Butterfly

Indice

Chi siamo 1
Il problema 2
La nostra soluzione 3
Scelte e preferenze tecniche 4
Obiettivi del progetto 5
Riferimenti aziendali 6

Chi siamo

Da oltre 30 anni Imola Informatica aiuta i propri clienti a sfruttare l'innovazione tecnologica nei processi organizzativi e nella progettazione di sistemi informativi complessi. Lavorando per i principali gruppi finanziari e assicurativi, affianca imprese e startup per gestire il cambiamento tecnologico e culturale necessario per rimanere al passo con la costante corsa all'innovazione propria dell'universo **IT**.

Il progetto proposto dal corso d'Ingegneria del Software dell'università di Padova ha suscitato il nostro interesse poiché offre all'azienda la possibilità di avvicinarsi ai programmatori del domani offrendoci cosi modo di mostrare strategia lavorative, modelli e strumenti tecnici in uso nelle realtà enterprise con cui lavoriamo.

L'esperienza aziendale accumulata ci ha permesso di capire che la chiave per un buon prodotto di successo è la collaborazione. Per questo motivo la nostra azienda è sempre stata molto attenta al mondo dell'open source incoraggiandone lo sviluppo e l'impiego sia internamente sia presso i clienti che a noi si rivolgono.

Fedeli a questo principio riteniamo fondamentale, ai fini di questo progetto, sposare il mondo dell'OpenSource **rendendo disponibile tutto il Software prodotto** attraverso un repository pubblico accompagnato da una licenza di tipologia **MIT License**, **Apache License** o **GNU General Public License (GPL) 3**.

