Politecnico di Torino

Dipartimento di Ingegneria Gestionale e della produzione

Corso di Laurea Magistrale in Ingegneria gestionale

Percorso Informatico

SVILUPPO AGILE DI APPLICAZIONI:

LE PIATTAFORME LOW - CODE

Relatore: Prof. Morisio Maurizio

Candidato: Lorenzano Leandro

Anno accademico 2021/2022

RINGRAZIAMENTI

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.

Ringrazio il mio relatore Professor Morisio, per aver accettato la mia tematica di tesi, mostrando un interesse che mi ha motivato a far del mio meglio durante la stesura. Lo ringrazio, inoltre, per la disponibilità offerta durante tutta la fase di scrittura.

Ringrazio di cuore la mia famiglia. Grazie per avermi sempre sostenuto e per avermi permesso di portare a termine gli studi universitari.

Un ringraziamento particolare va al mio tutor aziendale Loris Gabriele, al Manager Massimo Pacileo e a tutti i colleghi che mi sono stati vicini, che in questi cinque mesi di lavoro, hanno saputo guidarmi, con suggerimenti pratici, nelle ricerche e nella stesura dell'elaborato.

Infine, vorrei dedicare questo piccolo traguardo a me stesso, che possa essere l'inizio di una lunga e brillante carriera professionale.

INTRODUZIONE

La creazione di un software potente è vitale per le organizzazioni, poiché la necessità di una rapida trasformazione supera il tradizionale approccio unico, in cui grandi team di sviluppatori costruiscono soluzioni proprietarie. A tal proposito, negli ultimi anni, si sta sempre più affermando la metodologia di sviluppo software "Low-Code" e con essa, le varie piattaforme che la implementano. Le piattaforme low-code consentono agli sviluppatori di creare applicazioni attraverso interfacce grafiche piuttosto che con i metodi di codifica tradizionali e aiutano a colmare il divario tra gli stakeholder aziendali e gli sviluppatori esperti, creando un linguaggio visivo comune per collaborare più efficacemente. I team che lavorano con strumenti low-code sono più agili, forniscono valore più rapidamente e lavorano con gli stakeholder in modo più efficace. Collaborare con le parti interessate in un linguaggio visivo comune permette di concentrarsi sulla spiegazione della logica di business piuttosto che sul codice. Con cicli di feedback più brevi, i team diventano più produttivi, liberando risorse limitate per affrontare il backlog dei progetti in continua espansione. Al fine di esplorare il mondo della LCPD, si è deciso di riservare un intero capitolo ad una piattaforma in particolare, Pega Cosmos, offerta dall'azienda Pegasystems, al fine di esplorarne le funzionalità e di riportare eventuali benefici dall'adozione di tale prodotto nelle varie aziende. Ad affiancare le piattaforme, sono presenti le diverse tipologie di organizzazione del lavoro come ad esempio Waterfall o Agile. In caso di sviluppo di prodotti riguardanti applicativi, l'esperienza maturata da numerosi project manager ha evidenziato come l'approccio Agile ha permesso di ottenere una serie di benefici non ottenibili attraverso l'approccio a cascata. A tal fine, si è deciso di trattare tale tematica e di approfondire, in particolare, il framework Scrum. L'approfondimento inserito in tale tesi ha consentito al tesista di poter conseguire abilmente la certificazione "Scrum Fundamentals Certified" che attesta la conoscenza del framework Scrum. Durante tutta la tesi verranno raccolte le esperienze di utilizzo di varie categorie di persone ma, per poter fornire un'esperienza diretta tipica di un utente avente una conoscenza pari a zero della piattaforma, si è deciso di riportare l'esperienza diretta

del tesista riguardante lo sviluppo di un applicativo avente lo scopo di automatizzare un processo aziendale, utilizzando il framework Scrum.

INDICE

1. PIATTAFORME LOW-CODE

Il Low-Code è un metodo per lo sviluppo di applicazioni che consente agli utenti di creare app utilizzando la funzionalità di trascinamento e rilascio di componenti precostruiti attraverso un'intuitiva interfaccia visiva, con poca o nessuna esperienza o conoscenza nell'ambito della scrittura di codice. Per far ciò sono state create le Low-Code Development Platforms (LCDPs) che nascono con l'obiettivo di aumentare la produttività e di ridurre i costi dello sviluppo e del mantenimento dei software per i sistemi aziendali. Tali piattaforme permettono, infatti, di avere all'interno dello stesso ambiente di sviluppo, numerosi strumenti della programmazione tradizionale, quali: l'IDE, gli strumenti di database management, il Graphical User Interface builder, il Mapping framework, e altri componenti ancora solitamente utilizzati dagli sviluppatori (1). Lo sviluppo di applicazioni low-code presenta tuttavia alcune sfide: sebbene l'attività di scrittura di codice manuale richiesta sia limitata, i vari team IT sono comunque coinvolti, in quanto devono guidare gli sviluppatori professionisti e non nelle loro operazioni.

1.1. Il confronto con la metodologia tradizionale

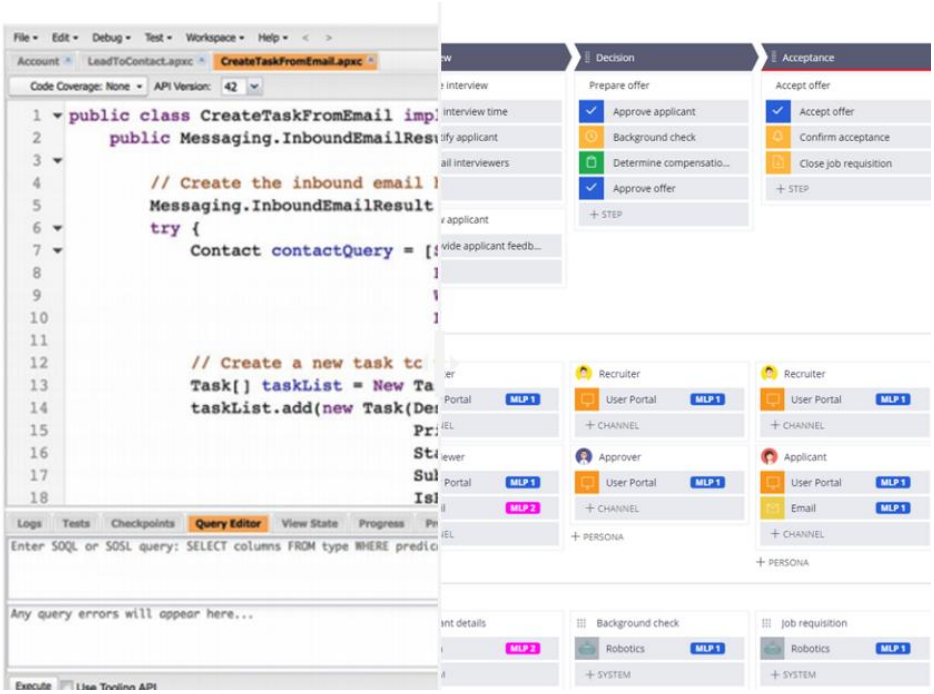

Figura 1 Tradizionale vs LC

Il confronto della metodologia Low-Code con il metodo Tradizionale è inevitabile e già in Figura 1 è possibile notare la diversità dei due ambienti di sviluppo. Troviamo, infatti, a sinistra il sistema tradizionale caratterizzato da linee di codice di difficile comprensione per un non-sviluppatore; mentre, a destra troviamo l'ambiente di una piattaforma Low-Code, che riporta già numerose informazioni facilmente intuibili. L'interfaccia delinea chiaramente processi, sotto processi, utenti e altre sezioni, e grazie a delle piccole icone permette di capire rapidamente, in linee generali, lo scopo di una determinata azione. Per una migliore analisi di confronto tra le due metodologie, si osservi la Tabella 1 sottostante che riporta una valutazione di entrambe su vari aspetti:

CARATTERISTICHE	TRADIZIONALE	LOW-CODE
Time to Market	-Codice manuale -Lento a cambiare	-Drag and drop -Componenti precostruiti -Veloce al rilascio
Ambito	-Costruire grandi progetti	-Costruire piccole soluzioni in tempi diversi grazie ai framework
Sviluppo	-Tempi di consegna lenti	-5 volte più veloce -Test efficienti -Incrementa il ROI
Manutenzione	-Costoso da supportare -Richiede uno sviluppo aggiuntivo	-Facile/rapido da aggiornare -Eccellente per la prototipazione
Integrazione	- Tempo e investimenti onerosi -Richiede sviluppatore/i e documentazione -Soggetto a ritardi nei test	-Debug in tempo reale -Creare servizi web/API senza codice
Rilascio	-Lento e complesso -Molteplici passaggi che richiedono risorse di sviluppo	-Sviluppo su più ambienti -Cloud o in-loco

Tabella 1 Sviluppo tradizionale e sviluppo Low-Code a confronto **(1)**

Il Low-Code grazie ai suoi componenti precostruiti riesce a far risparmiare molto in termini di tempo, a differenza del sistema tradizionale che spesso non fornisce porzioni di codice precompilate e nei pochi casi, l'utilizzo di questo può causare problemi di compatibilità dovuti all'integrazione. L'ambiente di sviluppo segmentato permette maggiori rilasci e test efficienti al termine di ognuno di essi, garantendo una superiore affidabilità rispetto al modello base. Inoltre, tale ambiente può essere condiviso da più programmatori, favorendo una maggiore collaborazione e produttività. La manutenzione del prodotto richiede l'accesso al codice e spesso questo può causare problemi in termini di comprensione dello stesso, rallentando così il processo effettivo di assistenza. Questo difficilmente accade nelle piattaforme Low-Code dove tutto è ben organizzato e di facile intuizione grazie all'interfaccia grafica. Come citato prima, l'integrazione rimane un problema del sistema tradizionale che spesso richiede il lavoro di numerose risorse; nelle LCDPs questo problema viene arginato dalla creazione in no-code di API o servizi web che agevolano di gran lunga tale processo.

