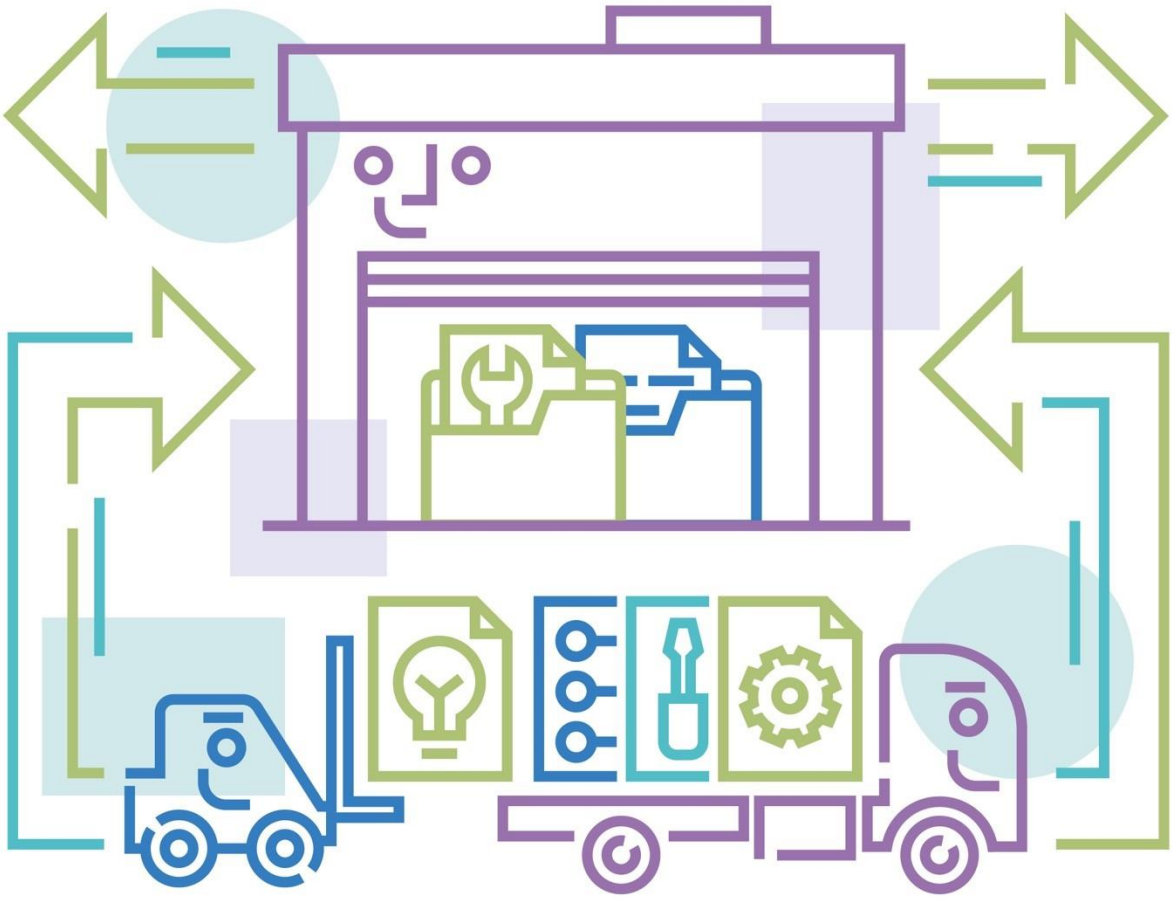
APPENDICE

Documenti di Approfondimento della Linea Guida

"La gestione dei Repository di una Open Comunità della PA"



UNIONE EUROPEA
Fondo Sociale Europeo
Fondo Europeo di Sviluppo Regionale

Agenzia per la Coesione Territoriale

PON GOVERNANCE E CAPACITÀ ISTITUZIONALE 2014-2020

Presidenza del Consiglio dei Ministri
Dipartimento per gli Affari Regionali e le Autonomie

AGID Agenzia per l'Italia Digitale

.0 puntozero
LA NUOVA VERSIONE DELL'INNOVAZIONE

Indice

2. MODELLO TECNOLOGICO DEL REPOSITORY LOCALE 4
2.1 Software configuration Management 5
2.2. Implementazione ambiente del KIT 6
2.2.1. Organizzazione per la gestione dell'Open Source 6
2.2.2. Organizzazione e items del KIT OCPA 9
2.2.3. Progetti GIT nel Repository 11
2.2.4. Accesso ai progetti del Repository 12
3. MODELLO DEL CATALOGO NAZIONALE "DEVELOPERS ITALIA" 14
3.1 Linee Guida Agid 16
3.1.1. Rilascio del codice e organizzazione del repository 16
3.1.2. Rilascio dei documenti che descrivono una pratica amministrativa 16
3.1.3. Processo di accreditamento su Developers Italia (on-boarding) 17
4. APPROFONDIMENTO SULLA GESTIONE DEL REPOSITORY 18
4.1 Identificazione della soluzione nel Repository 18
4.2 Modalità di alimentazione e fruizione dei prodotti del repository 20
4.3 Parametrizzazione profili sul Repository 22
5. APPROFONDIMENTO SULLA GESTIONE DI SOLUZIONI NEL REPO 26
5.1 Gestione dei branch per diverse "versioni" 26
5.2 Rilascio di una nuova versione 26
5.3 Gestione degli issue 27
5.4 Apporto esterno al repository 27
5.5 Controllo della qualità dei prodotti riusabili 27
5.6 Qualità di base (Unit Test) 28
5.7 Qualità dei layer (Integration test) 28
5.8 Test front-end 28
5.9 Performance 28
5.10 Qualità del processo e affidabilità della release 29
6. LICENZE PRODOTTI NEL REPOSITORY 29

1. PREMESSA

La finalità della Linea Guida "La gestione dei Repository di una Open Comunità della PA" , di cui questo documento costituisce un appendice, non è quella solo di definire e disegnare un repository in grado di contenere e gestire le sorgenti dei prodotti software di proprietà o di interesse e i documenti tecnici e di progetto relativi ai riusi messi a disposizione da un Cedente, ma anche quella di rappresentare il contenitore delle "Buone Pratiche", cioè di tutto il materiale di interesse per un utilizzo concreto e sinergico degli stessi.
Per questo il repository oltre ai contenuti che ne caratterizzano il contenitore specialistico delle soluzioni tecnologiche, dovrà prevedere anche un contesto documentale statico e dinamico per contenere i seguenti contenuti tematici per ogni soluzione:

- **Documentazione descrittiva della "buona pratica"** con indicazione dei *software* a supporto, ma anche dei Soggetti che la utilizzano, dei tipi di utilizzo attuati, delle evoluzioni funzionali nel tempo, delle informazioni di promozione e di conoscenza della buona pratica;
- **Documentazione tecnica – Amministrativa per il riuso della soluzione** e per l'iscrizione ai servizi messi a disposizione, cataloghi dei servizi e referenze;
- **Progetti di attuazione della buona pratica** come elemento principale o compartecipe, con tutta la documentazione prodotta e certificata;
- **Ambiente news della buona pratica, con i piani di evoluzione e di gestione esistenti,** i soggetti promotori e partecipanti, le schede di iscrizione a eventuali progetti e/o finanziamenti di programma;
- **Analisi dei *Benchmark* e delle certificazioni o validazioni** da parte di soggetti terzi titolati;
- **Formazione sulla buona pratica,** documenti e calendari, schede di iscrizione, contenuti dei web seminari e altra documentazione.

Stante il contesto di utilizzo del Repository è chiaro che la problematica di creazione dello stesso prima e della sua gestione dopo, sia quella di predisporre un ambiente integrato che risolva la situazione di dispersione del *software* e dei documenti in archivi, uffici e competenze diverse, senza che vi sia un ambiente centralizzato e disponibile ad acquisire e far accedere alla conoscenza ed agli strumenti definiti con le soluzioni adottate dall'Amministrazione per "funzionare meglio".

Questa, peraltro, è una questione annosa ogni volta che un Ente si accinge a effettuare un censimento delle soluzioni e in uso al momento presso di lui. Tutto ciò ha reso difficile da sempre arrivare ad un quadro complessivo del patrimonio di conoscenze e soluzioni in possesso dell'Ente con la conseguente dispendiosa e difficoltosa gestione del suo patrimonio. La dispersione delle informazioni su diversi repository/uffici rende problematica l'individuazione dei documenti, delle prassi e dei software, soprattutto relativamente alle versioni, ponendo per queste ultime esigenze di regole chiare e certe di gestione.

Già limitatamente al solo *software*, da quanto detto, risulta difficoltoso poter estrarre una certa versione dal codice immagazzinato in uno di tali repository e distribuire a terzi il sorgente con la relativa documentazione. Tale scenario è quello tipico del riuso del *software* dove è richiesto di distribuire ad altri enti una versione dello stesso che sia congruente non solo con i diversi componenti che compongono tale versione ma anche con la relativa documentazione. Pertanto, la necessità di definire un modello chiaro di gestione di un repository centralizzato che contenga, come già detto, non solo i codici sorgenti ma anche tutti i documenti della "buona pratica", così come descritta, diventa la chiave del processo di organizzazione di uno strumento che sia funzionale agli obiettivi dell'Ente, nonché al contesto avviato con le proprie iniziative di ingegnerizzazione, innovazione, revisione dei processi interni e degli strumenti digitali di supporto, nell'ottica delle aspettative di interoperabilità, di *workflow* e di tracciamento. Proprio per questa valenza istituzionale

e formale, nelle specifiche del *software* il modello che verrà predisposto dovrà definire sia le modalità operative di alimentazione del repository sia gli *items* (denominati *deliverables*) che dovranno essere inseriti nel repository.