Il problema

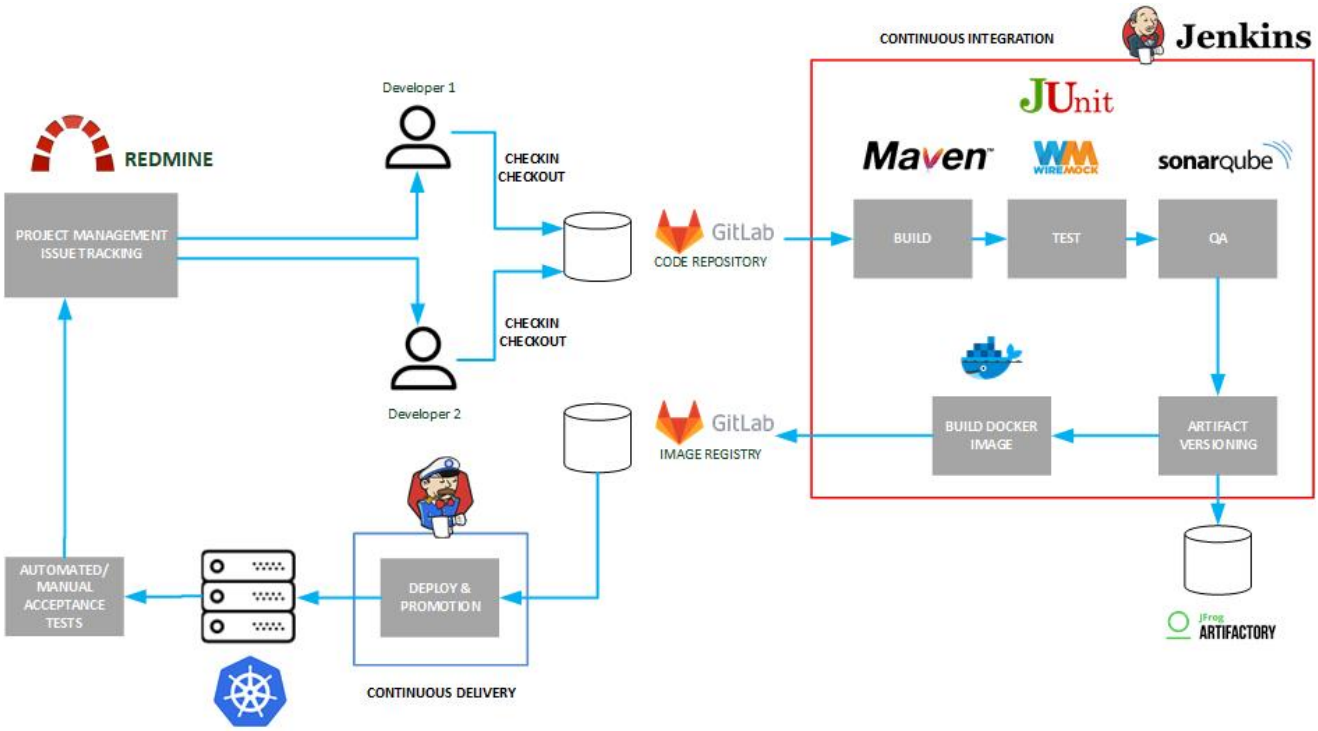

Figura 1 - Software Lifecycle

La figura soprastante è un esempio di disegno architetturale ad alto livello dei processi di Continuos Integration[i] e Continuos Delivery[ii] all'interno di una realtà enterprise di grandi dimensioni. Per costruirlo sono stati utilizzati diversi strumenti come:

- **Git**[iii] per il versionamento
- **Jenkins**[iv] per la continuos integration (build e testing ad ogni versionamento del codice sorgente)
- **SonarQube**[v] per l'analisi statica della qualità del codice
- **Redmine**[vi] come gestore di progetto per tracciare le attività
- etc.

La maggior parte di questi strumenti fornisce *out-of-the-box* dei meccanismi di segnalazione che permettono la notifica di eventuali problematiche riscontrate nelle varie fasi. Ognuno di questi strumenti ha, però, un proprio meccanismo specifico di esposizione dei messaggi/segnalazioni, spesso con limitate capacità di configurazione.

I limiti sopra indicati sfociano spesso nella necessità per l'utente di interfacciarsi con molteplici dashboard specifiche, ognuna delle quali con propria struttura. Spesso, poi, queste interfacce sono anche di difficile accessibilità, sia a causa della complessità applicativa, sia a causa di limitazioni di visibilità in rete. Spesso ciò risulta in una situazione di poca flessibilità in ordine

alla gestibilità da parte del destinatario finale della segnalazione (ad es. la persona può non essere in grado di accedere fisicamente alla rete aziendale, etc.).

Il progetto Butterfly nasce dalla necessità di accentrare e standardizzare queste segnalazioni oltre a permettere una gestione automatizzata e personalizzabile di quest'ultime.

La nostra soluzione

Attraverso un pattern di Publisher/Subscriber[vii] è possibile sviluppare una serie di componenti che si interfaccino con gli strumenti, recuperino o intercettino le segnalazioni e provvedano a riportarle nella forma desiderata dall'utilizzatore finale.