1.2. Previsioni di crescita del mercato

La diffusione delle piattaforme Low-Code sembra esser destinata ad aumentare nei prossimi anni. Numerose ricerche e previsioni confermano questo andamento ascendente. Già nel 2019, Gartner prevedeva che nel 2024 lo sviluppo di applicazioni tramite Low-Code sarebbe stato responsabile di più del 65% delle attività totali di sviluppo. Il numero crescente di citizen developers, ovvero coloro che hanno una bassa o nulla esperienza nello sviluppo di tali applicazioni, e la crescente domanda di applicazioni per le organizzazioni aziendali sono fattori chiave che guidano l'avanzamento delle piattaforme low-code. Le operazioni IT si svolgono in ambienti dinamici che richiedono opzioni di personalizzazione rapide durante tutto il processo di sviluppo del software e le piattaforme low-code cercano di venire incontro a queste necessità. Sempre secondo Gartner, molte delle più grandi organizzazioni implementeranno numerosi strumenti Low-Code. In particolare, l'azienda afferma che entro il 2024, il 75% delle grosse imprese useranno almeno quattro strumenti di sviluppo Low-Code per la realizzazione di applicazioni IT (2). Questa è un'ottima notizia per tutte le aziende che offrono LCDPs le quali hanno già visto incrementare i propri introiti già a partire dal 2018. Inoltre, l'ascesa non sembra fermarsi e come testimonia il grafico sottostante, l'andamento dei ricavi delle piattaforme low-code è destinato a crescere:

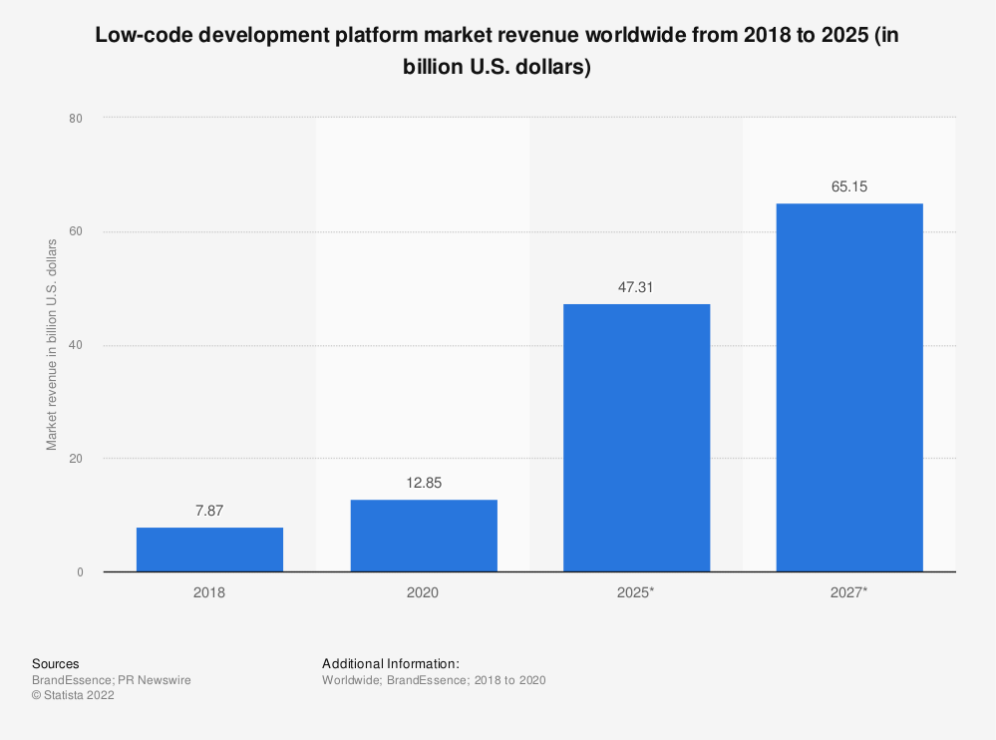

Grafico 1 Low-code development platform market revenue worldwide from 2018 to 2025 **(3)**

Le entrate globali del mercato della piattaforma low-code sono state valutate quasi 13 miliardi di dollari nel 2020 e si prevede che raggiungeranno circa 65 miliardi di dollari nel 2027. Si prevede, inoltre, che il mercato crescerà con un CAGR del 26,1% in questo periodo (3).

1.3. Magic Quadrant Piattaforme Low-Code

Al fine di generare una guida alla selezione del fornitore di LCAP, Gartner ha deciso di redigere un Magic Quadrant comprendente tutte le principali aziende fornitrici di tale servizio. Il Magic Quadrant consiste in una serie di rapporti di ricerche di mercato pubblicati dalla società di consulenza IT Gartner che si basano su metodi di analisi dei dati qualitativi proprietari per dimostrare le tendenze del mercato, come direzione, maturità e partecipanti (2).

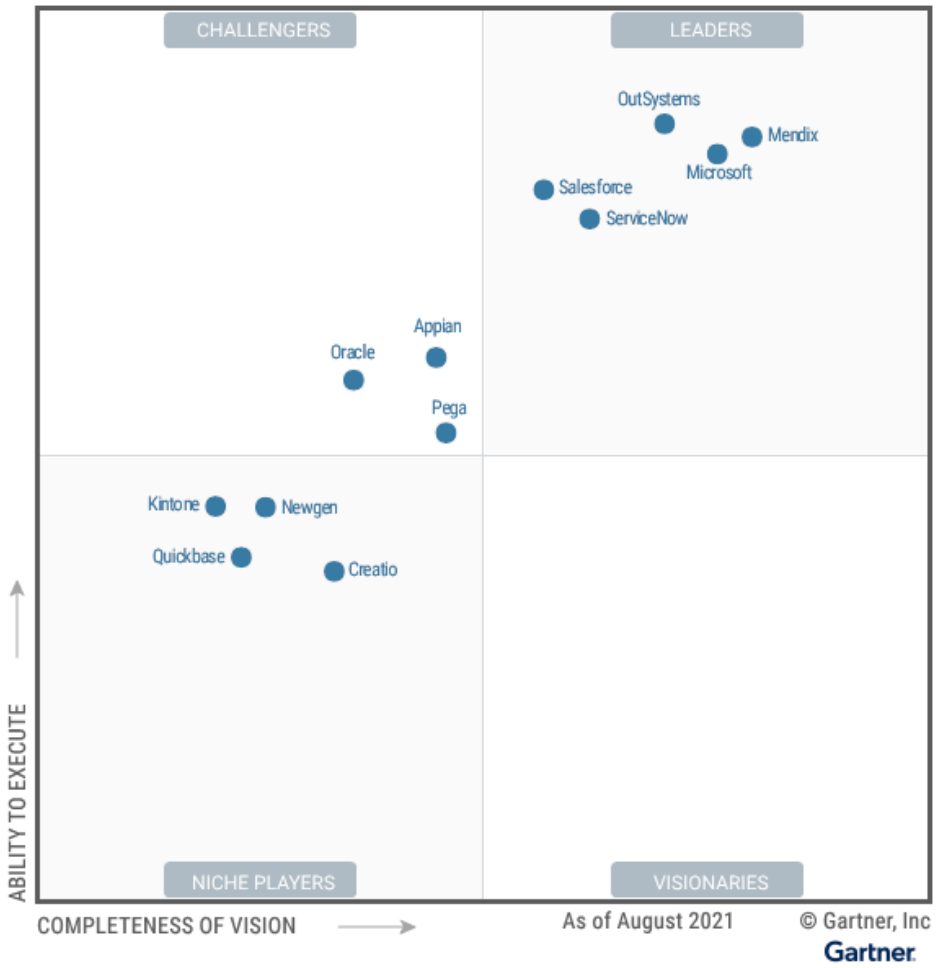

Figura 2 Magic Quadrant LCAP **(2)**

I quattro quadranti rappresentano rispettivamente le posizioni di:

- **Leaders**: dimostrano sia una forte esecuzione (in particolare in termini di prestazioni aziendali) sia una forte visione (in termini di prodotto e strategie di go-to-market). Questi fornitori si distinguono in un mercato globale altamente competitivo e coprono un'ampia gamma di organizzazioni e casi d'uso di applicazioni con le loro solide offerte LCAP
- **Challengers**: dimostrano forza nell'esecuzione ma mancano della visione dei leader (soprattutto nell'offerta e nelle strategie di go-to-market per casi d'uso e mercati più ampi). Questi fornitori hanno mostrato un ottimo operato nelle rispettive aree di interesse e stanno espandendo la propria base di clienti. Tuttavia, non hanno dimostrato la visione di espandere la loro offerta

oltre i loro clienti principali per soddisfare diversi tipi di acquirenti ed esigenze.

- **Visionaries**: I visionari sono più avanti di molti concorrenti in termini di fornitura di prodotti innovativi. Indicano la strada da seguire. Anticipano le esigenze emergenti e mutevoli del servizio clienti e si muovono nei nuovi settori ad esse associati. Hanno un forte potenziale per influenzare la direzione del mercato, ma sono limitati in termini di esecuzione o di track record. Di solito, i loro prodotti e la loro presenza sul mercato non sono ancora abbastanza completi o consolidati da sfidare i Leader, il che li colloca in una categoria a più alto rischio/ricompensa.
- **Niche Players**: hanno prodotti importanti con funzioni uniche per certi settori o aree geografiche. Possono offrire portafogli completi ma mostrare debolezze in una o più aree importanti o, per esempio, essere esperti regionali con una capacità limitata di soddisfare le esigenze globali. Possono concentrarsi sul supporto di un piccolo numero di grandi imprese o di un grande numero di PMI.

Basandoci su tale report, è possibile identificare le principali piattaforme Low-Code attualmente presenti sul mercato partendo dalle aziende proprietarie:

- **Mendix**: (Leader), è una filiale dell'azienda Siemens e la piattaforma offerta da questa azienda è Mendix Platform. I suoi punti di forza sono: la continua innovazione, offrendo nuovi strumenti low-code avanzati in grado di venire in contro a particolare necessità del cliente come nei campi dell'IoT e del digital twins; il suo prodotto permette un ottimo supporto alla fusione di vari team attraverso vari editor che semplificano questo processo anche in caso di applicazioni aziendali complesse; la promozione offerta dal conglomerato multinazionale Siemens. Secondo Gartner, Mendix dovrebbe rivedere i suoi prezzi, giudicati premium, che risultano leggermente al di sopra della media di tale mercato.
- **Microsoft**: (Leader) ha una sua personale piattaforma chiamata Power Apps che ha logicamente un elevato livello di integrabilità con tutti i prodotti Microsoft come Office, Dynamics, Azure e a ciò aggiunge la presenza di altri

importanti componenti come “Power BI” che consente un’agile analisi dei dati. Si stima che Microsoft Power Apps abbia la base di utenti più ampia di qualsiasi LCAP, principalmente a causa del diffuso utilizzo aziendale di Office 365 e Dynamics.

- **OutSystems**: (Leader), offre la piattaforma OutSystems Platform che risulta robusta in termini di sicurezza e assicura uno sviluppo rapido facilitato da numerose funzionalità basate sull’intelligenza artificiale. Dispone di un proprio editor di interfaccia utente che consente una buona personalizzazione pur fornendo vari layout precostruiti.
- **Salesforce**: (Leader), la sua piattaforma, Salesforce platform, è una delle più diffuse nel mondo e gode di una forte community online. I suoi clienti sono spesso grande aziende che utilizzano già i loro prodotti CRM e sfruttano la loro piattaforma anche per creare applicazioni industriali personalizzate per i loro clienti. Due grandi punti di debolezza dell’azienda sono il prezzo e la bassa flessibilità contrattuale che portano molti clienti a scegliere piattaforme concorrenti con prezzi più bassi o con pacchetti vendita più convenienti.
- **Appian**: (Challenger), offre la piattaforma Appian Platform. La caratteristica di questa piattaforma è la presenza di un modellatore di processi basato sulla tecnologia BPMN che permette di costruire complessi flussi di processi minimizzando di molto la necessità di codice. A questo aggiunge la segnalazione intelligente di alcune automazioni applicabili al flusso di lavoro. Il punto di debolezza è la UX design che risulta spesso limitante e restrittiva.
- **Pega**: (Challenger), la sua piattaforma è Pega Infinity Platform e oltre alla piattaforma Low-code, presenta delle suite di gestione dei processi aziendali intelligenti (iBPMS), del CRM e dell’automazione dei processi tramite robotica (RPA). La componente della piattaforma che le permette di differenziarsi nel mercato è Pega Cosmos, una libreria ricca di schermate precostruite per arricchire l’interfaccia utente e rendere gradevole l’esperienza utente. Ci sono, però, aspetti che non le permettono di essere un Leader: i prezzi elevati continuano a essere citati come ostacolo all'adozione, soprattutto per le organizzazioni più piccole. Alcune recensioni di Gartner hanno menzionato che il modello di licenza di Pega può essere complesso da comprendere; inoltre,

l'esperienza del cliente può essere compromessa dall'approccio della piattaforma basato su modelli richiedenti, spesso, delle competenze troppo elevate capaci di sopraffare i clienti che non dispongono di un team esperto.