Proprio per queste formalizzazioni volte a standardizzare i processi di innovazione interni all'Ente, motivo principale e scatenante il riuso attivo e/o passivo, le modalità operative di accesso al repository dovranno essere previste con la definizione di ruoli e privilegi in funzione dei diversi profili che potranno e dovranno utilizzare il repository sia le operazioni che andranno eseguite sul repository.

E' necessario, altresì, definire un flusso operativo che stabilisca quali *deliverables* vadano inseriti nel repository e le responsabilità di tali operazioni.

L'obiettivo è quello di:

- Identificare e memorizzare gli *item (o artifacts*) che vanno gestiti nel repository
- Controllare e monitorare i cambiamenti di ciascun *artifact*
- Definire ed organizzare il versionamento degli *artifacts*
- Definire ed organizzare le *baseline*
- Tracciare le richieste di cambiamento
- Definire la struttura del repository
- Assicurare la riproducibilità del software sorgente

Questo deve essere pensato, avviato e portato avanti attraverso un Progetto di istituzione del Repository che tenga conto di obiettivi precisi come:

- Essere il concentratore di tutti i prodotti realizzati o acquisiti come *software* pubblico dall'Ente;
- Acquisire all'interno i prodotti informatici e di buona pratica presenti sul territorio regionale e messi a disposizione dagli Enti, sia per i prodotti realizzati, in conformità alla pratica internazionale legata ad *open source* ed *open content* sotto licenze di tipo "*copyleft* " come EUPL e CCBY, o similari;
- Essere il "Contenitore degli output delle soluzioni adottate dall'Ente". In questa eccezione il Repository Locale dovrà essere predisposto per ospitare Progetti di realizzazione delle soluzioni con la possibilità di gestire in qualità di archivio le varie fasi di attuazione in cui si possa:
  - Gestire i rilasci dei prodotti che scompongono il singolo progetto attivato dall'Ente, soprattutto quando esiste compresenza in uso o realizzazione di altri Enti;
  - Gestire il rilascio finale del prodotto alla gestione ordinaria delle attività (regime) dell'Ente, con la possibilità di iscriverlo al catalogo del riuso;
  - Consentire l'accesso ad utenti di tipo pubblico ed a Soggetti economici per l'acquisizione delle informazioni di interesse e prevedere ambienti di interazione in convenzione o accordo per l'acquisizione e l'utilizzo delle soluzioni.

2. MODELLO TECNOLOGICO DEL REPOSITORY LOCALE

Developers Italia ha scelto come tecnologia di riferimento GIT per la gestione degli ambienti tecnologici. Questa tecnologia è disponibile in rete per la realizzazione di Repository con caratteristiche di funzionamento e con architetture di collegamento come quella di interesse OCPA.

GIT è un sistema di controllo di versione moderno che offre le funzionalità familiari di CVS o *Subversion*, ma fornisce, al contempo, rispetto agli altri prodotti, una nuova visione delle modalità di gestione del software. GIT estende la nozione stessa di sistemi di controllo versione (VCS) per la sua capacità di offrire quasi tutte le sue caratteristiche per l'utilizzo offline e senza un server centrale.
Oggi gli sviluppatori di tutto il mondo stanno migrando in massa verso questa piattaforma. Gli utenti apprezzano le sue prestazioni, la flessibilità di utilizzo, le funzionalità offline, e le funzioni di collaborazione.
GitHub è la versione cloud di GIT nel senso che offre le stesse funzionalità di GIT ma con la possibilità di creare repository in *cloud*.
GitHub è in grado di condividere un repository e un *changeset* direttamente con un altro utente semplicemente accedendo via web ed effettuando un *check-in/ check-out* degli *items*.
Questi strumenti consentono nativamente una serie di attività, assistiti da specifiche funzioni, e offrono la possibilità di caratterizzare un'architettura di governo dei manufatti secondo logiche strutturate per tipologia, per cronologia, per organizzazione del lavoro, per tipologia di utenza.
Di seguito gli aspetti tecnologici caratteristici che ne consentono l'applicazione rispetto alle problematiche OCPA.

2.1 Software configuration Management

La gestione del codice sorgente e dei documenti, è una pratica più comunemente indicata come *SCM Software Configuration Management*. Con SCM si intende la gestione delle modifiche apportate ai documenti, a prodotti *software* e più in generale ad un insieme di informazioni. Il *Configuration Management*, in altri termini, ha lo scopo di permettere la gestione ed il controllo degli oggetti (siano questi prodotti *software* o documenti).
Nell'ambito del *Configuration Management* vengono gestiti gli *input/output* direttamente o indirettamente legati alla costruzione di un prodotto *software*. Si tratta, quindi, non solo di archiviare in modo controllato le varie versioni del codice sorgente sviluppato, ma anche di gestire le altre entità create nel corso delle diverse fasi dello sviluppo. Ogni elemento oggetto delle attività di gestione della configurazione viene normalmente chiamato *configuration item*. Perché tutti i *configuration items* possano essere adeguatamente gestiti è necessario che siano definite le possibili tipologie e le operazioni che su di essi possono essere compiute, compresi i ruoli dei vari attori coinvolti. Tutto ciò viene tipicamente definito all'inizio del processo di sviluppo del *software* (ovvero di un progetto) ed è dipendente in una certa misura dal progetto. La gestione della configurazione può essere applicata, in generale, a diverse categorie di oggetti (comunemente noti come "artefatti"). Qui di seguito viene riportata una breve lista di tali categorie di oggetti:

- Documenti di progetto:
  - Prodotti di fase;
  - Documenti tecnici;
- Codice sorgente;
- Manuale utente;
- *Test Case*;
- Manuali di installazione e gestione del prodotto;
- Prodotti finali dello sviluppo (compilati, eseguibili, ecc.);
- Schemi dei database;
- Documenti relativi alla manutenzione del prodotto;
- Standard e procedure adottate;
- Documenti per la formazione tecnica e funzionale degli utilizzatori;
- Documenti descrittivi della buona pratica gestita attraverso lo strumento, presentazioni e seminari;

- Progetti in corso derivati dall'uso della buona pratica/software, istanze di partecipazione se bando aperto, istanze di risultato raggiunto se bando chiuso;
- Documenti descrittivi delle Amministrazioni cedenti e riusatrici, documenti per il riuso della soluzione.

Compito di queste tecnologie è allora quello di garantire la "gestione del Repository" per organizzare e governare il materiale e i cambiamenti apportati, a ciascuno di tali oggetti. Questo esercizio è pensato per dare supporto operativo e documentativo nelle fasi del ciclo di vita di un progetto, garantendo la gestione di tutti gli aspetti descritti.

Rimanendo confinati negli aspetti di gestione della documentazione del *software* e del sorgente (per gli altri temi di interesse del repository i criteri di modellazione sono gli stessi) GitHub può essere inteso in modo efficace come repository finale dei prodotti che si intendono mettere a riuso.

2.2. Implementazione ambiente del KIT

Partendo dalle caratteristiche del CSM, utilizzato anche come semplice archivio della documentazione oggetto di riuso, è necessario definire un modello di implementazione del repository basato sulla costituzione di modelli di ambienti digitali in cui riversare i prodotti del ciclo di vita di una soluzione e delle sue esperienze.

In generale si possono individuare tre categorie di *pattern*:

- *Core pattern;*
- *Workspace pattern;*
- *Code line pattern;*

Ciascuno di questi tre *patterns* incide su un particolare aspetto dello sviluppo, della gestione del repository, dell'allineamento del codice sorgente e della documentazione tutta secondo il modello OCPA.

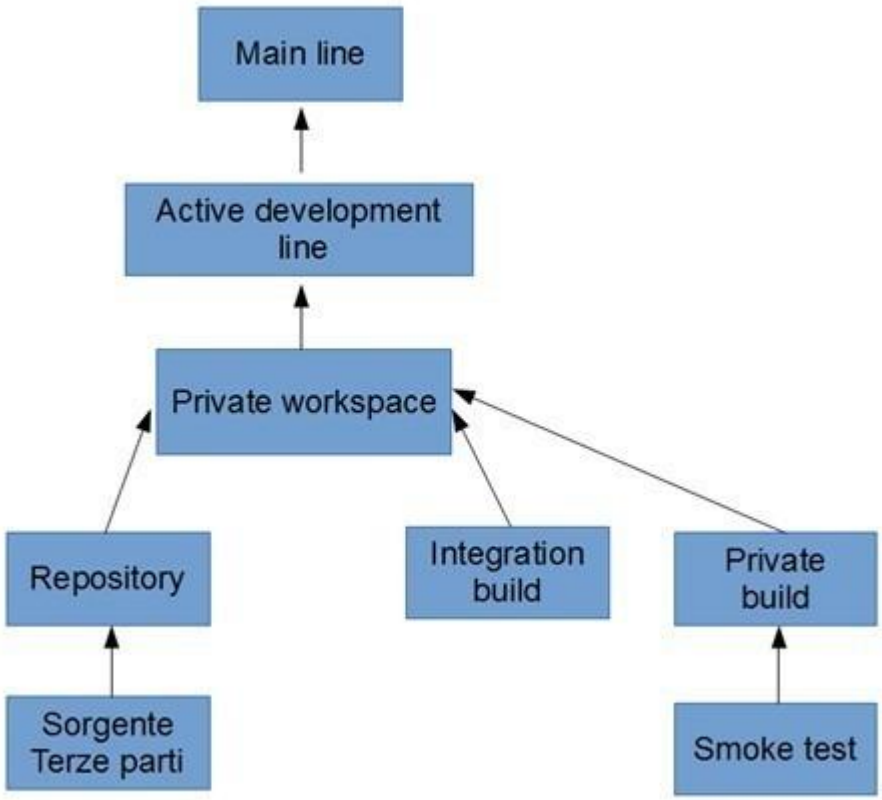
In questo contesto l'obiettivo è quello di delineare una architettura che consenta ad un Repository locale:

- Di gestire il ciclo di vita delle soluzioni *software* a riuso realizzate nel contesto di un progetto di realizzazione o di riuso di una pratica amministrativa ed organizzativa (esperienza);
- Di gestire la documentazione del KIT OCPA che supporta la possibilità di creare, gestire e consultare la documentazione dell'esperienza con accesso ai documenti ed al materiale reso disponibile;
- di gestire l'accesso ai progetti del Repository.