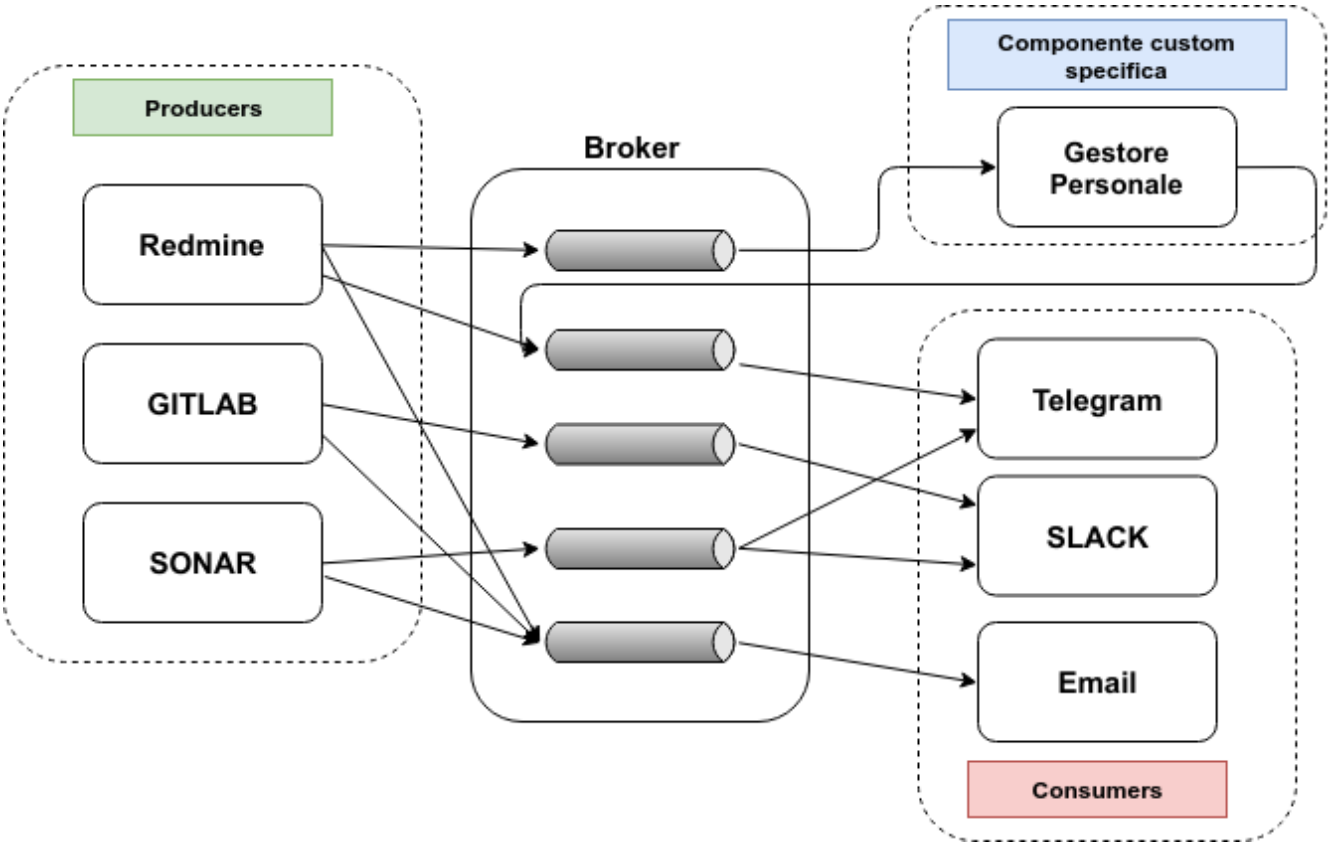

Figura 2 - Visione alto livello Butterfly

Butterfly sarà, pertanto, composto da quattro tipologie di componenti:

- **Producers**:
  I componenti producer avranno il compito di recuperare le segnalazioni e pubblicarle, sotto forma di messaggi, all'interno dei Topic[viii] adeguati.
  - **Redmine**: è richiesta la creazione di un componente applicativo in grado di recuperare le informazioni dal gestionale Redmine (attraverso la ricezione di Email o tramite batch che effettua una raccolta delle segnalazioni Redmine) e le inserisca nel Topic appropriato

  - **GitLab**: è richiesta la creazione di un componente applicativo in grado di effettuare una raccolta delle commit di git che segnali l'eventuale presenza di alcune keyword (es. TODO, WORKAROUND, BUG, etc.) all'interno di essi e le inserisca nel Topic appropriato. Git fornisce un meccanismo di estensione tramite l'utilizzo di web hook. Ne viene consigliato l'utilizzo per il recupero delle informazioni.

  - **SonarQube**: è richiesta la creazione di un componente applicativo in grado di indirizzare le segnalazioni Sonar all'interno dei Topic desiderati.

- **Broker**:
  Strumento per istanziare e gestire i Topic.

