Si considerano tutti i possibili rischi che possono emergere a causa di fattori interni ed esterni. I rischi identificati vengono riportati in una tabella esplicativa;
  - **Analisi dei rischi individuati:** Ai rischi presenti nell'elenco è necessario associare una probabilità di occorrenza ed una stima dell'impatto sulla corretta esecuzione del processo;
  - **Pianificazione dei rischi individuati:** Le attività devono essere pianificate in modo tale da minimizzare sia la probabilità di occorrenza dei rischi che l'effetto che essi possono avere sul progetto. Le attività sono pianificate su compiti brevi in rapporto alla quantità di lavoro da svolgere (tempo/persona), al fine di ridurre al minimo l'impatto dei rischi associati.
- **Un insieme di attività coerenti è associato ad una specifica tappa temporale nel calendario.**
  Tuttavia, queste attività devono essere sufficientemente brevi o suddivise in sotto-attività brevi per consentire un controllo e una verifica$_g$più facili ed efficaci degli effetti sul prodotto associati all'incremento risultante, garantendo un maggiore controllo durante la fase di verifica$_g$sull'introduzione di eventuali errori nel prodotto complessivo.
  I fattori da considerare per minimizzare i rischi includono: la completezza dei requisiti, il coinvolgimento del cliente, un'opportuna allocazione delle risorse, la fondatezza delle aspettative, la presenza di supporto esecutivo, la corretta gestione della fluttuazione dei requisiti.
  Durante l'esecuzione del processo si misurano degli indicatori. Questo può portare a dover rivedere la pianificazione delle attività in corso.
  Riordinamento delle attività identificate in base alle dipendenze ingresso-uscita per comprendere, attraverso i diagrammi di Gantt: la sequenzialità temporale delle attività rispetto alle loro dipendenze, il possibile parallelismo tra le varie attività, come la durata effettiva di un'attività si sovrapponga alla durata pianificata, come le stime fatte corrispondano ai progressi, come ogni attività può essere associata al tempo di calendario, limitato superiormente dall'ultima scadenza contrattuale e discretizzato in unità di tempo/persona, il margine di slack assegnabile a ciascuna attività per poter ammortizzare più ritardi possibili.
- **Stima delle risorse da assegnare a ciascuna attività**: Sono considerate due tipologie di risorse: sforzo, tempo di calendario.

Come stimare le risorse: assegnando a ciascuna attività un valore, secondo la metrica tempo/persona, della quantità di lavoro necessaria per portarla a termine, identificando delle tappe temporali nel tempo di calendario e assegnando le attività alle corrispondenti tappe.
La pianificazione delle attività (cioè l'assegnazione del tempo/persona sul tempo di calendario) viene effettuata seguendo due criteri: il primo criterio è la pianificazione all'indietro per capire se sia possibile pianificare all'interno dei limiti di tempo di calendario imposti dalle scadenze contrattuali, sfruttando dove possibile la parallelizzazione delle attività. Applicata nella stesura del Piano di progetto v1.0.0 D. Si pianifica all'indietro a partire dalle tappe identificate nel calendario. Il secondo criterio è la pianificazione in avanti per rispettare le dipendenze ingresso-uscita delle attività. Applicata nella stesura del Piano di progetto v1.0.0 D e dinamicamente durante la gestione dei ticket.

- **Assegnazione delle risorse stimate ad ogni attività**: Ogni risorsa umana viene destinata ad un ruolo di progetto. Il personale quindi viene assegnato ai task associati al ruolo assunto, attraverso un sistema di segnalazione dei ticket su un repository$_g$. Il repository di segnalazione viene inizialmente popolato dai task derivati dalla WBS e viene dinamicamente arricchito e svuotato a mano a mano che emergono e vengono risolti i problemi e i task. E' importante che i task all'interno del repository siano ordinati in base alle dipendenze ingresso-uscita e in base ad una priorità loro assegnata.