2.2.1. Organizzazione per la gestione dell'Open Source

Ciascuno dei tre *patterns* sopra citati incide su un particolare aspetto dell'allineamento del codice sorgente e della documentazione a corredo.

Qui di seguito viene riportato uno schema di massima dei *pattern* e della struttura del repository e di come dovrebbe essere utilizzato lo stesso al fine di massimizzare l'efficienza. Il modello è indipendente dal tipo di strumento utilizzato.



Nome Pattern	Descrizione
Mainline	Minimizza l'attività di *merge,* mantenendo il numero di linee di codici attive sotto controllo
Active Development Line	Mantiene stabile una linea di sviluppo che si evolve rapidamente.
Private Workspace	Evita che i problemi di integrazione influiscano sulla linea di sviluppo di ciascun sviluppatore.
Repository	Inizializza un nuovo *workspace* popolandola con i dati provenienti dal repository.

Integration Build	Assicura che la *baseline* sia sempre affidabile e compilabile attraverso una attività periodica di integrazione e successive *build.*
Third Party Codeline	Gestisce il codice di terze parti
Smoke Test	Assicura che dopo le modifiche il sistema continui a funzionare eseguendo uno *smoke Test*

Più nello specifico, il sistema Github permette la creazione di *branch* e di organizzare tali *branch* in modo gerarchico. Questo risulta particolarmente utile per implementare i pattern prima descritti. Una tipica struttura a *branch* di un repository GIT è qui di seguito rappresentata:

- ***refs/heads/master*** (main line for future releases);
- ***refs/heads/releases/stable-x.y.z*** (set of branches for integrating changes and stabilizing code for release);
- ***refs/heads/user-xyz/mybranch*** (optional but very useful: namespace for individual users).

A riguardo del modello adottato, si riporta la seguente raccomandazione:
La gestione dell'*Open Source* in un contesto di rispetto delle linee guida AgID, cui si allinea lo stesso KIT per la parte delle tecnologie software a riuso utilizzate, richiede, specificatamente, che siano almeno presenti (vedi anche "Linee guida su acquisizione e riuso di software per le pubbliche amministrazioni"):la certificazione e comunicazione di una delle *Public licence* previste dalle Linee guida AgID del 9 maggio 2019 il codice sorgente degli strumenti digitali a riuso.
Nella sezione degli allegati alla Sezione del KIT FASE B documentazione Tecnologica il set di allegati ai documenti master di Argomento (B2 e RB2):

- *Scheda AgID*
- *Documento architetturale*
- *Documento di specifica dei requisiti*
- *Manuale di installazione*
- *Manuale di configurazione*
- *Codice sorgente compilato*

2.2.2. *Organizzazione e items del KIT OCPA*

Organizzazione di un Repository e degli items che caratterizzano il KIT costituiscono il patrimonio fondante il riuso delle pratiche amministrative ed organizzative. Gli *items*, di norma sono definiti nella fase di progettazione, costituiscono un set "minimo" di oggetti, necessario a descrivere e caratterizzare una soluzione IT o una pratica all'interno del repository. Il set varierà significativamente a seconda che l'oggetto del repository sia una soluzione e/o una esperienza. Questo comporta due esigenze: il modello di architettura del riuso e la definizione e trattazione degli items.

Un repository GITHub lavora per progetti, per cui un KIT deve essere considerato come Progetto nel repository. Per questo è possibile sempre creare un ambiente denominato *Project-template*, che funge da modello avente una struttura, in genere, standardizzata. Nel caso del repository OCPA e del KIT dell'esperienza viene data la seguente articolazione che tiene conto del ciclo di vita:

- **KIT <NOME RIUSO > OCPA**
    - Guide al riuso (Linee guida e manuali di approccio alle FASI)
    - Stato del KIT del riuso:
        - Documenti descrittivi
        - Stati avanzamento
        - Documenti di collaudo
    - FASE A - elementi di analisi e decisione
        - Gestione della scelta e comparazione e stima dei costi
        - Verifica rispetto all'Organizzazione
        - Tecnologie da prevedere (previsto Link a materiale ambiente *Open Source*)
        - Fattori Amministrativi di interesse
        - Contesto informativo e comunicazione
        - Supporti e servizi di ausilio alla Fase
        - Ambiente format di riuso vuoti
    - FASE B - elementi di Acquisizione e Realizzazione
        - Gestione del piano di realizzazione
        - Organizzazione e configurazione processi dei servizi
        - Acquisizione e configurazione delle tecnologie (previsto link a materiale ambiente *Open Source*)
        - Procedure Amministrative da prevedere
        - Supporti ai piani di formazione e comunicazione
        - Supporti e servizi di ausilio alla Fase
        - Ambiente format di riuso vuoti
    - FASE C
        - Sviluppo del piano di gestione e indicatori per i costi di esercizio
        - Fattori organizzativi a regime e check list conformità
        - Scelta e indicazioni sull'impianto tecnologico a regime (previsto link a materiale ambiente *Open Source*)
        - Procedure amministrative da prevedere
        - Descrizione dei supporti e dei servizi OCPA per il riusante
        - Supporti e servizi di ausilio alla Fase
        - Ambiente format di riuso vuoti

Ambiente di servizio:

- Format amministrativi di contatto
- Scheda di valutazione prodotto per Riusante
- Materiale informativo per diffusione a Riusanti
- Catalogo servizi cedente disponibili
- Catalogo servizi *cloud* disponibili
- Catalogo soggetti esterni capacitati ai servizi
- FAQ

Gli item presenti nell'ambiente dovranno essere inseriti, catalogati e mappati secondo la logica degli "*items* di progetto" come d'uso per il repository di GIT. Essi hanno il compito, analogamente al materiale dell'*Open Source*, di supportare e valorizzare sicuramente le soluzioni, ma soprattutto le esperienze sviluppate intorno al riuso da parte dei riusanti (ad iniziare dal cedente originario, riusante di sé stesso). Gli *items* sono strutturati secondo documenti *Master* (vedi KIT riuso OCPA) e pertanto vanno organizzati nel repository secondo un contenitore predisposto appositamente come sopra esposto.
La struttura è costituita da riferimenti ad *items* classificati secondo tre tassonomie:

1. Una rappresentata nell'elenco sopra descritto rispetto alle FASI del ciclo di vita del KIT, per la cui dizione si rimanda all'elenco stesso

2. Una legata alla tipologia di Argomenti che consente un accesso trasversale alle Fasi, basato sui 5 argomenti del kit. Di seguito il set di *items* tematici in relazione al ciclo di vita della pratica Amministrativa e organizzativa e/o della soluzione:
   - *Items* descrittori del modello di gestione del processo del riuso
   - *Items* descrittori del modello di organizzazione degli iter di riuso
   - *Items* descrittori delle tecnologie che concorrono a supportare la pratica
   - *Items* descrittori dei modelli di comunicazione e formazione previsti nel ciclo di vita
   - *Items* descrittori dei processi amministrativi necessari per Fase

3. Una legata agli output delle attività previste per il riuso della pratica. Un elenco tipo può essere:
   - *Items* descrittori dello scopo del riuso e dell'area dell'intervento
   - *Items* descrittori delle attività operative che concorrono a caratterizzare il riuso
   - *items* che descrivono gli eventi del riuso per le verifiche e i controlli sui processi
   - *Items* che descrivono gli adempimenti previsti necessari nelle fasi
   - *Items* che descrivono gli obiettivi definiti per la pratica attraverso il riuso

Nel caso in cui nel KIT sia presente una soluzione *Open Source* della Pubblica Amministrazione, essa viene trattata indipendentemente dal KIT, nel senso che manterrà le regole di costituzione e gestione previste dalle Linee Guida Agid. Il legame tra KIT e soluzione nel Repository OCPA avviene attraverso link previsti tra gli oggetti presenti nella soluzione e nella pratica, che consentono di caratterizzare la soluzione IT disponibile nel riuso.
Nel caso di soluzioni *software*, la definizione dei singoli *items è* abbastanza consolidata e possiamo indicare come presenza di livello minimo (vedi linee guida Agid 9 maggio 2019):

- Codice sorgente
- Documentazione tecnica
- Documento di architettura della soluzione
- Documenti tecnici

- Specifica dei requisiti
- *Use case*
- Piano dei test
- Riferimenti a *standard* e metodologie adottate
- Documenti su struttura *Database*
- Documenti su specifiche di colloquio con il sistema e tra il sistema e altri componenti esterni
- Manuali
- Manuali di installazione
- Manuale utente
- Manuale di gestione del prodotto
- FAQ

2.2.3. Progetti GIT nel Repository

Si è detto nel paragrafo precedente che GIThub lavora per progetti. Al fine di dare una nomenclatura *standard* e funzionale alla gestione del Repository, ciascun progetto si propone con una radice iniziale di un *file* denominato *README.md*. Tale file viene visualizzato da GitHub al momento in cui si accede al progetto. Qui di seguito viene riportato un esempio:

Project name

Descrizione

Questa è una breve descrizione del progetto. La descrizione deve essere chiara con lo scopo di illustrare le finalità del progetto e l'ambito di intervento.