- **Consumers**:
  I componenti consumer avranno il compito di abbonarsi ai Topic adeguati, recuperarne i messaggi e procedere al loro invio verso i destinatari finali.

  - **Telegram**: è richiesta la creazione di un componente applicativo Consumer che, agganciato al Topic designato, effettui una segnalazione Telegram verso la persona desiderata.

  - **Slack**: è richiesta la creazione di un componente applicativo Consumer che, agganciato al Topic designato, effettui una segnalazione Slack verso la persona desiderata.

  - **Email**: è richiesta la creazione di un componente applicativo Consumer che, agganciato al Topic designato, effettui l'invio di una email verso la persona desiderata.

- **Componente custom specifico**:
  Per componente custom specifico si intende un componente mirato a risolvere un'esigenza specifica dell'azienda.

  - **Gestore Personale**: è richiesta la creazione di un applicativo Consumer/Producer in grado di recuperare i messaggi da un Topic designato e, in base alla lettura di alcuni metadati, effettuarne il reindirizzamento verso la persona più appropriata attraverso i canali di comunicazione da lei desiderati.

Scelte e preferenze tecniche

La nostra azienda da sempre ama esplorare nuove soluzioni tecnologiche all'avanguardia e, pertanto, predilige non imporre tecnologie specifiche per lo sviluppo degli applicativi o come strumento Broker.

Di seguito sono elencate delle possibili scelte e preferenze tecniche consigliate dall'azienda; la dicitura **"si consiglia"** sta ad intendere tecnologie non obbligatorie ma fortemente consigliate al fine di poter meglio usufruire delle competenze dei referenti aziendali; mentre la dicitura **"si richiede"** impone un requisito difficilmente contrattabile.

- Si consiglia di utilizzare Java (versione 8 o superiori), python[ix] o nodejs[x] per lo sviluppo dei componenti applicativi.
- Si richiede che le applicazioni sviluppate rispettino, per quanto applicabile al componente specifico, i 12 fattori esposti dal documento "The Twelve-Factor App"[xi]
- Si consiglia l'utilizzo di Apache Kafka[xii] come Broker.
- Si richiede l'utilizzo della tecnologia di containerizzazione Docker[xiii] per l'istanziazione di tutti i componenti.
- Si richiede che i componenti espongano, in aggiunta ad eventuali altri protocolli richiesti per l'interazione con il servizio specifico, delle API Rest[xiv] attraverso le quali sia possibile utilizzare l'applicativo.
- Si richiede che tutte le componenti applicative siano correlate da test unitari e d'integrazione. Inoltre è richiesto che il sistema venga testato nella sua interezza tramite test di sistema. I punteggi minimi verranno concordati una volta individuate, con l'aiuto dei referenti aziendali, le metriche software più adeguate.

Eventuali alternative potranno venire discusse con i gruppi aderenti al capitolato ed i referenti aziendali durante lo svolgimento del progetto attraverso i canali di comunicazione descritti nella sezione Riferimenti aziendali.

Obiettivi del progetto

L'aspettativa minima per la conclusione del progetto dovrà comprendere:

- Almeno 2 componenti applicativi Producer (priorità: Redmine > Git > Sonarqube), 2 componenti applicativi Consumer (priorità: Telegram > Email > Slack) oltre al componente custom Gestore Personale composti da:
    - Codice sorgente versionato su repository pubblico con una delle licenze OpenSource proposte.
    - Bug reporting
    - DockerFile che permetta l'istanziazione della componente applicativa.
    - README, opportunamente versionato insieme al codice sorgente, con documentazione delle API esposte dal servizio e le istruzioni per il suo utilizzo.
- DockerFile (o ComposeFile in caso di bisogno) che permetta l'istanziazione del componente Broker con le configurazioni necessarie. Il file dovrà essere versionato in un repository git insieme a:
    - Componenti configurativi necessari (script, file cfg, etc.)
    - README con le istruzioni per l'avvio e documentazione delle configurazioni custom selezionate.