1.4. Features comuni alle piattaforme Low-Code

Se si confrontano le principali piattaforme Low-Code presenti sul mercato, si possono delineare una serie di caratteristiche comuni. La prima riguarda la presenza di un tool per la definizione della struttura dati che spesso corrisponde ad un linguaggio di modellazione concettuale. Questo può essere o proprietario e dunque sviluppato internamente dall'azienda fornitrice del servizio, oppure può essere un classico strumento di modellazione dei processi come, ad esempio, un BPMN (Business Process Model and Notation). Un'altra caratteristica è la capacità del sistema di accedere a risorse dati esterni (documenti XML, file CSV, Database) grazie all'utilizzo delle API (Application Programming Interface). Ciò che non manca mai in ogni piattaforma è la presenza di un GUI designer, un componente in grado di sviluppare interfacce utente che semplifica e accelera l'elaborazione dell'aspetto visivo del software finale. Non sempre i componenti offerti bastano per espletare tutte le possibili funzionalità, per questo motivo alcune piattaforme offrono la possibilità di integrare con del codice tradizionale, molto spesso realizzato in Java o in Javascript, che apportano utili funzionalità riutilizzabili. In molti casi, però, vengono offerti anche dei framework molto simili a dei piccoli applicativi capaci di contenere funzionalità che sfruttano la Business Intelligence, l'Artificial Intelligence e il Machine Learning (4). AI e ML vengono spesso impiegati per aggiornamenti in tempo reale o per la rilevazione di eventuali errori, inconsistenze e altri check qualitativi. Il software, inoltre, è in grado di capire e di memorizzare il comportamento dell'utente, riuscendo così a fornire dei suggerimenti e delle raccomandazioni riguardo lo sviluppo dell'applicativo. Questa implementazione permette alla piattaforma di essere adatta anche allo sviluppo di applicazioni complesse e le offre un vantaggio competitivo sulle altre (5). Di seguito, un'immagine che riporta l'anatomia tipo di un'applicazione Low-Code:

Figura 3 Anatomia di un'App Low Code. **(6)**

1.5. Pro e Contro della metodologia

In merito all'utilizzo della metodologia Low-Code nello sviluppo di applicativi, sono state condotte numerose ricerche basate su questionari rivolti sia a sviluppatori esperti sia ai cosiddetti "citizen developers". Questo ha permesso di costruire un quadro generale dei punti a favore e dei punti a sfavore di questa metodologia. Logicamente, gli sviluppatori esperti hanno avuto modo di confrontare la metodologia in questione con la tecnica tradizionale, evidenziandone le principali differenze. Un riassunto schematico di questa analisi è presente nella Tabella 1.2, di seguito riportata (5).

Ragioni per adottare le LCDP	Diffusione della digital transformation
	Rapido Sviluppo
	Incremento reattività al business
	Privacy delle informazioni
	Riduzione Costi
	Semplicità
	Manutenibilità
	Coinvolgimento del profilo aziendale
	Minimizzazione delle inconsistenze
Fattori che impediscono l'adozione di LCDP	Mancanza di conoscenza su LCDP
	Lock-in Fornitore
	Poca fiducia sulle capacità delle LCDPs
	Problemi di Scalabilità
	Problemi di Sicurezza

Tabella 2 Motivi per adottare o evitare l'uso di LCDP **(5)**

Un questionario rivolto a più di 3300 sviluppatori IT ha riportato che i principali fattori che hanno portato all'utilizzo del LCDP risiedono nella spinta tecnologica portata dalla digital transformation che ha richiesto una forte necessità di rapidità nello sviluppo di nuovi applicativi (e-commerce, sistemi informativi aziendali, ecc.) per rispondere all'insistente domanda del mercato digitale emergente. Se dovessimo rifarci al modello tradizionale, per seguire l'andamento del mercato, vi sarebbe una cospicua richiesta di tempo al fine del rilascio di un applicativo in tempi accettabili e soprattutto per non perdere di vista il sempre più inferente time-to-

market. Quest'ultimo, allo stesso tempo, richiede un'ingente disponibilità di denaro, risorse umane formate e sforzo. Il Low-code nasce per sopperire alle mancanze di questi tre fattori che viaggiano in parallelo con l'ingente crescita della domanda del mercato. Esso, infatti, garantisce un'ottima riduzione di tempi e costi e allo stesso tempo riduce notevolmente i requisiti per poter sviluppare un applicativo. Questo grazie alle nuove interfacce grafiche che semplificano il lavoro dello sviluppatore, riducendo la programmazione ad un semplice drag-and-drop dei contenuti desiderati (5). In particolare, alcuni studi riportano che le soluzioni Low-Code permettono di risparmiare ben 50-90% del tempo di sviluppo rispetto ai sistemi tradizionali (7).

In base al prodotto che si vuole realizzare, può non servire uno sviluppatore professionista, infatti, anche il semplice utente/dipendente può migliorare e personalizzare i propri processi interni. Inoltre, il fatto che a programmare siano coloro che hanno bisogno dell'applicazione permette loro di ottenere un risultato migliore rispetto a quando un utente cerca di spiegare le proprie esigenze a uno sviluppatore professionista. Il low-code è un moltiplicatore di produttività che rende più facile per i dirigenti della linea di business e per l'IT collaborare per ottimizzare l'impatto aziendale nelle integrazioni delle applicazioni e, nel caso di applicazioni leggermente più complesse, le due figure, professionale e citizen, possono coesistere e lavorare insieme rendendo più facile la collaborazione tra business e IT (6).

Un altro punto a favore delle LCDPs è la manutenibilità poiché offrono un ambiente unico e centralizzato per tutti gli aspetti della gestione delle applicazioni, il che riduce la complessità e la difficoltà della manutenzione. Gli attori critici coinvolti nella manutenzione del software possono, infatti, collaborare in modo efficiente sulla stessa piattaforma operando solo su poche linee di codice.

L'aspetto che, invece, fa storcere il naso agli sviluppatori è la mancanza di conoscenza ed esperienza in LCDP che non permette loro di confidare nelle reali capacità della metodologia. Fortunatamente, negli ultimi anni il numero di ricerche riguardanti le piattaforme Low-Code è in forte crescita. A testimoniare la nuova tendenza è la ricerca condotta dall'Università di Scienze Applicate Konstanz che,

attraverso il seguente grafico, illustra come negli ultimi anni, c'è stato un crescente interesse dalla comunità accademica nella ricerca di pubblicazioni scientifiche riguardanti le piattaforme Low-Code (8).

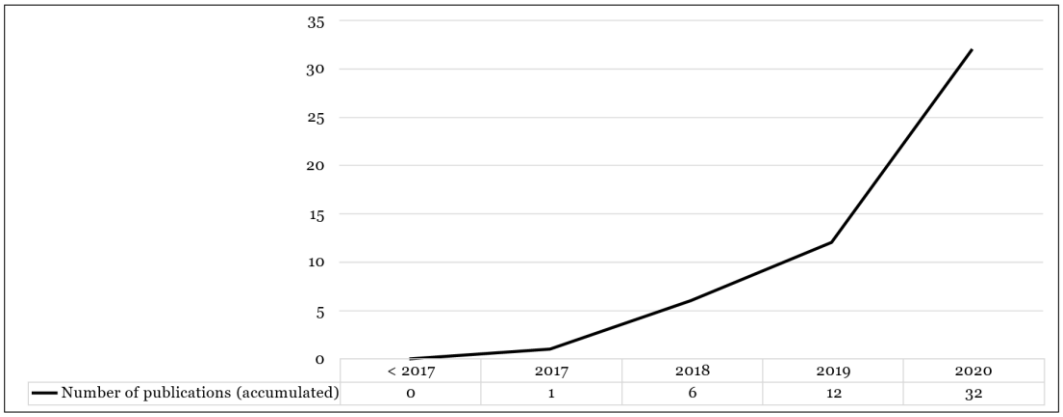

Grafico 2 Trend di ricerche 2017-2020 **(8).**

Altro punto a sfavore delle piattaforme Low-Code è la bassa scalabilità che potrebbe limitare le funzionalità e le capacità della metodologia, portando la maggior parte dei programmatori ad affidarsi al sistema tradizionale di sviluppo. Fortunatamente questo è un aspetto non comune a tutte le piattaforme e, per certi aspetti, legato al passato. Ad oggi, infatti, LCDPs come PEGA offrono un livello di scalabilità elevato in grado di supportare anche grossi carichi di volumi, essendo facilmente installabile sia su macchine dedicate che su istanze cloud. Un discorso simile può essere fatto per il tema della sicurezza dei dati: la possibilità di gestire dati sensibili richiede la presenza di alcuni standard di sicurezza per garantirne la protezione. Per questo motivo, per rassicurare il cliente, oltre a garantire lo sviluppo nativo interno del codice e, dunque degli strumenti presenti sulla piattaforma, alcune aziende ricorrono a delle certificazioni di standard di sicurezza, come ad esempio la ISO/IEC 27001:2013 (9).

L'aspetto che, invece, preoccupa il cliente è la possibilità di Vendor Lock-In: molte piattaforme si basano su sistemi proprietari che nascondono del codice contorto, o di difficile interpretazione, che non è né leggibile né riutilizzabile. Ciò rende molto difficile spostare l'app in una diversa piattaforma, generando il famoso effetto lock-in con il primo fornitore.

1.5.1. Developer Experience e review

Nei precedenti paragrafi, è stato introdotto il concetto di Citizen Developer, ovvero colui il quale ha delle competenze medio-basse in termini di programmazione e si interfaccia per la prima volta con lo sviluppo di applicativi tramite le LCDPs. Questa figura, però, non rappresenta l'unico target delle piattaforme Low-Code, infatti, anche gli sviluppatori professionisti si interfacciano con questa nuova metodologia. Nasce, dunque, il bisogno di comprendere le necessità di questo gruppo di clienti, le quali possono essere affini o completamente diverse da quelle richieste dai citizen developers. Si sviluppa così il concetto di Developer Experience che, attraverso tre grosse categorie, valuta l'esperienza dello sviluppatore nell'utilizzo delle LCDPs. Esse sono: cognizione (come viene percepita l'infrastruttura di sviluppo), impatto (le sensazioni che vengono generate nello sviluppatore) e conazione (come percepiscono il valore del loro contributo). La figura sottostante specifica quali elementi sono contenuti all'interno di ogni categoria (1):