Struttura del repository

Il repository ha la seguente struttura

Folder	Descrizione
bin	Questo folder contiene i file compilati o binari.
documenti	Questo folder contiene la parte documentale del progetto. Il folder è suddiviso in sub folders per contenere documenti tra loro omogenei.
screenshots	Questo folder contiene alcuni screenshots delle schermate principali del prodotto in modo da dare a chi legge un'idea immediata della UI del prodotto
src	Questo folder contiene la parte del codice sorgente del prodotto. In questo folder vanno inseriti non solo il codice sorgente ma anche tutti gli scripts necessari alla creazione del database
tools	Questo folder contiene tutti gli eseguibili dei prodotti necessari al corretto funzionamento dell'applicativo e alla sua compilazione (es. maven, ant, ecc.)

Ambiente di sviluppo

In questa sezione vanno inserite tutte le informazioni relative all'ambiente di sviluppo utilizzato per il prodotto. In particolare va indicato con chiarezza sia il nome sia la versione dell'IDE utilizzata.

Licenza

In questa sezione va inserita la parte relativa alla licenza con cui si intende distribuire il codice. Nel caso in cui si preferisca utilizzare una file apposito (LICENSE.md) allora è necessario inserire il link a tale file.

Riferimenti

In questa sezione vanno inseriti i riferimenti alla persona/persone ovvero alla struttura che ha in carico la gestione del prodotto e può fornire informazioni utili.

Tale file deve essere scritto utilizzando *Markup language*. Per maggiori informazioni si può consultare il seguente link: https://guides.github.com/features/mastering-markdown/.
In generale, ciascun progetto deve avere un proprio repository così da poter gestire gli accessi allo stesso in maniera indipendente.

2.2.4. Accesso ai progetti del Repository

In generale, l'accesso ai progetti presenti su GitHub avviene in due modalità:

- Accesso libero
- Accesso tramite autenticazione

Le due modalità di accesso sono mutuamente esclusive per uno stesso progetto e ciascun progetto/repository può essere configurato in maniera indipendente dall'altro.
Nella prima modalità di accesso al progetto/repository, ovvero accesso libero, chiunque può connettersi a GitHub e scaricare i *files* del progetto, tramite un'operazione che viene chiamata clone del repository. E' evidente che in questa modalità non vi è controllo sugli accessi e il progetto può essere scaricato ed utilizzato da chiunque. In questa modalità di accesso il repository (ovvero il progetto) è pubblico.
Nella seconda modalità, ovvero accesso tramite autenticazione, il progetto/repository è privato. Gli utenti, intesi come aziende, privati o pubbliche amministrazioni, che intendono accedere al repository devono prima registrarsi su GitHub e successivamente essere abilitati all'accesso del progetto di interesse. Tale abilitazione viene effettuata dal gestore del repository o dai soggetti abilitati a tale funzione.

Gestione ruoli

Github prevede tre ruoli fondamentali:

- *Owner*
- *Billing manager*
- *Member*

L'***Owner*** è il proprietario del repository e dovrebbe sempre corrispondere ad un utente titolato, dell'Ente proprietario del Repo, ad assicurare il pieno controllo sul repository e sulle attività che possono essere compiute su di esso.

Il ***Billing manager*** è colui che gestisce, ovvero delega, la gestione economica dell'account di GitHub.
Il ruolo ***Member*** è il ruolo di *default*.

Gestione permessi

Dal punto di vista dei permessi, GitHub supporta diversi tipi di permessi per gli utenti che appartengono ad una organizzazione:

- *Read*
- *Write*
- *Admin*
- *Owner*

In allegato alla presente Linea Guida si riporta una mappatura esempio di configurazione dell'azione sul repository da parte dei profili istruiti in GitHub.

Gestione collaboratori esterni

GitHub, come detto, prevede diversi profili e modalità di accesso al progetto/repository con differenti livelli di autorizzazione. In generale, il profilo di accesso consigliato da parte di un soggetto (privato o P.A) non appartenente all'organizzazione dell'Ente presente nel Repo è quello di "*external collaborator*". Con questo profilo, è possibile accedere in sola lettura al progetto ed eventualmente fare il clone del repository nel caso in cui l'utente sia interessato.

Utilizzando il profilo di *external collaborator* è possibile organizzare e controllare gli accessi per i singoli progetti senza che questi utenti abbiano gli stessi ruoli e gli stessi privilegi degli utenti appartenenti all'organizzazione dell'Ente.

Una volta che l'utente ha raggiunto il progetto/repository ha a disposizione diverse funzionalità ed in particolare può, o semplicemente navigare all'interno della struttura del progetto, oppure fare il *download* dell'intero progetto tramite la funzione di clonazione del repository. Una volta clonato il repository può essere gestito e modificato in maniera indipendente dal repository originale. Questo, per esempio, consente agli utenti di poter provare quanto rilasciato su GitHub.

Per consentire all'utente di avere un quadro complessivo del progetto e al fine di permettere a chi accede di orientarsi è necessario che ciascun progetto presente sul repository abbia un testo di presentazione (denominato in GitHub *readme.md*) che contenga la descrizione del progetto, le sue finalità e quali processi supporta.

Distribuzione immagine del prodotto tramite *docker*

Al fine di permettere all'utente di provare l'applicativo in maniera semplice e rapida è possibile aggiungere al repository GitHub un'immagine compatibile con *Docker.*

Docker è l'applicazione leader per la gestione di *"container"* applicativi che eseguono processi in ambienti isolati.

Docker permette di pacchettizzare un applicativo utilizzando un proprio *filesystem* che contiene tutto quello che serve all'applicativo stesso per funzionare, come per esempio librerie, tools ecc. Tutto questo può essere installato, come una singola unità *software,* direttamente su un server/pc.

Tale sistema permette di semplificare notevolmente il processo di installazione di un nuovo prodotto al fine di valutarne le caratteristiche. Un ulteriore vantaggio è dato dal fatto che l'utente, che intende provare l'applicativo, non ha bisogno di conoscere i diversi passi di installazione e di configurazione del prodotto stesso.

Docker, inoltre, è una piattaforma *open-source* gratuita e perfettamente integrabile con Github.

3. MODELLO DEL CATALOGO NAZIONALE "DEVELOPERS ITALIA"

Il repository locale, come riportato in precedenza, è il deposito dove sono disponibili tutti i materiali necessari a replicare una soluzione IT, o una pratica amministrativa, nell'ambiente operativo di una Amministrazione che ha deciso di acquisirla in riuso. Tutte le soluzioni, o le pratiche amministrative, sono anche presenti su *Developers Italia*, la piattaforma nazionale individuata da AgID prevista all'art 69 del Codice dell'Amministrazione digitale, che rappresenta il catalogo nazionale delle soluzioni a riuso.

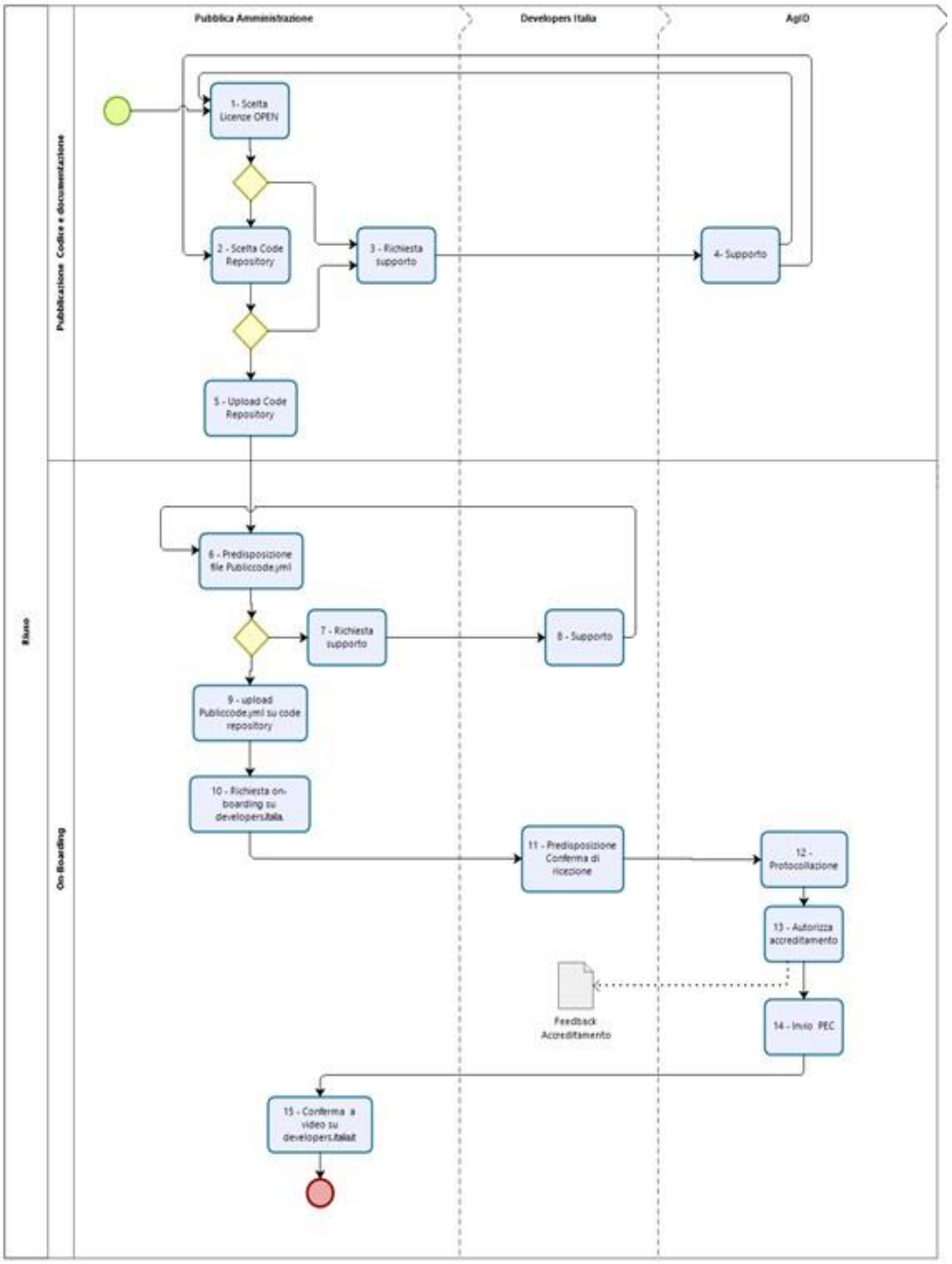
Developers Italia non è un repository nazionale, ma semplicemente un catalogo delle soluzioni presenti su tutti i repository delle Amministrazioni che si sono accreditate.