4.3.4 Ruoli

Il processo$_g$di sviluppo software coinvolge molteplici figure professionali, ognuna con specifiche responsabilità. Tra queste figure troviamo l'Analista$_g$, il Progettista$_g$, il Programmatore$_g$, il Verificatore$_g$, il Responsabile$_g$e l'Amministratore di progetto.
L'Analista, presente soprattutto nella fase iniziale del progetto, si occupa di comprendere il problema e definire i requisiti espliciti ed impliciti.
Il Progettista, invece, effettua lo studio di fattibilità del prodotto e costruisce l'architettura, partendo dal lavoro svolto dall'Analista e perseguendo efficienza$_g$ed efficacia$_g$.
Il Programmatore si occupa di implementare le specifiche fornite dal Progettista, scrivendo codice orientato alla futura manutenzione e alla riusabilità.
Il Verificatore$_g$, invece, ha il compito di controllare ciò che viene prodotto dagli altri membri del team, individuando eventuali errori e segnalando al responsabile del prodotto analizzato.
Il Responsabile di progetto, figura fondamentale, rappresenta il team presso il committente e guida e coordina il team verso il raggiungimento degli obiettivi di progetto. Si occupa di prendere decisioni e approvare documenti, coordinare i membri del team e valutare i rischi e i costi, mantenendo le relazioni con i soggetti esterni e rispettando le scadenze e l'allocazione delle risorse.
Infine, l'Amministratore di progetto gestisce e controlla l'ambiente di lavoro, definendo le norme e le procedure alla base del lavoro, regolando le infrastrutture e i servizi utili per lo svolgimento dei processi, gestendo il versionamento dei prodotti e la loro configurazione, individuando strumenti utili a migliorare e/o automatizzare i processi e gestendo la documentazione di progetto.

4.4 Formazione dei membri del team

Lo scopo della formazione è di garantire che ogni componente del team abbia le competenze necessarie per svolgere con successo le proprie attività e che vi sia un costante aggiornamento delle conoscenze del personale nel tempo, in modo da mantenere un team altamente qualificato.

4.4.1 Formazione interna

È richiesto che ogni componente del team acquisisca le competenze necessarie per svolgere i compiti assegnati in modo autonomo, cercando di colmare eventuali lacune attraverso lo studio individuale. I membri più esperti sono incoraggiati a condividere le proprie conoscenze e risorse con il resto del gruppo. Nel caso in cui un membro riscontri difficoltà nell'esecuzione di un compito, è possibile rivolgersi al Responsabile$_g$di Progetto per richiedere supporto nell'organizzazione di attività di apprendimento.

Sebbene siano disponibili documenti di riferimento, è consigliabile integrare la formazione con materiali di approfondimento individuale, in quanto la documentazione fornita potrebbe non essere esaustiva.

4.4.2 Strumenti a supporto

Il processo di gestione organizzativa del team sarà supportato da una serie di strumenti che includono:

- **Telegram**$_g$: Una piattaforma di messaggistica utilizzata dal team per comunicazioni meno urgenti e decisioni di minor importanza;
- **Git**$_g$: Uno strumento di controllo versione utilizzato dal team per tenere traccia dei cambiamenti nei documenti;
- **GitHub**$_g$: Una piattaforma online utilizzata dal team per il controllo versione e per il salvataggio di tutti i file creati dai membri del team;
- **GitHub Issues**$_g$: Un sistema integrato in GitHub che consente la gestione dei ticket e la segnalazione dei problemi;
- **GitHub Actions**$_g$ è uno strumento fornito da GitHub che permette l'automazione di compiti di varia natura.
- **GitHub Desktop**$_g$: GitHub Desktop è un'applicazione gratuita e open source per Windows e Mac per gestire senza problemi i tuoi progetti, creare commit significativi e tenere traccia della cronologia del progetto in un'applicazione anziché nella riga di comando.
- **Google Drive**$_g$: Un servizio di condivisione di documenti che consente la collaborazione in tempo reale e la modifica condivisa;
- **Gmail**$_g$: Un servizio di posta elettronica scelto dal team per la comunicazione interna ed esterna;
- **Zoom**$_g$: Uno strumento che si può utilizzare per le riunioni virtuali tra i membri del team;
- **Discord**$_g$: Uno strumento standard per le riunioni tra i membri del team e per la comunicazione interna;
- **Teams (Microsoft)**$_g$: Uno strumento che si può utilizzare per le riunioni virtuali.