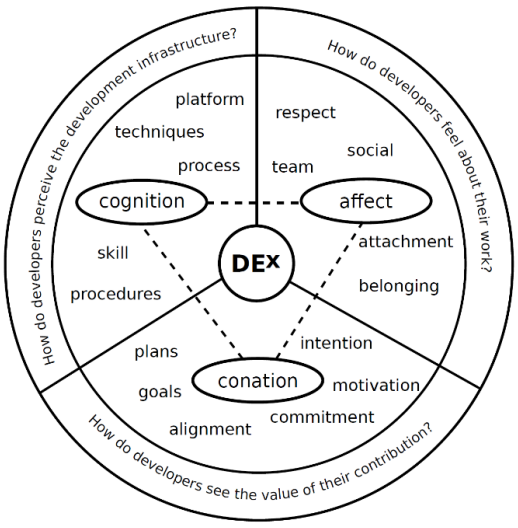

Figura 4 Development Experience **(1)**

L'esigenza di comprendere la DX nasce dall'impatto che essa può avere nella bontà del risultato finale. Alcuni studi, infatti, hanno dimostrato che lo stato di demotivazione dello sviluppatore può influenzare la produttività e le performance

negativamente, mentre uno sviluppatore motivato risulta essere molto più abile nella risoluzione di problemi analitici. Lo studio ha analizzato le tre categorie nel dettaglio ponendo delle precide domande ai candidati in merito alle sensazioni generate dall'utilizzo di LCDPs. Dai risultati si evince che la maggioranza degli sviluppatori ha percepito soddisfazione e felicità per l'essere maggiormente produttivo grazie alla piattaforma. Anche il rapporto con il cliente risulta migliorato grazie ad una maggiore trasparenza, comunicazione e feedback. Dall'altro lato, hanno percepito delle emozioni negative riguardo la possibilità di aggiungere un tocco personale al prodotto finale, sentendosi quasi limitati nel numero di soluzioni disponibili offribili al cliente. Viene meno, dunque, la creatività dello sviluppatore che è possibile trasferire solo con del codice tradizionale o aumentando in numero di tools nella piattaforma. Diventa utile prendere in considerazione delle soluzioni ibride che massimizzino sia la produttività (fornendo soluzioni pre-sviluppate) sia la creatività (permettendo di implementare delle soluzioni nella maniera tradizionale) dello sviluppatore. Per garantire un'ottima resa, è essenziale capire se le richieste del progetto incontrano completamente o parzialmente il set di soluzioni proposto dalle LCDPs. Se così non fosse, si perderebbero numerosi benefici delle LCDPs e il grado di difficoltà e complessità dello sviluppo aumenterebbe di gran lunga facendo sentire il citizen developer frustrato e il professional developer vincolato (1). Un secondo studio ha raccolto una serie di recensioni, esperienze e opinioni da due dei principali forum di informatica ovvero Stack Overflow e Reddit. La raccolta è relativa al periodo che va da Gennaio 2021 ad Aprile 2021. I risultati confermano quanto detto nei paragrafi precedenti riguardo i principali punti di forza e di debolezza ma aggiungono un ulteriore grande beneficio: le piattaforme LC possono supportare lo sviluppo anche di singole parti dell'applicazione, che potenzialmente estende le LCDPs a delle piattaforme di integrazione. Nella stessa ricerca, si sono riscontrate numerose incongruenze nelle varie recensioni delle piattaforme. Questo è simbolo che ogni LCDPs fornisce dei servizi diversi rendendo difficile capire, in generale, quali sono gli effettivi difetti o punti di forza di esse. Proprio per tale motivo, nel terzo capitolo verrà analizzata nello specifico la piattaforma PEGA e verrà confrontata con l'immaginario complessivo delle piattaforme presenti sul mercato (10).

1.6. Applicazioni e Case Study

Dopo aver analizzato nel dettaglio le caratteristiche più o meno comuni a tutte le piattaforme e aver evidenziato i relativi punti di forza e di debolezza, si passa adesso alla visione dei campi di applicazione. È possibile, infatti, delineare in quali casi conviene utilizzare queste piattaforme e quando invece è meglio evitare. Partendo dal secondo caso, se dobbiamo sviluppare un applicativo che richiede una forte personalizzazione o riguarda moduli di data science, utilizzare una LCDP sarebbe errato (11). Vengono meno, infatti, tutti i benefici precedentemente citati e risulterebbe solo una scelta forzata e frustrante per lo sviluppatore. Contrariamente a questo caso, le piattaforme Low-Code risultano avere un'ottima ed efficiente applicazione nei seguenti campi:

APPLICAZIONI	DESCRIZIONE
Portali web	Lo sviluppo no-code e low-code elimina la scrittura manuale di codice HTML e dei componenti back-end. È possibile costruire una varietà di portali con interfacce coerenti.
Sistemi line-of-business	Gli esperti di line-of-business sanno meglio di chiunque altro di cosa hanno bisogno e la piattaforma dà loro gli strumenti per costruire sistemi senza affidarsi a sviluppatori professionisti.
Processi di business digitalizzati	Le piattaforme sono ideali per creare flussi di lavoro digitalizzati end-to-end.
Mobile apps	Poiché i sistemi permettono di costruire app da componenti esistenti, sono ideali per creare Mobile apps. Possono sviluppare applicazioni per Android e iOS da una base di codice comune, risparmiando enormi quantità di tempo.
Applicazioni di Microservizi	I microservizi possono essere uniti per costruire applicazioni più grandi e complesse. No-code e low-code sono ideali per costruire questi piccoli componenti.
IoT-Apps	Molte app IoT prendono i dati raccolti dai sensori e usano queste informazioni per intraprendere automaticamente delle azioni. Un'applicazione agricola che utilizza i dati dai sensori di umidità e temperatura per controllare automaticamente l'irrigazione e l'illuminazione interna per le colture in serra è un esempio diretto in cui una piattaforma low-code sarebbe eccellente.

Tabella 1.3. Campi di applicazione delle LCDPs **(6)**.

Per comprendere al meglio le potenzialità di queste piattaforme e per capire il valore aggiunto che si riesce ad apportare ai progetti, seguiranno alcuni casi di Studio a testimonianza (6).

1.6.1. Il caso Thinkmoney

Thinkmoney è un'azienda fintech che offre ai clienti privi di conto bancario alcuni servizi finanziari che permettono loro di tenere traccia delle spese. Nel 2018, Thinkmoney è andata alla ricerca di un metodo agile per migliorare i propri servizi digitali e per accelerare il miglioramento della customer experience. Questo perché i precedenti tentativi di innovazione avevano diffuso un sentimento di frustrazione dovuti alle ingenti richieste di tempo delle varie modifiche. Nel Low-Code hanno trovato la risposta alle loro esigenze, utilizzando la piattaforma OutSystem, superando anche le difficoltà in termini di risorse umane specializzate da assumere a supporto della trasformazione. Con l'adozione di una piattaforma Low-Code, anche l'azienda ha subito delle modifiche. Sono stati, infatti, implementati dei ruoli agili in tutta l'azienda a supporto del nuovo metodo di sviluppo. I benefici ottenuti da Thinkmoney sono straordinari e sono di seguito riportati:

- Sono stati assunti e formati otto sviluppatori junior;
- Sono stati stabiliti processi agili e ruoli di Product Owner in tutta l'azienda;
- Hanno sviluppato un nuovo processo di onboarding clienti in sette settimane. Il miglioramento della CX li ha aiutati a crescere del 5% nei primi tre mesi dopo il lancio;
- Il completamento dell'onboarding cliente è migliorato del 30%;
- I costi di marketing sono stati ridotti del 21%;
- Hanno sviluppato una nuova applicazione bancaria in sole 14 settimane;
- Hanno sviluppato un nuovo sistema bancario online in 26 settimane;
- Ha chiuso la precedente piattaforma off-the-shelf risparmiando circa 300.000 £ l'anno;

- Ha aumentato esponenzialmente la propria capacità di rilascio;

Grazie a questa mossa, Thinkmoney ha vinto nel 2019 il Banking Technology Award per il “Miglior uso dell’IT nel Retail Banking” (12).

1.6.2. Il caso New York

Durante la pandemia di Covid-19, Il Dipartimento IT della città di New York ha sentito la necessità di avere dati riguardanti la situazione contagi in tempo reale in modo tale da fornire servizi a chi ne avesse bisogno. Per far ciò, ha deciso di rivolgersi ad Unqork, una delle principali aziende offerenti la realizzazione di servizi grazie alla propria piattaforma No-code, per la realizzazione di un portale online tramite tale piattaforma. Il portale consentiva alla città di raccogliere e mappare i dati sull'impatto del COVID-19, in modo da poter comunicare con i residenti e metterli in contatto con i servizi essenziali. Il portale era disponibile in 11 lingue e i rappresentanti del servizio 311 della città di New York potevano inserire informazioni per conto delle persone che necessitavano di aiuto, Il servizio è stato attivo per quasi tre settimane e risulta oggi chiuso a seguito del miglioramento della situazione pandemica. La piattaforma è stata utilizzata per creare un portale personalizzabile al fine di incontrare le necessità di città, Paesi e Stati. Il risultato è stato strabiliante: in un contesto in cui il tempo fa da padrone, gli sviluppatori sono riusciti a realizzare un portale sicuro e scalabile in sole 72 ore grazie all’ambiente intuitivo ricco di strumenti drag-and-drop (13).

1.6.3. Covid-19 Employee Safety and Business Continuity Tracker App

A causa della diffusione del coronavirus (COVID-19) in tutto il mondo, le organizzazioni leader devono garantire la salute e la sicurezza di centinaia di migliaia di lavoratori, oltre che di milioni di clienti. Questa situazione confusa e

mutevole ha richiesto tecnologie scalabili e flessibili. Nel marzo 2020, Pega ha collaborato con un'azienda sanitaria leader del settore per implementare un'app di risposta alle emergenze basata su Pega Platform™. L'app permette all'azienda sanitaria di monitorare e gestire salute, sicurezza e disponibilità dell'intera forza lavoro. Ispirandosi a questo lavoro e al relativo impatto significativo su milioni di persone, il team di Pega ha implementato l'app COVID-19 Employee Safety and Business Continuity Tracker e l'ha resa disponibile sul suo marketplace per i clienti a titolo gratuito. L'app è stata progettata come un acceleratore che può essere scaricato e configurato per essere usato così com'è o modificato in base alle esigenze dell'azienda. Le organizzazioni possono utilizzarlo per (14):

- Consentire ai dipendenti di informare il datore di lavoro sul proprio stato di salute
- Prendere le opportune misure laddove lo stato di salute dei dipendenti subisse variazioni
- Gestire la disponibilità dei dipendenti per interventi in loco o da remoto
- Monitorare il numero di dipendenti esposti al COVID-19, sottoposti al tampone o in quarantena fiduciaria
- Aiutare i dipendenti guariti a rientrare al lavoro

2. AGILE

L'industria dell'Information Technology (IT) è in continua crescita ed espansione, costringendo chi lavora in tale ambito a migliorare continuamente i propri processi di lavoro e i metodi di rilascio del software. A tal proposito, la tendenza degli ultimi vent'anni è stata quella di migrare dall'approccio Waterfall (a cascata), che consiste nella consegna del prodotto al termine del progetto, verso l'approccio Agile che implica il rilascio di piccoli incrementi di lavoro costantemente all'interno di determinati slot temporali. Questo perché l'ambito dello sviluppo prodotto richiede una certa elasticità nelle modifiche, dato che i progetti riguardanti software sono in continua evoluzione ed è consuetudine che il prodotto finale sia fortemente diverso da quello ideato inizialmente. All'inizio di un progetto, il Project Manager e il relativo team decidono sulla metodologia da utilizzare, ad esempio l'Agile. Fatto questo, si passa all'esplorazione dei vari metodi di implementazione per la realizzazione del software. Esso verrà deciso in base a chi tra i tanti soddisferà nel miglior modo possibile i requisiti progettuali. Se vengono, dunque, rispettati i requisiti citati nel precedente capitolo, si otterrà la combinazione tra la metodologia Agile, per quanto riguarda il processo di controllo e gestione del progetto, e tra l'utilizzo di una piattaforma Low-Code per lo sviluppo del prodotto in sé. La composizione di questi due aspetti, se correttamente implementati, può portare ad enormi benefici in termini di qualità, tempo e costi. Avendo già analizzato i benefici delle piattaforme Low-Code, in questo capitolo verrà analizzata nel dettaglio la metodologia Agile e suoi corrispettivi punti di forza.