All'interno della piattaforma viene resa disponibile una sezione (il catalogo) dedicata al *software* messo a riuso dalle Amministrazioni. Prossimamente lo stesso processo verrà reso disponibile per gestire le pratiche amministrative. In particolare:

- è disponibile un «motore di ricerca» delle soluzioni in riuso per mezzo del quale l'Amministrazione potrà cercare *software* (libero), o pratiche, rilasciati da altre Amministrazioni, utilizzando strumenti di consultazione messi a disposizione dalla piattaforma (es: filtri per tipologia di oggetto, tipologia di amministrazione, etc);
- è disponibile una modalità per «registrare» in *Developers Italia* le soluzioni delle Amministrazioni rilasciate ai fini del riuso, perché diventi facilmente individuabile da parte di altre Amministrazioni.

La funzionalità del catalogo nazionale è comunque subordinata alla pubblicazione della soluzione in un repertorio pubblico con licenza aperta, è possibile quindi suddividere il processo di riuso in due parti (vedi schema successivo):

- il primo processo riguarda la pubblicazione della soluzione e la relativa documentazione su un Repository pubblico;
- il secondo è il processo di accreditamento e pubblicazione (*on-boarding*) sulla piattaforma *developers.italia.it*.



Per quanto riguarda il riuso del sw, l'intero processo può essere sintetizzato come segue:

- l'Amministrazione deve ottenerne sempre la piena titolarità;
- il software deve essere pubblicato in un repositorio pubblico (ad es. GitHub, GitLab, BitBucket ecc., anche *on-premises* purché pubblicamente accessibile);
- al *software* deve essere applicata una delle licenze approvate da *Open Source Initiative* (le Linee Guida ne suggeriscono alcune in particolare, per consentire la massima riusabilità);
- nel repositorio deve essere incluso il file *publiccode.yml* che ne descrive le caratteristiche e consente di popolare il catalogo di *Developers Italia* automaticamente tramite una modalità di indicizzazione (*crawler*), al fine della generazione della relativa scheda nel catalogo stesso.

Relativamente al riuso di una pratica amministrativa, il percorso è del tutto analogo a quello delle soluzioni IT, *Developers* avrà una sezione dedicata alle pratiche amministrative che funzionerà esattamente con la stessa logica riservata alla gestione del riuso di sistemi IT.

3.1 Linee Guida Agid

Si riportano di seguito in forma sintetica i paragrafi delle Linee Guida pubblicate da AgID nei quali si descrivono due aspetti rilevanti che riguardano i repository: l'organizzazione e la registrazione sul Catalogo Nazionale.

3.1.1. Rilascio del codice e organizzazione del repository

Il codice sorgente deve essere rilasciato in versione integrale e senza omissioni in modo che un soggetto terzo possa, seguendo la documentazione, compilarlo (ove applicabile) e metterlo in funzione senza doverlo modificare. I nomi delle variabili, delle funzioni, delle classi e degli altri simboli devono essere mantenuti in chiaro e devono essere comprensibili; parimenti, il codice non deve essere sottoposto ad alcun trattamento di compressione (c.d. *minification*) che ne ostacoli la leggibilità. Qualsiasi tentativo di offuscamento è considerato violazione dell'obbligo di rilascio.

Deve essere posta massima attenzione sulla leggibilità del codice, che deve essere correttamente indentato e commentato in ogni suo passaggio. È richiesta l'adozione di un *coding style* coerente e pulito. Alcuni esempi di convenzioni:

- https://github.com/google/styleguide
- https://www.gnu.org/prep/standards/
- https://www.kernel.org/doc/Documentation/process/coding-style.rst
- http://www.php-fig.org/psr/psr-2/
- http://pear.php.net/manual/en/standards.php

È raccomandata l'adozione di un'architettura modulare, basata sulla suddivisione della logica in librerie specializzate e riutilizzabili singolarmente, con API interne definite e documentate nei commenti del codice. In caso di integrazione di librerie esterne, si raccomanda l'uso dei *package manager*, per facilitare la manutenzione e l'aggiornamento.

Il rilascio in *open source* non deve essere considerato come mero adempimento da svolgersi al termine della lavorazione, ma deve essere previsto sin dalla fase di sviluppo, ad esempio strutturando il *software* in modo che tutte le specificità dell'Amministrazione committente (nomi, indirizzi, server) siano modificabili attraverso file di configurazione e che il *software* sia pronto al riuso da parte di altro soggetto.

Il repository deve essere organizzato con una struttura di *directory* chiara e comprensibile, ad esempio separando in *directory* distinte documentazione, librerie, eseguibili, *script* di servizio, *test suite*, etc.

3.1.2. Rilascio dei documenti che descrivono una pratica amministrativa

Analogamente a quanto previsto per il rilascio di codice sw, il motore di ricerca della piattaforma *Developers Italia* andrà a ricercare ogni notte la presenza di contenuti sui repository delle Amministrazioni che si sono accreditate su Developers. In caso positivo, quest'ultimo provvederà, in automatico, ad alimentare, o aggiornare, la piattaforma con le informazioni utili al riuso della pratica.

Il repository:

- conterrà documenti in formato aperto, secondo quanto previsto dalle linee guida AgID
- quando possibile, utilizzerà formati che permettano facilmente l'individuazione delle modifiche intercorse fra una versione e l'altra, come ad esempio il formato *ReStructured Text* (.rst) che permette una visualizzazione su docs.italia.it
- utilizzerà nomi significativi per i documenti, in modo da rendere chiaro all'utilizzatore il loro scopo e la loro scansione temporale
- sarà corredato da un file *README.md* che descriverà:

  - il progetto e le scelte fondamentali in linea generale
  - l'organizzazione dei file
  - la scansione temporale con la quale sono stati presi gli atti
- sarà opportunamente suddiviso in cartelle che separino le diverse funzionalità di documenti, fra quelli amministrativi, le deliberazioni, i pareri necessari, le pratiche organizzative.

3.1.3. Processo di accreditamento su Developers Italia (on-boarding)

Non appena il repository pubblico è stato aperto, è necessario effettuare la registrazione dell'Ente interessato al riuso su *Developers Italia*, per garantire che venga indicizzato e presentato nel motore di ricerca presente sul sito.

La registrazione avviene seguendo due passaggi:

1. pubblicazione di un file *publiccode.yml* nella directory *root* del repository che deve essere indicizzato da *Developers italia. «publiccode.yml»* è uno standard che identifica il progetto come «*software* utile per la Pubblica Amministrazione», e contemporaneamente offre una serie di informazioni utili alla valutazione del *software* stesso per il riuso. Tale file verrà rilevato automaticamente dall'indicizzatore *(crawler*) di *Developers Italia* al fine della generazione della relativa scheda nel catalogo. La documentazione sul formato può essere trovata qui: https://github.com/italia/publiccode.yml, è inoltre disponibile un editor https://publiccode-editor.developers.italia.it/
2. aggiunta dello strumento di *code-hosting* al motore di ricerca. Al fine di accertarsi che *Developers Italia* identifichi correttamente il repository come di proprietà della Pubblica Amministrazione, è necessario registrare lo strumento di *code-hosting* (o meglio, l'«organizzazione» all'interno dello stesso) la prima volta che viene usato, associandolo alla Pubblica Amministrazione.