- Documentazione su:
    - Scelte implementative e progettuali effettuate e relative motivazioni
    - Problemi aperti e eventuali soluzioni proposte da esplorare

Riferimenti aziendali

L'azienda, per il progetto, metterà a disposizione figure di diverso livello in modo tale da poter coprire nella maniera più appropriata le esigenze degli studenti.

In particolare seguiranno il progetto:

- Due lavoratori Junior, tra i 2 e i 3 anni d'esperienza in azienda, che si occuperanno di dare supporto da un punto di vista tecnico agli studenti.
- Un Senior dell'azienda, con oltre 20 anni d'esperienza, che fungerà da responsabile del progetto lato aziendale, nonché fornirà supporto per quanto riguarda le decisioni architetturali degli studenti e, nel caso gli Junior non fossero in grado di fornire il supporto adeguato, anche per quelle tecniche.

Inoltre l'azienda mette a disposizione, in caso di bisogno, dei server nei quali gli studenti potranno effettuare le installazioni dei componenti applicativi sviluppati.

A causa della distanza tra l'università di Padova e la sede Imolese dell'azienda, le comunicazioni fra i gruppi e i referenti aziendali avverranno, principalmente, tramite chat (preferibilmente Telegram) o tramite videochiamate (Hangout o Skype). In caso di necessità sarà comunque possibile organizzare incontri di persona e/o definire ulteriori strumenti di comunicazione con i team durante il corso del progetto.

---

[i] https://martinfowler.com/articles/continuousIntegration.html
[ii] https://martinfowler.com/bliki/ContinuousDelivery.html
[iii] https://git-scm.com/
[iv] https://jenkins.io/
[v] https://www.sonarqube.org/
[vi] https://www.redmine.org/

[vii] In questo schema, mittenti e destinatari di messaggi dialogano attraverso un tramite, che può essere detto *dispatcher* o *broker*. Il mittente di un messaggio (detto *publisher*) non deve essere consapevole dell'identità dei destinatari (detti *subscriber*); esso si limita a "pubblicare" (in inglese *to publish*) il proprio messaggio al *dispatcher*. I destinatari si rivolgono a loro volta al *dispatcher* "abbonandosi" (in inglese *to subscribe*) alla ricezione di messaggi. Il *dispatcher* quindi inoltra ogni messaggio inviato da un *publisher* a tutti i *subscriber* interessati a quel messaggio.

In genere, il meccanismo di sottoscrizione consente ai *subscriber* di precisare nel modo più specifico possibile a quali messaggi sono interessati. Per esempio, un *subscriber* potrebbe "abbonarsi" solo alla ricezione di messaggi da determinati *publisher*, oppure aventi certe caratteristiche.

Questo schema implica che ai *publisher* non sia noto quanti e quali siano i *subscriber* e viceversa. Questo può contribuire alla scalabilità del sistema. (fonti: https://it.wikipedia.org/wiki/Publish/subscribe)

[viii] I messaggi all'interno di un *Broker* vengono contenuti all'interno di dei "canali". Quando i subscriber si "abbonano" ad uno di questi canali si stanno abbonando ad uno specifico argomento (in inglese *Topic*). Pertanto per *Topic* si intende un canale di messaggi, presente all'interno di un *Broker,* associato ad uno specifico argomento (es. Topic Redmine – canale in cui sono presenti tutti i messaggi provenienti da Redmine). In base al *Broker* utilizzato la

creazione di questi Topic può essere automatizzata (ogniqualvolta un *Producer* pubblica un messaggio di un nuovo argomento genera un nuovo Topic associato ad esso) o manuale (è il Broker stesso a definire quali Topic sono messi a disposizione dei Producer scartando eventuali nuovi argomenti).

[ix] https://www.python.org/
[x] https://nodejs.org/it/
[xi] https://12factor.net/
[xii] https://kafka.apache.org/
[xiii] https://docs.docker.com/
[xiv] https://www.restapitutorial.com/