2.1. I principi dell'Agile

L'Agile nasce dalla necessità di avere una metodologia snella e meno laboriosa per velocizzare il processo di rilascio software. Tale metodologia raggruppa un insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000 e fondati su un insieme di principi comuni, direttamente o indirettamente derivati dai principi

del "Manifesto per lo sviluppo agile del software" (Manifesto for Agile Software Development, impropriamente chiamato anche "Manifesto Agile") pubblicato nel 2001 da Kent Beck, Robert C. Martin, Martin Fowler e altri. Oltre allo sviluppo software, ci sono altri casi in cui l'Agile risulta efficace come ad esempio: nello sviluppo di nuovi prodotti; nei progetti digitali e high tech; nei progetti con alta imprevedibilità; nella ricerca e sviluppo e nei progetti di scoperta. L'obiettivo della metodologia è quello di rispondere ad un turbolento cambiamento nel business con:

- Forte interazione e integrazione con i clienti;
- Riprioritizzazione rapida dell'uso delle risorse;
- Consegna evolutiva, incrementale e iterativa dei prodotti;
- Generazione di valore just in time.

C'è, dunque, un cambio di paradigma in tre aspetti principali:

- Governance: da controllo sulle risorse a fiducia alle risorse;
- Processo: da prescrittivo (rule-based) a non-prescrittivo (principle-based):
- Risultati: da profitto (shareholders based) a valore (stakeholders based).

Il Manifesto Agile contiene i 12 principi sulla quale di basano tutte le metodologie appartenenti a questa categoria, essi sono:

1. La nostra massima priorità è soddisfare il cliente rilasciando software di valore, fin da subito e in maniera continua.
2. Accogliamo i cambiamenti nei requisiti, anche a stadi avanzati dello sviluppo. I processi agili sfruttano il cambiamento a favore del vantaggio competitivo del cliente.
3. Consegniamo frequentemente software funzionante, con cadenza variabile da un paio di settimane a un paio di mesi, preferendo i periodi brevi.
4. Committenti e sviluppatori devono lavorare insieme quotidianamente per tutta la durata del progetto.
5. Fondiamo i progetti su individui motivati. Diamo loro l'ambiente e il supporto di cui hanno bisogno e confidiamo nella loro capacità di portare il lavoro a termine.

6. Una conversazione faccia a faccia è il modo più efficiente e più efficace per comunicare con il team ed all'interno del team.
7. Il software funzionante è il principale metro di misura di progresso.
8. I processi agili promuovono uno sviluppo sostenibile. Gli sponsor, gli sviluppatori e gli utenti dovrebbero essere in grado di mantenere indefinitamente un ritmo costante.
9. La continua attenzione all'eccellenza tecnica e alla buona progettazione esalta l'agilità.
10. La semplicità - l'arte di massimizzare la quantità di lavoro non svolto - è essenziale.
11. Le architetture, i requisiti e la progettazione migliori emergono da team che si auto-organizzano.
12. A intervalli regolari il team riflette su come diventare più efficace, dopodiché regola e adatta il proprio comportamento di conseguenza. (15)

2.2. Le metodologie Agile e Waterfall a confronto

Ad oggi, nelle fasi iniziali del progetto, ogni project manager si trova a dover decidere tra principalmente due metodologie: Waterfall (in italiano “a cascata”) e Agile. Questi due approcci sono caratterizzati da procedure simili ma con una sequenza e temporalità diversa. Il modello Waterfall è stato chiamato così per l’analogia con la relazione fine-inizio che esiste tra le fasi del progetto. Ogni fase coinvolge diverse competenze e ha una formale accettazione e approvazione alla fine. Per tutto lo sviluppo del progetto, non sono previsti feedback con il cliente e ci si basa sui requisiti stabiliti durante la fase di planning. La figura sottostante presenta il ciclo di vita di sviluppo del software in tale metodologia suddiviso nelle varie fasi.

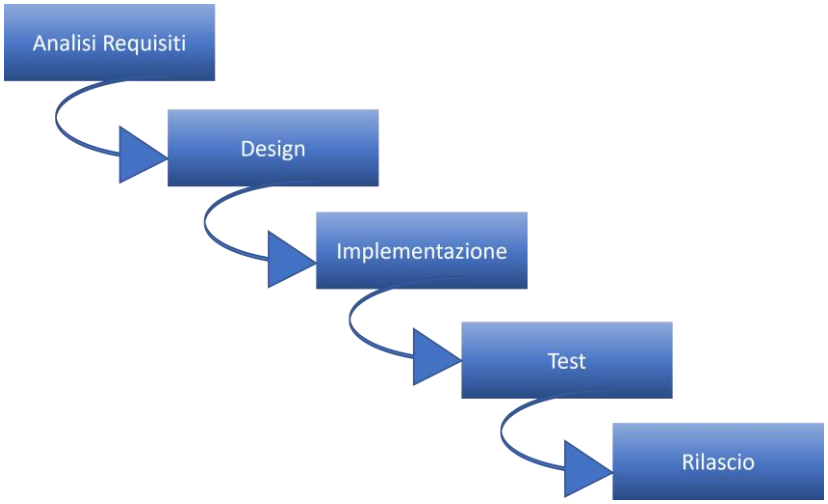

Figura 5 Steps Waterfall

L'Agile, invece, suddivide lo sviluppo del prodotto in piccoli incrementi rilasciati al termine di ogni iterazione. Si otterranno, dunque, più cicli che man mano contribuiranno allo sviluppo finale del prodotto e al termine di ognuno verrà collezionato il parere del cliente così da poterlo implementare già nell'iterazione successiva. Questo è il principale motivo per il quale si preferisce l'Agile per lo sviluppo di software. Un esempio visivo della procedura è riportato nell'immagine sottostante.

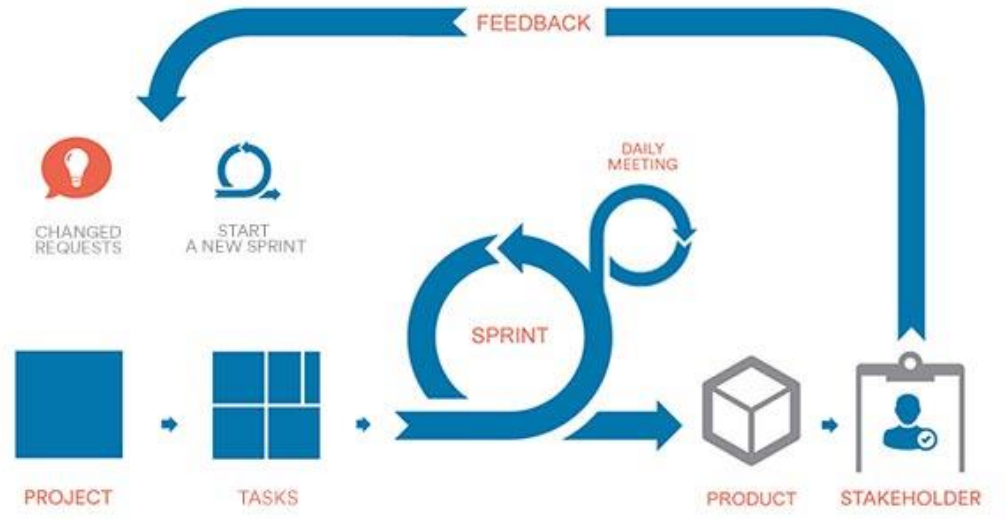

Figura 6 Processo Agile, di Silvia Mantovani, Febbraio 2019.

Le differenze tra questi due processi non si limitano solo all'aspetto organizzativo ma riguardano numerosi aspetti, soprattutto il triangolo che comprende qualità, tempo e costo. Nella tabella sottostante sono stati confrontate le due metodologie su numerosi aspetti (16):

Metrica	Waterfall	Agile
Tempistiche di pianificazione	Lunghe. I processi vanno definiti prima di partire con l'effettivo sviluppo	Corte. Il tempo per pianificare è limitato in vista dell'imminente prima iterazione
Direzione di sviluppo	Focalizzata sul piano	Focalizzata sui cambiamenti
Flessibilità	Bassa	Alta. Possibilità di modifiche dopo ogni iterazione
Tempo di rilascio di un prodotto utilizzabile	Lungo. Al termine del progetto	Breve, alla fine di ogni iterazione
ROI e Time-to-market	Fissato al termine del progetto	Flessibile, i primi rilasci possono già portare dei profitti
Forme contrattuali	a Prezzo fissato con la possibilità di avere dei premi rischi	di tipo Tempo e materiale focalizzati sul valore creato
Modifiche in fase di sviluppo	Richiedono tempo per rischedulare tutto il piano	Non richiedono ingenti tempi aggiuntivi
Rischi	Alti al termine del progetto	Bassi al termine del progetto grazie alla suddivisione nelle varie iterazioni
Focus	Processi	Valore cliente
Collaborazione team di sviluppo e cliente	Bassa	Alta, cliente come membro del team
Modi di comunicazione	Tramite il PM	con tutto il team
Relazione Venditore-Cliente	Basata su negoziazione iniziale	Basata su collaborazione fin da subito

Tabella 3 Confronto Waterfall e Agile **(16)**.

L’Agile garantisce, dunque, dei livelli di flessibilità elevati che risultano essenziali nel contesto di uno sviluppo software. Tale contesto, infatti, comprende un’area molto vasta che si propaga con nuovi e futuri standard e tecnologie ogni giorno. I linguaggi di programmazione vengono introdotti sul mercato quasi ogni mese utilizzando nuove versioni e frameworks capaci di introdurre anche nuove caratteristiche e tecnologie utili al miglioramento del progetto. A questo scopo, il project manager deve monitorare le varietà future dei linguaggi di programmazione al fine di soddisfare i requisiti del cliente e di coordinarsi con il team (17).

2.3. Ciclo di vita di uno sviluppo software in Agile

Il ciclo di vita dello sviluppo di un software consiste prevalentemente in tre macro-fasi: fase di avvio, fase di iterazioni, chiusura progetto. La prima fase, detta anche fase della scoperta, avviene prima dell’inizio del progetto. Solitamente, il team che andrà a partecipare a questa fase sarà diverso rispetto a quello che opererà nelle varie iterazioni. Spesso, questo team fa parte della sezione commerciale dell’azienda il quale riceve le varie offerte/proposte dal cliente e le analizza stimandone i requisiti, le risorse necessarie per poi dare inizio alla fase di negoziazione. Tutti questi passaggi devono essere svolti in una piccola finestra temporale che solitamente va dalle due alle quattro settimane. Dopo la firma del contratto, un project manager viene assegnato al progetto, e viene avviata la fase di pianificazione. Quest’ultima ha già inizio con la prima iterazione, rendendo il primo step leggero in termini di obiettivi da raggiungere perché il team inizierà ad impostare il progetto, raffinare i requisiti per la prossima iterazione e costruire il backlog che contiene un elenco prioritizzato dei requisiti di business e di progetto scritti sotto forma di User Story di alto livello. Una volta che questi elementi sono completati, il lavoro di sviluppo vero e proprio può essere iniziato. Nel frattempo, il project manager si assicurerà di assegnare le risorse necessarie, riunirà il team del progetto e inizierà con la prossima iterazione. A questo punto il team di sviluppo del prodotto dovrebbe avere tutte le competenze necessarie per consegnare il primo incremento del prodotto.