Il repository deve inoltre contenere un file denominato *README.md* per ogni soluzione/pratica messa a riuso con le seguenti informazioni:

- il titolo del repository ed un sottotitolo descrittivo;
- descrizione estesa del repository in un linguaggio comprensibile anche dai non addetti ai lavori (evitare acronimi e gergo tecnico), ma sufficientemente completa da facilitarne la ricerca all'interno del catalogo, In particolare:
  – contesto di utilizzo e casi d'uso;
  – finalità del *software*;
  – *screenshot (*se il *software* dispone di interfaccia grafica, anche web);
  – link ad eventuali pagine istituzionali relative al progetto o al contesto di utilizzo;
  – elenco di norme o procedure che la soluzione permette di rispettare
- link ad eventuale documentazione aggiuntiva non inclusa nel presente repository;
- spiegazione struttura del repository anche a beneficio dei potenziali contributori (struttura delle *directory* e dei *branch*);
- elenco dettagliato dei prerequisiti e delle dipendenze (sistemi operativi, librerie, *framework,* ecc..) con esplicita indicazione di eventuali dipendenze da software commerciali;
- istruzioni per l'installazione:
  – procedura di installazione di requisiti e dipendenze;
  – *build system* (se previsto dal progetto);
  – comandi per la compilazione o il deployment, possibilmente automatizzati da uno *script/Makefile* (se previsto dal progetto);
- eventuali indicazioni sullo status del progetto:
  – stato di alpha/beta/stabile eccetera;

– importanti limitazioni o *known issues*;

- link ad eventuali sistemi di *Continuous Integration (TravisCI, CircleCI*), *code coverage* (copertura del codice) ed altre metriche associati al repository;
- documentazione relativa all'eventuale utilizzo di sistemi per semplificare e accelerare il *deployment* in ambiente di sviluppo, test e produzione (ad esempio immagini *Docker* o altri sistemi di virtualizzazione con predisposizione di immagini preconfigurate);
- nomi dei detentori di *copyright*, ovvero l'Amministrazione committente;
- nomi dei soggetti incaricati del mantenimento del progetto *open source* (è richiesto il nome dell'azienda e facoltativamente si possono aggiungere nomi delle persone incaricate);
- indirizzo e-mail a cui inviare segnalazioni di sicurezza

Infine, il repository dovrà indicare in un file *LICENSE.txt* la licenza scelta per il codice pubblicato, all'interno di quelle ammissibili dalle Linee Guida.

4. APPROFONDIMENTO SULLA GESTIONE DEL REPOSITORY

4.1 Identificazione della soluzione nel Repository

Premesso l'obbligo di pubblicazione delle soluzioni tecnologiche previsto dall'art. 68 del CAD, è necessario predisporre i meccanismi di valutazione tecnico/amministrativa/funzionale necessari per rendere disponibile una soluzione all'interno di un repository di una Amministrazione.
E' la situazione di qualificazione della soluzione nel Repo locale per la quale intervengono una serie di valutazioni e considerazione tra cui:

- Universalità del prodotto
- Qualità del prodotto
- Integrabilità del prodotto
- Altre considerazioni

Il primo punto riguarda l'Universalità del prodotto. Questo termine indica se il prodotto, inteso sia come componente software sia come buona pratica, supporta un processo che può essere generalizzato e quindi presente anche in altre PA oppure un processo specifico di un Ente. Più il processo supportato ha valenza anche al di fuori di un Ente, maggiore è la probabilità che il prodotto possa essere messo a riuso.
La qualità del prodotto sta ad indicare le modalità con cui il prodotto stesso è stato costruito e/o implementato. Ad incrementare la qualità concorrono diversi fattori tra cui:

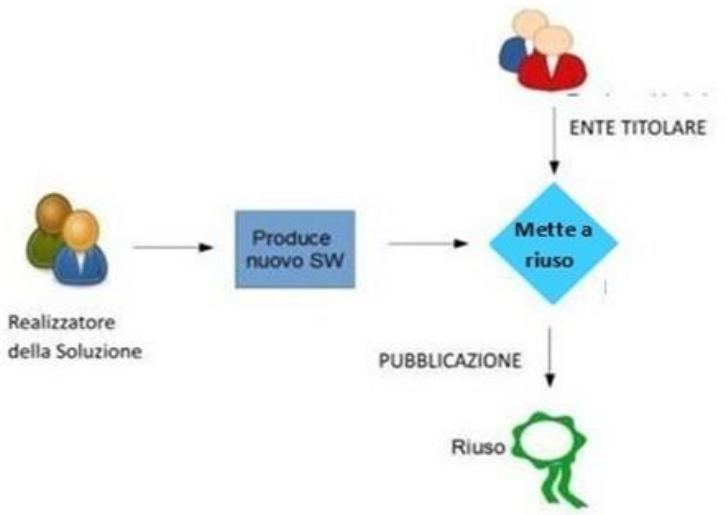
- Adesione del prodotto a standard tecnici/tecnologici
- Adesione del prodotto a normative regionali/nazionali
- Qualità della documentazione a supporto del prodotto
- Stabilità ed affidabilità del prodotto

Un aspetto importante da considerare è l'integrabilità del prodotto. A meno che il prodotto offerto in riuso non sia un componente stand-alone con un ambito di intervento circoscritto e ben definito, è importante valutare quanto il prodotto sia integrabile con altre soluzione e componenti normalmente presenti nel sistema informatico di un Ente:

- Protocollo
- Sistema gestione contabilità
- Sistema gestione documenti

Questo aspetto si interseca anche con la qualità del prodotto e l'adesione a standard e normative.

Oltre agli aspetti prima descritti, nella valutazione dell'opportunità o meno di mettere a riuso un prodotto vi sono altri tipi di considerazioni legate all'origine del prodotto. In particolare, se il prodotto è già stato preso a riuso da un'altra PA. In tal caso, può essere considerata l'opzione di contribuire al prodotto già esistente aggiungendo le nuove funzionalità sviluppate. Infine, è necessario valutare le licenze dei componenti utilizzati nel prodotto. Tale aspetto verrà approfondito successivamente.
Qui di seguito lo schema che descrive brevemente il processo:



Come rappresentato all'interno dello schema, si possono individuare due attori:

- Realizzatore per Amministrazione Titolare
- Amministrazione Titolare

Entrambi collaborano in fase di rilascio al fine di rendere disponibile la soluzione tecnologica ad altre Amministrazioni.
Come precisato all'interno della Linea Guida sul Repository, l'art. 68 del CAD prevede l'obbligo di pubblicazione del *software* realizzato da parte dell'Amministrazione Titolare. La riusabilità si ottiene mediante l'apposizione della licenza aperta che consente alla soluzione tecnologica di diventare "*open source*".
La pubblicazione del *software* prevede inoltre la:

- Produzione della scheda del riuso OCPA (AGID e ACT, Agenzia per la Coesione Territoriale)
- Produzione della documentazione necessaria secondo quanto previsto nel documento tecnico delle modalità di gestione del repository.

4.2 Modalità di alimentazione e fruizione dei prodotti del repository

In questa fase verranno analizzati due aspetti importanti:

- Le modalità di alimentazione del repository
- Le modalità di fruizione dei prodotti del repository

Per quanto riguarda le modalità di alimentazione del repository si possono distinguere due tipologie:

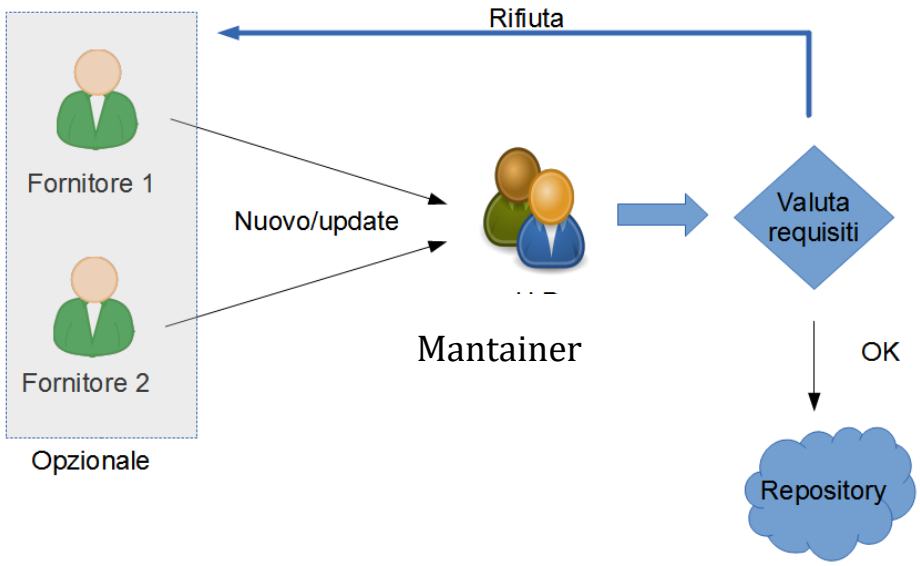
- Inserimento di un nuovo prodotto
- Inserimento di un aggiornamento ad un prodotto esistente

L'inserimento di un nuovo prodotto segue le regole prima descritte. Il repository viene alimentato con un nuovo prodotto attraverso la creazione di uno specifico ambiente di repository che conterrà il prodotto da mettere a riuso secondo le modalità descritte in queste linee guida.

Nel caso di aggiornamento di un prodotto già precedentemente messo a riuso, si procederà ad aggiornare il repository secondo le modalità definite dal processo organizzativo definito nel progetto del Repository predisposto dall'Ente.

In entrambi i casi, verranno effettuati controlli dal punto di vista tecnico/tecnologico per assicurare che quanto rilasciato sia conforme sia al modello di sviluppo del software, sia al modello definito dal repository.

Qui di seguito è schematizzato il processo di inserimento nel repository:



Da quanto sopra riportato, si evidenzia che l'accesso al repository in fase di inserimento e di aggiornamento è sempre mediato dal *Maintainer* del repository stesso, che valuta se il nuovo prodotto/update rispetta i requisiti tecnici per essere inserito nel repository centrale.

Accanto a queste attività di accesso e di reperimento delle soluzioni messe a riuso, vi sono un insieme di attività, molto importanti, di animazione e diffusione delle informazioni sui prodotti presenti nel repository e pronti al riuso.

Tali attività seguono due linee di azione:

- Stimolare l'interesse verso i prodotti messi a riuso dall'Ente al fine di favorire la diffusione delle conoscenze e delle buone pratiche
- Attrarre altre Pubbliche Amministrazioni e/o soggetti privati al fine di evolvere le soluzioni proposte sia dal punto di vista funzionale sia tecnologico. L'obiettivo è quello di concentrare gli sforzi e le risorse su soluzioni già disponibili che possono costituire un punto di partenza comune per ulteriori sviluppi e mettere a fattor comune esperienze maturate in ambiti diversi.

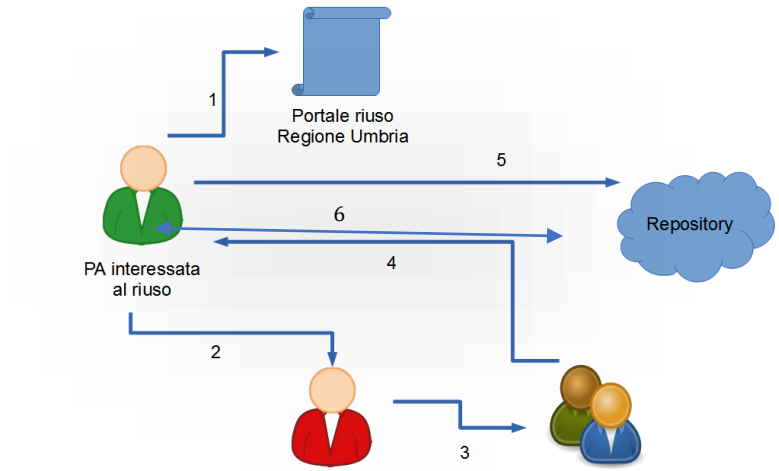
Il secondo aspetto è quello relativo alla modalità di fruizione dei prodotti a riuso da parte delle altre Pubbliche Amministrazioni.

In generale gli attori possono essere suddivisi in attori appartenenti agli Enti pubblici e quelli appartenenti alle aziende private. Per quanto riguarda gli attori "pubblici" questi possono essere divisi in:

- *Cedenti:* soggetti che hanno sviluppato una soluzione e la mettono a disposizione sul repository;
- *Riusanti:* soggetti che accedono al repository e scelgono di adottare una delle soluzioni offerte.

Il ruolo di coordinare le attività di gestione e di "animazione" del repository spetterà all'Ente titolare del Repo (nel caso OCPA alla Comunità ad esempio), mentre il coordinamento tecnico sulla gestione del Repo al Maintainer incaricato. Quest'ultimo, rifacendosi al modello delle *community open source* svolge il ruolo di riferimento tecnico e di laboratorio per le comunità dei Riusanti (anche non ICT) di ognuna delle soluzioni o buone pratiche presenti nel Repo.

Di seguito vengono illustrate le modalità di accesso operativo al Repository da parte di una PA interessata a riusare uno o più prodotti messi a disposizione dall'Amministrazione Cedente, proprietaria del Repo.



Descriviamo i passi presenti nello schema:

- La PA interessata al riuso accede al portale del riuso di *Developers Italia* o dello stesso Repository Locale e ottiene, utilizzando gli strumenti di ricerca offerti, la lista dei prodotti messi a disposizione insieme alla lista delle funzionalità che ciascun prodotto dispone. La PA potrebbe anche accedere direttamente alla parte pubblica del repository ottenendo le stesse informazioni;
- La PA contatta l'Amministrazione Cedente chiedendo la possibilità approfondire alcune informazioni acquisite nel Repository;
- Il Cedente, direttamente o tramite il *Maintainer*, sulla base delle richieste (FAQ o modulo predisposto eventuale) provvede a contattare o interloquire con il riusante al fine di mettere a disposizione le conoscenze necessarie per informare o per consentire l'analisi comparativa di OCPA prevista dalle Linee guida di AgID;
- Su richiesta di servizi, supporto, consulenza, il *Maintainer* attivato dal riusante interagisce direttamente e procede alle informazioni necessarie;
- La P.A accede ai prodotti presenti sul repository ed esegue l'acquisizione degli stessi secondo il modello di estrazione fissato dal Cedente (proprietario del Repo).

In alternativa esiste il metodo anonimo privo di registrazione in cui la PA accede ai prodotti presenti sul repository ed esegue l'acquisizione degli stessi secondo il modello di estrazione fissato dal Cedente (proprietario del Repo).

Insieme alle modalità di fruizione dei prodotti presenti sul repository è necessario definire le modalità con le quali l'Amministrazione cedente intende concedere in riuso i prodotti.
Le linee guida AgID, classificano le tipologie di riuso del *software* in:

- riuso in cessione semplice in cui una Amministrazione cede completamente l'applicativo a un'altra;
- riuso con gestione a carico del Cedente in cui oltre a cedere l'applicativo, l'Amministrazione proprietaria del *software* si fa carico della manutenzione dello stesso;
- riuso in *facility management* in cui l'Amministrazione cedente si fa carico oltre che della manutenzione anche della predisposizione e gestione dell'ambiente di esercizio;
- riuso in *Application Service Providing* (ASP), oggi in corso di definizione, prevede la possibilità secondo il modello *Cloud*, in cui ci potrà essere anche la presenza di un soggetto terzo che si farà carico della manutenzione e dell'esercizio del software per più Amministrazioni, che riconoscono il corrispettivo in relazione al servizio ricevuto.

In funzione del modello di manutenzione della soluzione deciso dal cedente si potranno configurare degli scenari di riuso con un differente livello di centralizzazione che l'Amministrazione titolare del Repository dovrà organizzare per il funzionamento conseguente del Repository, quali:

- *software* di proprietà del Cedente concesso a riuso passivo;
- *software* di proprietà del Cedente concesso a riuso con esigenze di evoluzione da parte dell'Ente Riusante;
- *software* NON di proprietà dell'Ente cedente preso dalla stessa come riuso passivo;
- *software* NON di proprietà dell'Ente cedente preso dalla stessa come riuso in cui saranno generate evoluzioni o personalizzazioni del prodotto.

4.3 Parametrizzazione profili sul Repository

Posti i seguenti profili di una organizzazione di utenti GitHub (tecnologia scelta per gli esempi sull'architettura del Repository):

- *Read*
- *Write*
- *Admin*
- *Owner*

Si riporta una mappatura esempio di configurazione della Azione sul repository da parte dei profili istruiti in GitHub. In essa l'*owner* (proprietario) del repository ha i permessi su tutte le possibili azioni che possono essere effettuate sul repository.

Repository action	***Read permissions***	***Write permissions***	***Admin permissions***	***Owner permissions***
Pull (read), push (write), and clone (copy) all repositories in the organization				**X**
Promote organization members to team maintainer				**X**
Convert organization members to outside collaborators				**X**
Create repositories (see "Creating repositories" for details)	**X**	**X**	**X**	**X**
Delete repositories (see "Deleting repositories" for details)			**X**	**X**
Change a repository's settings (see "Changing repository settings" for details)			**X**	**X**
Change a repository's visibility			**X**	**X**
Transfer repositories into, and out of, the organization account			**X**	**X**
Add a repository to a team (see "Adding a repository to a team" for details)			**X**	**X**

Add outside collaborators to a repository			**X**	**X**
Remove outside collaborators from a repository			**X**	**X**
Pull from (read) the team's assigned repositories	**X**	**X**	**X**	**X**
Push to (write) the team's assigned repositories		**X**	**X**	**X**
Fork (copy) the team's assigned repositories	**X**	**X**	**X**	**X**
Send pull requests from forks of the team's assigned repositories	**X**	**X**	**X**	**X**
Merge and close pull requests		**X**	**X**	**X**
Merge pull requests on protected branches, even if there are no approved reviews			**X**	**X**
Submit reviews on pull requests	**X**	**X**	**X**	**X**
Submit reviews that affect a pull request's mergeability		**X**	**X**	**X**
Open issues	**X**	**X**	**X**	**X**
Close, reopen, and assign issues		**X**	**X**	**X**

Close issues they opened themselves	**X**	**X**	**X**	**X**
Apply labels and milestones		**X**	**X**	**X**
Have an issue assigned to them	**X**	**X**	**X**	**X**
Create and edit releases		**X**	**X**	**X**
View draft releases		**X**	**X**	**X**
View published releases	**X**	**X**	**X**	**X**
Edit and delete their own comments on commits, pull requests, and issues	**X**	**X**	**X**	**X**
Edit and delete anyone's comments on commits, pull requests, and issues		**X**	**X**	**X**
Edit wikis	**X**	**X**	**X**	**X**
Create statuses		**X**	**X**	**X**

5. APPROFONDIMENTO SULLA GESTIONE DI SOLUZIONI NEL REPO

5.1 Gestione dei branch per diverse "versioni"

Può accadere, specialmente per tutti gli applicativi che hanno una diffusione sui comuni, che vi siano versioni differenti di uno stesso prodotto adattate alle necessità di un singolo comune.
In questo caso lo sviluppo/rilascio non può essere portato avanti sulla "*Active Development line*" in quanto la funzionalità o le funzionalità che vengono implementate non verranno rilasciate nell'applicativo nel suo complesso, ma sono customizzazioni specifiche per ogni singolo Comune/Ente. Anche se questa pratica è sconsigliata, tuttavia tale necessità può sorgere. In tal caso sarà necessario utilizzare un *branch* specifico applicando il pattern (*task branch*).
In generale tale pattern risulta particolarmente utile quando si verifica almeno una di queste condizioni:

- Un gruppo di lavoro deve lavorare/ha lavorato su un *task* che diverge dalla linea principale dello sviluppo;
- Quando vi è la necessità di iniziare rilasciare una nuova release, ovvero nuove funzionalità, prima che la release corrente venga consolidata.