Ogni iterazione ha una struttura standard formata da quattro fasi:

1. Definizione obiettivi;
2. Sviluppo;
3. Revisione;
4. Rilascio incremento.

Le iterazioni continueranno finché tutti gli incrementi non saranno stati completati e si sarà raggiunto l'obiettivo finale complessivo. A questo punto, ha inizio la fase di chiusura del progetto dove l'attenzione ricade nella formale accettazione del prodotto da parte del cliente e la riallocazione del team ad altri progetti.

Figura 7 Schema Agile.

Potrebbe esserci la possibilità che sia presente al termine del progetto un'ulteriore fase chiamata "lesson learned" in cui si analizzano le criticità riscontrate durante il progetto e se ne fa tesoro per i progetti futuri.

2.4. Scrum

Come accennato precedentemente, l'Agile comprende numerosi framework di sviluppo, uno tra questi è Scrum. Tale framework è stato sviluppato da Ken Schwaber e Jeff Sutherland ed è stato pensato per risolvere problemi complessi di tipo adattivo e al tempo stesso di creare e rilasciare prodotti in modo efficace e creativo del più alto valore possibile. Scrum è leggero, semplice da comprendere ma difficile da padroneggiare. Questo perché spesso richiede che non solo cambino le modalità di sviluppo prodotto ma che anche l'azienda, o un determinato reparto, si adatti completamente alle ideologie del metodo. Questo framework è costituito dagli Scrum Team e dai ruoli, dagli eventi, dagli artefatti e dalle regole a essi associati. Le regole legano insieme i ruoli, gli eventi, gli artefatti governando le relazioni e le interazioni tra essi.

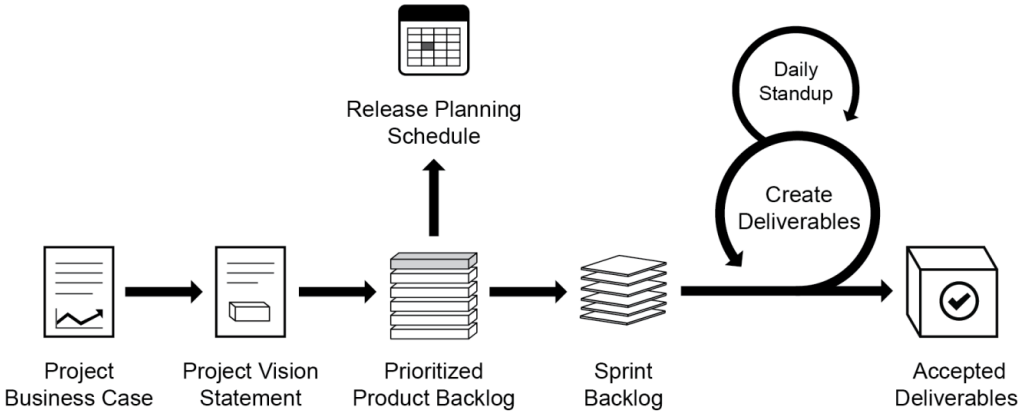

Figura 8 Scrum Framework **(18)**

I vari team sono l'essenza di questo metodo poiché forniscono flessibilità e adattabilità. La teoria del metodo si basa su tre pilastri fondamentali: trasparenza, ispezione e adattamento. Gli aspetti significativi del processo devono essere visibili ai responsabili del risultato poiché essi devono poter ispezionare frequentemente gli artefatti e l'avanzamento del singolo sprint così da individuare eventuali deviazioni indesiderate. Una volta individuate, il processo deve essere riadattato così da evitare sprechi e ulteriori deviazioni (18).

2.4.1. Lo Scrum Team

Lo Scrum team è composto da un Product Owner, dal Team di Sviluppo e da uno Scrum Master. Gli Scrum Team sono auto-organizzati e cross-funzionali e scelgono come meglio compiere il proprio lavoro invece di essere diretti da altri al di fuori del team. I team cross-funzionali hanno tutte le competenze necessarie per realizzare il lavoro senza dover dipendere da nessuno al di fuori del team. Di seguito verranno analizzati i singoli ruoli.

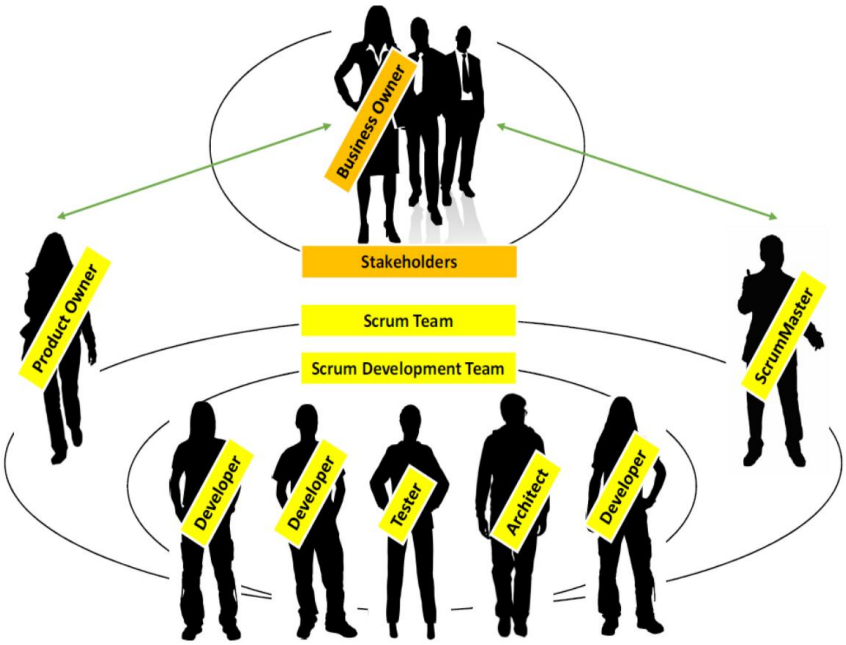

Figura 9 Lo Scrum Team **(18)**

Product Owner

È la figura che si colloca tra il cliente e il resto del team. Solitamente, è un esperto di Scrum, con un'ottima conoscenza del business e eccellenti capacità comunicative. Ha la responsabilità di massimizzare il ritorno sugli investimenti (ROI), di identificare le caratteristiche del prodotto, da tradursi in una lista di priorità, di decidere cosa dovrebbe andare in cima alla lista per il prossimo Sprint, e di riassegnare le priorità e perfezionare l'elenco con continuità. È l'unica persona responsabile del Product Backlog e le sue principali mansioni sono:

- Esprimere chiaramente gli elementi del Product Backlog;
- Ordinare gli elementi del Product Backlog per meglio raggiungere gli obiettivi e le missioni;
- Ottimizzare il valore del lavoro del Team di Sviluppo;
- Assicurare che il Product Backlog sia visibile, trasparente e chiaro a tutti e mostri su cosa lo Scrum Team lavorerà in seguito;
- Assicurare che il Team di Sviluppo comprenda gli elementi del Product Backlog al livello necessario;

Team di Sviluppo

Il Team di sviluppo è costituito da professionisti che lavorano per consegnare un Incremento di prodotto al termine di ogni sprint. È auto-organizzato e i membri decidono come tradurre le voci del Product Backlog in Incrementi. Solitamente sono cross-funzionali, ovvero i membri hanno competenze diverse ma tutte utili ai fini dello sviluppo del prodotto. Il numero di componenti varia dai tre alle nove persone, questo perché se al di sotto di tre, potrebbero mancare alcune competenze chiave per lo sviluppo; mentre, al di sopra delle nove persone, risulterebbe difficile coordinare il lavoro e dunque rende ostica la gestione complessiva dello Sprint. Chi decide di implementare Scrum deve confidare nelle capacità del Team e fidarsi ciecamente dei suoi membri.

Scrum Master

Lo Scrum Master principalmente si occupa di far eseguire i lavori secondo la guida ufficiale Scrum e cerca allo stesso tempo di comunicare i valori chiave della metodologia a tutti gli attori del processo. Può essere visto come un leader del Team di Sviluppo e come mediatore tra esso e tutto ciò che è all'esterno dello stesso Team. Si occupa di eliminare eventuali ostacoli all'avanzamento del Team di Sviluppo e forma il Team al fine di permettere l'implementazione di tecniche di auto-gestione e coordinamento. A ciò aggiunge un'altra serie di attività utili al lavoro sia del Product Owner sia dell'intera Organizzazione:

- Assicurare che obiettivi, portata e dominio del prodotto siano compresi nel miglior modo possibile da tutti i membri dello Scrum Team.

- Trovare le tecniche per una gestione efficace del Product Backlog;
- Aiutare lo Scrum Team a comprendere la necessità di avere elementi del Product Backlog chiari e concisi;
- Comprendere la pianificazione del prodotto in un ambiente empirico;
- Assicurare che il Product Owner capisca come ordinare gli elementi del Product Backlog per massimizzare il valore;
- Comprendere e praticare l'agilità;
- Facilitare gli eventi Scrum come richiesto e necessario;
- Guidare ed assistere l'organizzazione all'adozione di Scrum;

2.4.2. Le Cerimonie

Le cerimonie sono gli eventi previsti da Scrum e hanno una durata massima ben definita. Ad eccezione dello Sprint, tutte le altre cerimonie posso terminare prima della durata prevista per ottimizzare i tempi ed evitare sprechi. Ogni evento è un'occasione per poter ispezionare e adattare qualcosa e nascono con l'obiettivo di garantire i tre pilastri prima citati. I principali eventi sono: lo Sprint, lo Sprint Planning, il Daily Scrum Meeting, la Sprint Review e la Sprint Retrospective.

Sprint

Lo Sprint è l'iterazione presente in ogni ciclo del metodo Agile. Esso ha una durata variabile che oscilla tra le due e le quattro settimane, durante la quale viene creato un Incremento di prodotto potenzialmente rilasciabile. La decisione di limitare la durata dell'iterazione nasce dalla necessità di ridurre i rischi di sviluppo, la complessità del goal e permettere un più agevole monitoraggio dei progressi. Durante uno Sprint non possono essere fatte modifiche che possono mettere a rischio lo Sprint Goal (obiettivo dell'iterazione precedentemente prefissato) ed è essenziale monitorare che gli standard di qualità vengano rispettati. Prima di far

partire il primo Sprint, è fondamentale che venga definito tra Product Owner e Team di sviluppo quale sia il concetto di "Fatto". Un buon Product Owner vorrà sempre che la Definizione di Fatto sia la più vicina possibile al Potenzialmente Rilasciabile in quanto ciò aumenterà la trasparenza nello sviluppo e ridurrà i ritardi e i rischi. Se la Definizione di Fatto non è uguale a Potenzialmente Rilasciabile, allora il lavoro viene ritardato tornando a prima del rilascio comportando rischio e ritardo. Un Team Scrum dovrebbe migliorare continuamente, ciò si riflette nell'evoluzione della propria Definizione di Fatto. Nella fase iniziale di pianificazione, il Product Owner illustra la storia generale del progetto per poi scendere sempre più nello specifico, passando dalle Epic e terminando nelle varie User Stories di ciascuna Epic. Durante la presentazione delle storie, il team è invitato ad assegnare ad ognuno di esse, un punteggio basato sul numero di punti storia. Questa è una metrica utile alla pesatura di ogni storia e viene definita dal team attraverso il Planning Poker: si raggiunge la stima di ciascuna attività sfruttando il principio della "saggezza della folla" e all'unanimità (consensus). Il risultato, espresso in sforzo necessario all'implementazione di una funzionalità, è concettualmente molto vicino a un'indicazione di tempo piuttosto che di complessità, dato che ciò che conta per gli stakeholder, e dal punto di vista del business, è il time to market. Ogni membro del team di sviluppo ha in mano un mazzo di carte con una sequenza di numeri, spesso, ricavata dalla Successione di Fibonacci. In caso di discrepanze il gruppo riprende la discussione sul requisito cercando di capire i diversi punti di vista e i motivi che hanno portato a valutazioni discordanti. Il Product Owner introduce brevemente l'attività da svolgere. Quando tutti hanno un'idea abbastanza chiara su come procedere per la parte di implementazione, lo Scrum Master invita i partecipanti a scegliere una carta dal proprio mazzo e a mostrarla tutti insieme. In caso di discrepanze il gruppo riprende la discussione sul requisito cercando di capire i diversi punti di vista e i motivi che hanno portato a valutazioni discordanti. Una volta chiariti i dubbi, ciascuno sviluppatore ripete la stima scegliendo una nuova carta dal mazzo, o quella mostrata in precedenza in caso volesse confermare la propria stima precedente. Si procede così fino a quando il team non raggiunge il consensus, ovvero l'unanimità, e tutti, o quasi, mostrano la stessa carta. (19)

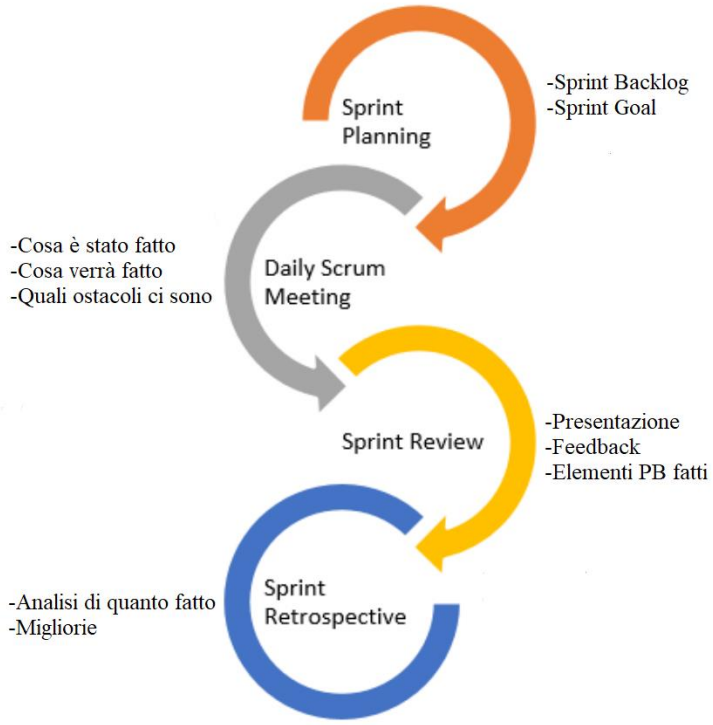

Figura 10 Cerimonie Scrum

Sprint Planning

Il lavoro da eseguire nello Sprint è pianificato durante lo Sprint Planning. Questo evento, solitamente, è suddiviso in due parti: la prima parte si focalizza sul "cosa"; mentre la seconda sul "come". A tale cerimonia, contribuisce attivamente il team di sviluppo mentre lo Scrum Master si preoccupa di fornire le informazioni necessarie alla corretta esecuzione. Il Product Owner discute l'obiettivo al quale lo Sprint dovrebbe aspirare e gli elementi del Product Backlog che, se completati durante lo Sprint, permetterebbero di raggiungere lo Sprint Goal. La sua durata massima è di otto ore e in tale tempo deve essere definito lo Sprint Goal del prossimo Sprint, come verrà organizzato il lavoro e che Incremento verrà rilasciato al termine dell'iterazione. Per definire questi tre punti, sono necessari elementi come il Product Backlog, l'ultimo incremento del prodotto e le prestazioni registrate dal team di sviluppo lo scorso Sprint. Lo Sprint Goal rappresenta l'obiettivo da raggiungere sulla base degli elementi selezionati dal Product Backlog e fornisce indicazioni al Team sulle motivazioni per le quali esso stesso sta

costruendo l'Incremento. Fatto ciò, il Team di Sviluppo deciderà quale Incremento fare e le modalità di implementazione. Queste informazioni verranno riportate nello Sprint Backlog che verrà visionato anche dal Product Owner. Una volta terminato lo Sprint Planning, il Team deve illustrare sia allo Scrum Master sia al Product Owner, come intende lavorare in quanto team auto-organizzato, al fine di raggiungere l'obiettivo prefissato.

Daily Scrum

Così come si evince dal nome, il Daily Scrum o Stand Up Meeting è un evento con ricorrenza giornaliera della durata massima di 15 minuti dove tutti i partecipanti sono in piedi. Viene pianificato ad una determinata ora e serve per pianificare il lavoro della giornata di Sprint. Ciò ottimizza la collaborazione e la prestazione ispezionando il lavoro svolto il giorno prima e prevendendo il lavoro in arrivo per i prossimi giorni dello Sprint. Durante tale evento, viene monitorato l'avanzamento dei lavori in ottica di raggiungimento dello Sprint Goal e si osservano il numero di elementi completati nello Sprint Backlog. A vigilare sulla durata e sulla corretta esecuzione c'è lo Scrum Master ma il vero attore principale del meeting rimane il Team di Sviluppo. Il Daily Scrum migliora la comunicazione, identifica gli ostacoli allo sviluppo allo scopo di rimuoverli, evidenzia e promuove il rapido processo decisionale e migliora il livello di conoscenza del Team di Sviluppo. Esso rappresenta un incontro chiave d'ispezione e adattamento.

Sprint Review

È l'evento svolto al termine di ogni Sprint al fine di ispezionare l'Incremento e aggiornare il Product Backlog. Viene analizzato tutto ciò che è stato fatto durante l'iterazione e si collabora per capire cosa migliorare nel prossimo Sprint. Dura al massimo quattro ore e c'è sempre lo Scrum Master a monitorare che tutto venga rispettato secondo il manuale Scrum. Alla Review sono presenti anche gli stakeholders invitati dal Product Owner, il quale presenterà loro quanto fatto durante lo Sprint e quali punti del Backlog sono stati completati. Il Team mostrerà cosa è andato bene e cosa no e risponderà ad eventuali domande. Viene trattata

anche una piccola parte di previsione dove si stimano le tempistiche future e i prossimi elementi da selezionare dal Product Backlog per il prossimo Sprint.

Sprint Retrospective

Al termine della Sprint Review, ha inizio la Sprint Retrospective che consiste in un ulteriore evento dove il Team si confronta su quanto fatto e cerca di creare un piano di miglioramento da attuare al prossimo Sprint. In tale Evento, infatti, si ispeziona e si adegua ciò che riguarda il processo e l'ambiente. Ha una durata massima di tre ore ed è sempre lo Scrum Master a vegliare sulla corretta esecuzione; inoltre, è consigliato che ogni Scrum Master vada a moderare retrospettive degli altri Team, permettendo così sia un approccio neutrale sia la condivisione di informazioni tra i vari Team. Viene attuata, dunque, un'analisi interna sia del modus operandi del Team sia delle relazioni interne, andando ad evidenziare cosa è andato bene e cosa andrebbe migliorato (20).

2.4.3. Gli Artefatti

Gli artefatti di Scrum sono i documenti redatti dai vari attori del metodo e hanno il fine di garantire trasparenza e comprensione a tutte le persone coinvolte nel processo di sviluppo. Contengono le informazioni chiave circa il prodotto da realizzare. Essi sono:

Product Backlog

Corrisponde ad un elenco ordinato di requisiti utili alla realizzazione del prodotto. Viene redatto dal Product Owner a seguito dell'interazione con il cliente e poi subisce numerosi aggiornamenti man mano che vengono eseguiti i vari Sprint e in funzione delle modifiche richieste in corso d'opera. Il Product Backlog, infatti, elenca tutte le caratteristiche, le funzioni, i requisiti, le migliorie e le correzioni che costituiscono le modifiche da apportare al prodotto nei futuri rilasci. I suoi elementi hanno i seguenti attributi: descrizione, ordine, stima e valore. Gli elementi ordinati più in alto sono solitamente più chiari e meglio dettagliati rispetto a quelli più in

basso. Stime più precise sono fatte sulla base di maggiori chiarezza e dettaglio; più basso è l'ordine e minore è il dettaglio. Per definire il criterio di prioritizzazione dei requisiti potrebbe essere usata la tecnica MoSCoW che distingue quattro categorie di elementi (21):

1. Must Have: requisito che deve essere presente ai fini della realizzazione;
2. Should Have: un aspetto di alta priorità che dovrebbe essere compreso nella soluzione;
3. Could Have: requisito auspicabile ma non necessario;
4. Won't Have this time: requisite che gli stakeholders hanno deciso di non includere ma che può essere incluso in futuro;

Scrum non fornisce una metodologia di ordinamento ben definita, infatti, il criterio che c'è dietro viene deciso arbitrariamente dal Product Owner. Ad esempio, un secondo criterio può essere l'ordinamento per valore generato dal completamento della singola voce del Product Backlog. Anche qui, però, è fondamentale chiarire sin dalla fase iniziale, cosa si intende per valore generato e, in questo caso, molto spesso viene in aiuto al PO lo Scrum Master.

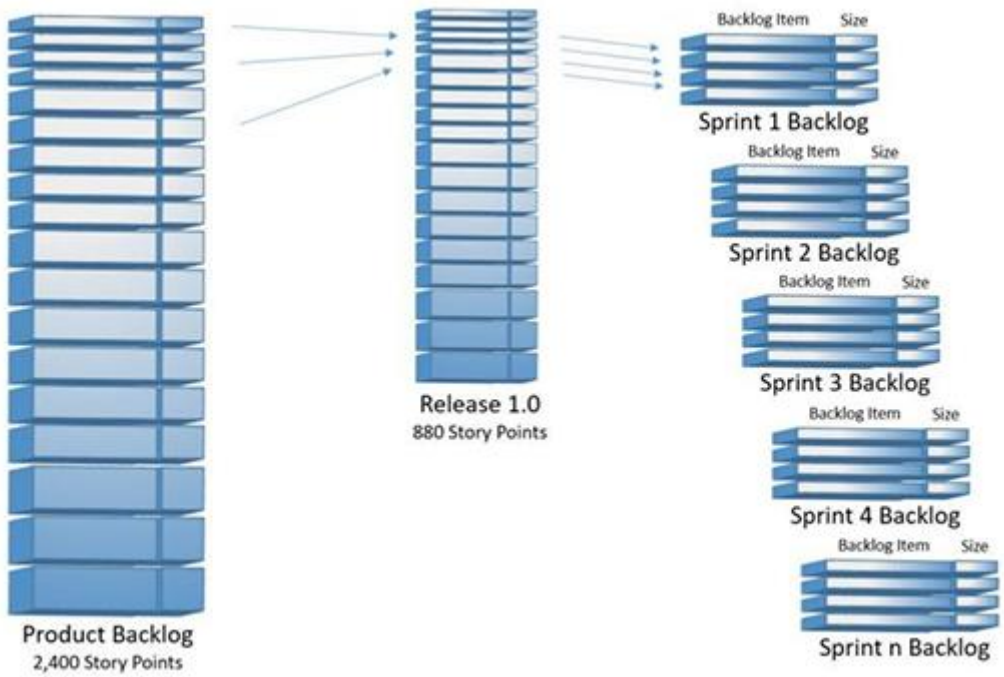

Figura 11 Artefatti (di Judicael Paquet, 2019)

Sprint Backlog

Lo Sprint Backlog è l'insieme degli elementi selezionati del Product Backlog per un determinato Sprint, più un piano d'azione per consegnare l'incremento e raggiungere lo Sprint Goal. Oltre questo, troviamo anche il report del miglioramento che si vuole implementare in questo Sprint a seguito dell'analisi dell'iterazione precedente. Man mano che lo Sprint va avanti, lo Sprint Backlog può essere aggiornato con numerosi dettagli così come alcuni elementi possono essere rimossi perché valutati inutili. È un documento di esclusiva appartenenza del Team di Sviluppo e si basa su una loro previsione del lavoro che andrà eseguito.