Tenendo conto che l'utilizzo di *branch* è una pratica che può portare a numerosi inconvenienti è necessario utilizzare tale approccio solo quando i benefici attesi siano superiori all'aumento di lavoro per gestire il *branch* ed il successivo riallineamento del codice.
In tale ottica, un continuo allineamento del codice con la *mainline* di sviluppo è auspicabile in modo da ridurre al minimo i problemi nella successiva fase di merge.
Un *branch* di tale natura, alla fine della fase di sviluppo, avrà due possibili alternative:

- Viene abbandonato;
- Viene integrato nella *mainline* aggiungendo le nuove funzionalità implementate nella versione principale del prodotto.

A queste due vie canoniche si aggiunge anche una terza via che prevede che il *branch* non venga né abbandonato né integrato con la *mainline,* ma dia origine ad una versione modificata del prodotto (versione divergente). E' evidente che tale strada non può essere perseguita senza notevoli difficoltà nella gestione del prodotto stesso e quindi è necessario utilizzarla solo quando le funzionalità implementate non possono essere integrate.

5.2 Rilascio di una nuova versione

Il rilascio di una nuova versione è un'attività molto comune nello sviluppo di un nuovo prodotto.
Il rilascio viene gestito attraverso il congelamento del codice ad una data prefissata. Tale operazione prende il nome di *baseline*, attività attraverso la quale le diverse versioni degli *items* vengono congelate e raggruppate simbolicamente. La *release* è la promozione della baseline che viene resa visibile all'esterno dell'organizzazione che l'ha gestita. Tale *release* sarà visibile nel repository di Github.
La *release* viene etichettata con un numero la cui struttura riflette la tipologia di interventi che sono stati applicati al prodotto. La numerazione è del tutto arbitraria e ciascuna organizzazione (ovvero fornitori esterni) può adottare una qualsiasi tipologia di numerazione purché venga normata e definita e sia univoca per ciascuna release.
Il rilascio di una nuova versione deve essere notificato al catalogo AgID in modo che il codice sia sempre aggiornato.

Eventuali *bug fixing* su una determinata versione devono essere gestiti tramite la creazione di un *branch* di sviluppo. Al termine della fase di sviluppo le modifiche apportate devono essere rilasciate nella versione corrente e verrà richiesta una nuova baseline a una successiva release la cui numerazione rispecchierà l'intervento o gli interventi effettuati.

5.3 Gestione degli issue

Durante l'utilizzo di un prodotto possono manifestarsi delle anomalie di funzionamento.
Il sistema di gestione del repository permette la gestione e la tracciatura degli *issue*. Tramite la gestione degli *issue* è possibile tracciare i *task*, nuove funzionalità ed i *bug*. A ciascun *issue,* il sistema assegna un numero univoco.
Il sistema consente l'apertura di un *issue* tramite un'apposita interfaccia. Le anomalie segnalate dagli utenti del prodotto o da altri sviluppatori devono essere gestite tramite tale sistema. Il *software* di gestione del repository deve permettere di catalogare e organizzare gli *issues* attraverso l'uso di *Milestone, labels* ad assegnatari. La *milestone* stabilisce in quale *release* futura il *bug* verrà risolto o la nuova funzionalità verrà implementata. L'utilizzo dei *labels* permette di categorizzare gli *issue* ed è un altro modo per organizzare le nuove *features* o *bugs.*
Infine, ciascun *issue* può essere assegnato ad un membro del team di lavoro per il successivo sviluppo ed implementazione. La gestione degli *issues* tramite GitHub risulta particolarmente utile al fine di evidenziare eventuali anomalie presenti nel software così che l'utente finale possa subito avere un riscontro su eventuali problemi (*known issue*).
Anche se, come detto, GitHub è il repository "statico" delle soluzioni *software*, è importante tracciare gli *issues* su tale sistema per permettere all'utente, che utilizza il prodotto/progetto, di avere un quadro d'insieme del prodotto, sul grado di attività della "comunità" che lavora al progetto stesso.

5.4 Apporto esterno al repository

Nell'ottica di condivisione della conoscenza sia dal punto di vista del *software* che da quello documentale, organizzazioni esterne ("*community*") possono contribuire a vario titolo allo sviluppo di quanto già realizzato, dal punto di vista *software*, di conoscenza o *best practice*.
Il sistema di gestione del repository favorisce questo approccio collaborativo ad un progetto tramite l'utilizzo di *pull-request*. Ciò consente di informare i gestori del progetto che alcune modifiche sono state apportate o al codice sorgente o alla documentazione o più in generale ad uno o più *items* del progetto stesso. Tali modifiche rimangono in attesa di accettazione fino a quando il gestore del progetto provvederà ad accettarle o rifiutarle.
In questo caso, è opportuno verificare che le modifiche apportate siano corrette e che non abbiano alterato il codice. A tale scopo, esistono dei tool automatici che permettono di effettuare la compilazione in *background* del progetto e di verificare se le modifiche sono compatibili con il progetto stesso.

5.5 Controllo della qualità dei prodotti riusabili

Al fine di automatizzare e garantire che le *release* rilasciate siano in linea con gli *standard* qualitativi attesi per un prodotto riusabile è necessario provvedere ad evolvere gli attuali standard di sviluppo oramai obsoleti e non più in linea con le attuali tecnologie.

Oltre al mero rilascio del codice sorgente è necessario fornire tutti gli strumenti che permettano di valutare la qualità del prodotto e del codice. Tale valutazione può essere fatta a diversi livelli e con diversi strumenti *open-source*. Tutto questo dà valore aggiunto al progetto e garantisce il soggetto che prende in riuso il prodotto della qualità di quanto rilasciato e delle modalità seguite nello sviluppo.
Esistono diversi livelli di intervento:

- *Unit Test*
- *Integration test*
- *Front-end Test*
- *Performante/Stress test*

Oltre a questi livelli di test è necessario tracciare la qualità del sorgente e dell'intero processo alla base del progetto.

5.6 Qualità di base (Unit Test)

Gli *unit test* sono i test a più basso livello delle singole unità atomiche alla base del prodotto. Per unità atomiche si intendono le singole funzioni (nel caso di linguaggi procedurali), singole classi (nei linguaggi ad oggetti). Per questi tipi di test si possono utilizzare prodotti *open-source/free* diventati oramai dei punti di riferimento: *JUnit, Mockito*.

5.7 Qualità dei layer (Integration test)

La valutazione della qualità dei diversi *layer* architetturali di un prodotto è fondamentale per comprendere anche la qualità dell'intero prodotto. La sua valutazione è importante anche per capire le possibilità di integrazione che un prodotto può offrire. I test di integrazione valutano la qualità di quanto realizzato a livello di integrazione fra i diversi *layer* applicativi garantendo ,da un lato la separazione logica, dall'altro che i diversi livelli siano tra loro correttamente integrati. A tale scopo uno tra i possibili prodotti è *Arquillian.*

5.8 Test front-end

La qualità del *front end* riveste una particolare importanza in quanto è il punto di contatto tra il prodotto e l'utente. Al fine di garantire standard qualitativi adeguati è necessario effettuare test massivi su questo *layer* architetturale. A tale scopo può essere utilizzato *Selenium*.

5.9 Performance

Un aspetto fondamentale è la valutazione del carico che è in grado di gestire e l'eventuale possibilità di poter scalare sia verticalmente che orizzontalmente. Tali informazioni possono essere dedotte utilizzando i test di *performance*. Questo aspetto è di solito molto trascurato ma, all'atto pratico, è quello che incide maggiormente sulla qualità percepita dall'utilizzatore finale.
Al fine di poter valutare la qualità prestazionale di un prodotto esistono tool in grado di "stressare" un applicativo sotto diversi punti di vista (es: richieste concorrenti, sessioni, memoria sotto stress, ecc.). Uno tra questi prodotti è *JMeter*.

5.10 Qualità del processo e affidabilità della release

Al fine di poter valutare la qualità dell'intero processo dalla progettazione al rilascio della *release,* è necessario che i prodotti rilasciati prevedano l'integrazione con sistemi automatici di compilazione e test. Tali strumenti consentono anche di gestire con più semplicità i rilasci in quanto garantiscono l'affidabilità di quanto rilasciato.

Un esempio è *Travis-ci* (https://travis-ci.org/) che permette di automatizzare il processo di *build d*el progetto (e dei *test case*) e può essere facilmente integrato con GitHub. Questo sistema è gratuito per i repository pubblici, cioè per tutti i repository per cui non è previsto nessun controllo di accesso mentre ha un costo di utilizzo per i repository privati.

Il vantaggio di utilizzo di tale sistema è evidente, infatti questo consente di avere la sicurezza che il codice rilasciato sul repository sia corretto e che sia possibile produrre una versione compilata.

6. LICENZE PRODOTTI NEL REPOSITORY

A riguardo, per non sovrapporsi a documenti già presenti, si rimanda per semplicità e immediatezza di descrizione alle Linee guida AgID del 9 maggio 2019, Allegato C: Guida alle licenze Open Source.