Burn Down Chart

A favore di una maggiore trasparenza e facilità di presentazione di quanto fatto nei vari Sprint, è stato ideato il Burn Down Chart, una rappresentazione grafica del lavoro da fare su un progetto nel tempo. Sull'asse verticale è presente il quantitativo di lavoro misurato attraverso punti per storia (simili a delle stime di sforzo in base a complessità e incertezza) o direttamente attraverso le ore, mentre, sull'asse orizzontale troviamo il tempo. Il suo scopo è quello di mostrare al Team l'avanzamento verso il loro obiettivo, non in termini di quantità di tempo che è stato già speso (un fatto irrilevante per l'avanzamento), ma in termini di quantità di lavoro che rimane da fare, ovvero cosa separa il Team dal proprio obiettivo. Se, quasi al termine dello Sprint, il grafico burn down non sta tendendo verso il basso, allora il Team ha bisogno di regolarsi, in modo da ridurre i contenuti del lavoro o di trovare un modo per lavorare in modo più efficace, pur mantenendo un ritmo sostenibile (15).

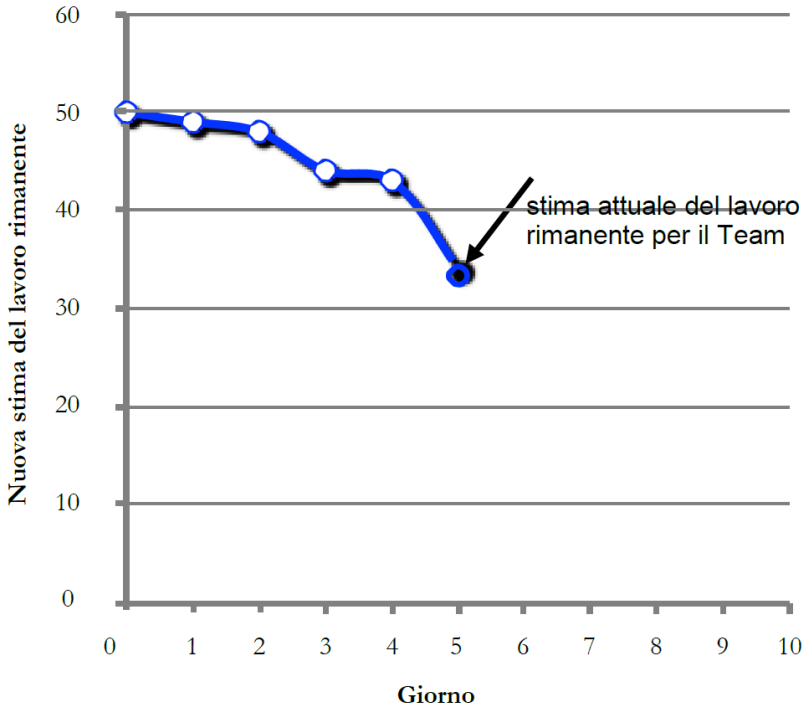

Figura 12- Burndown Chart

2.5. Conclusioni su Scrum

Scrum non è solo un insieme di prassi e regole ma è anche un ambiente di lavoro che fornisce trasparenza e un meccanismo che permette di ispezionare e adeguare. Permette di rendere visibili disfunzioni e impedimenti che stanno intaccando l'efficacia del Team, in modo tale da essere affrontanti e risolti. Soprattutto nelle prime implementazioni di Scrum, il Team, compreso il Product Owner, possono riscontrare numerose difficoltà dettate dal nuovo metodo e dalla difficoltà, ad esempio, di stimare lo sforzo necessario allo sviluppo dell'Incremento. Questo non deve scoraggiare il Team ma, al contrario, deve cogliere le potenzialità del metodo di mettere in luce le debolezze e la capacità di fornire un quadro di riferimento per le persone al fine di esplorare i modi per risolvere i problemi in cicli brevi e con piccoli esperimenti di miglioramento. Un errore comune è quello di modificare alcune procedure chiave del metodo, come ad esempio prolungare le durate degli Sprint, comportando un grave danno al Team, il quale non si porrà mai nelle

condizioni critiche di poter capire come poter migliorare l'utilizzo del tempo. La buona notizia è che a seguito del primo Sprint, il Team riuscirà a cogliere i benefici del metodo già prima della sua fine.

3. PEGA

La piattaforma Low Code Pega ha una presenza mondiale e due terzi dei suoi clienti appartengono ai settori dei servizi finanziari, amministrativi e della sanità. La sua architettura cloud-native, multitenant e basata su microservizi può scalare in alto e in basso, aiutando i clienti ad affrontare i casi d'uso in modo più conveniente e rapido. Un valido esempio è l'app COVID-19 Employee Safety and Business Continuity Tracker precedentemente citata. La piattaforma Pega è stata progettata per automatizzare processi complessi che coinvolgono numerosi utenti aziendali al fine di ottimizzare delle funzioni critiche. Come già detto nel primo capitolo, le piattaforme Low-code permettono ad un largo bacino di utenti di poter sviluppare applicazioni grazie ai vari componenti precedentemente inseriti che possono essere riutilizzati per nuovi casi d'uso. Pega concilia le richieste di numerose aziende circa la possibilità di portare sviluppatori, non sviluppatori e IT insieme all'interno dello stesso ambiente collaborativo; disporre di un'interfaccia grafica intuitiva che non richieda del codice per sviluppare applicativi; permettere ad IT di poter definire dei margini entro i quali i citizen developer possano operare senza compromettere gli standard qualitativi. A questo, aggiunge la presenza di un sistema di gestione agile dei progetti, chiamato Agile Workbench, che fornisce un accesso continuo allo stato degli elementi di lavoro e ne permette la creazione. Ogni elemento di lavoro rappresenta il backlog di ogni compito e Pega permette di inserire tre tipologie di elementi: User Stories (Storie utente) per descrivere i requisiti di business; Bugs (anomalie) per documentare i difetti delle caratteristiche; Elementi di feedback per registrare le richieste di miglioramento identificate durante le sessioni di riproduzione. Questi strumenti risultano oggettivamente utili ai team di sviluppo e alle parti interessate al fine di eseguire i progetti utilizzando la metodologia Scrum. Inoltre, collega senza difficoltà strumenti agili di terze parti per la gestione dei progetti, come Jira e Rally tramite il marketplace (22).

3.1. PEGA E SCRUM: Interfacce e ruoli

Come accennato appena sopra, Pega offre una struttura che si sposa perfettamente con la metodologia Agile e in particolare con il framework Scrum. Il modo in cui Pega cattura e definisce il lavoro attraverso il concetto di *microjourney,* (micro-processi comprendenti varie azioni) rende Scrum il veicolo ideale per realizzarlo in modo trasparente e senza rischi. Questa pratica garantisce la partecipazione sia del business che dell'IT, in modo che ciò che viene configurato soddisfi le aspettative degli stakeholder e raggiunga tutti i risultati aziendali stabiliti all'inizio di un progetto. Il modo in cui Pega utilizza Scrum unifica le best practices di Pega con un approccio di consegna accelerato e collaudato che crea un chiaro obiettivo di prodotto che i team devono perseguire definendo micro-processi e comprendendo come si allineano agli obiettivi dell'organizzazione. Risulta, inoltre, essenziale la corretta definizione degli attori coinvolti in ogni processo. Pega ha identificato alcuni ruoli che potrebbero prendere parte ai processi di sviluppo:

Figura 13 Ruoli Pega **(23)**

- Citizen developers: già citati precedentemente, corrispondono ai non-sviluppatori;
- System Administrator and IT resources: forniscono conoscenze aggiuntive;

- Product Owner: gestisce il backlog del prodotto e la prioritizzazione delle voci del backlog. Crea, inoltre, i criteri di accettazione;
- Scrum master: fa in modo che i membri del team comprendano la teoria e le pratiche di Scrum;
- Business Architect: collaborano con gli esperti di materia e stakeholder per comprendere le esigenze aziendali. In un'applicazione, definiscono le regole aziendali, gli accordi sui livelli di servizio e i processi;
- System Architect: progetta e configura l'applicazione;
- Subject Matter Expert: ha una profonda conoscenza di un particolare argomento o dominio aziendale, collabora con il team di progetto per trasmettere le esigenze aziendali e aiuta a convalidare l'accuratezza delle informazioni;
- Quality Assurance: crea ed esegue script di test funzionali e di performance;
- Project delivery leader: fornisce una guida generale al piano del progetto e alla sua realizzazione;
- Consulting Solutions Executive: fornisce leadership di pensiero, supervisiona la realizzazione e la direzione dei progetti;
- Specialty Architects: partecipa allo sviluppo del progetto in base alle esigenze e all'allineamento delle competenze;

I compiti e le responsabilità per ciascun ruolo possono sovrapporsi o essere svolti da più ruoli (23).

3.2. Confronto con le piattaforme esistenti

Il mercato delle LCDPs è molto ricco di aziende, ognuna mette a disposizione dei clienti una piattaforma propria con una svariata gamma di strumenti low-code per lo sviluppo e composizione di applicazioni. Data la diversità delle soluzioni, i clienti dovrebbero mappare tutte le capacità di ogni piattaforma e assicurarsi di selezionare la soluzione che soddisfi le necessità degli sviluppatori e i requisiti dei clienti. Per questo motivo, Gartner ha deciso di effettuare una ricerca di mercato atta a valutare le principali 12 piattaforme, presenti sul mercato, per aiutare i

software developers nella loro scelta. Le aziende sono state valutate in base a otto principali capacità:

- Ciclo di vita dello sviluppo software;
- Design della User Experience;
- Produttività dello sviluppo;
- Business Workflow;
- Integrazione e APIs;
- Ecosistema della piattaforma;
- Governance;
- Sicurezza e Qualità del servizio;

Queste capacità sono state valutate in tre principali casi d'uso:

1. Custom business applications: creazione di applicazioni aziendali che richiedono esperienze utente avanzate, integrazioni complesse, monitoraggio e gestione di grandi volumi di transazioni;
2. Business workflow automation: automatizzare flusso di lavoro che coinvolgono più sistemi applicativi e attori per raggiungere gli obiettivi aziendali;
3. Collaborative app development: permettere a differenti tipologie di sviluppatori (citizen o prefessional) o a team di sviluppare applicazioni collaborando.

Di seguito sono riportati i risultati delle 12 piattaforme nei 3 casi d'uso:

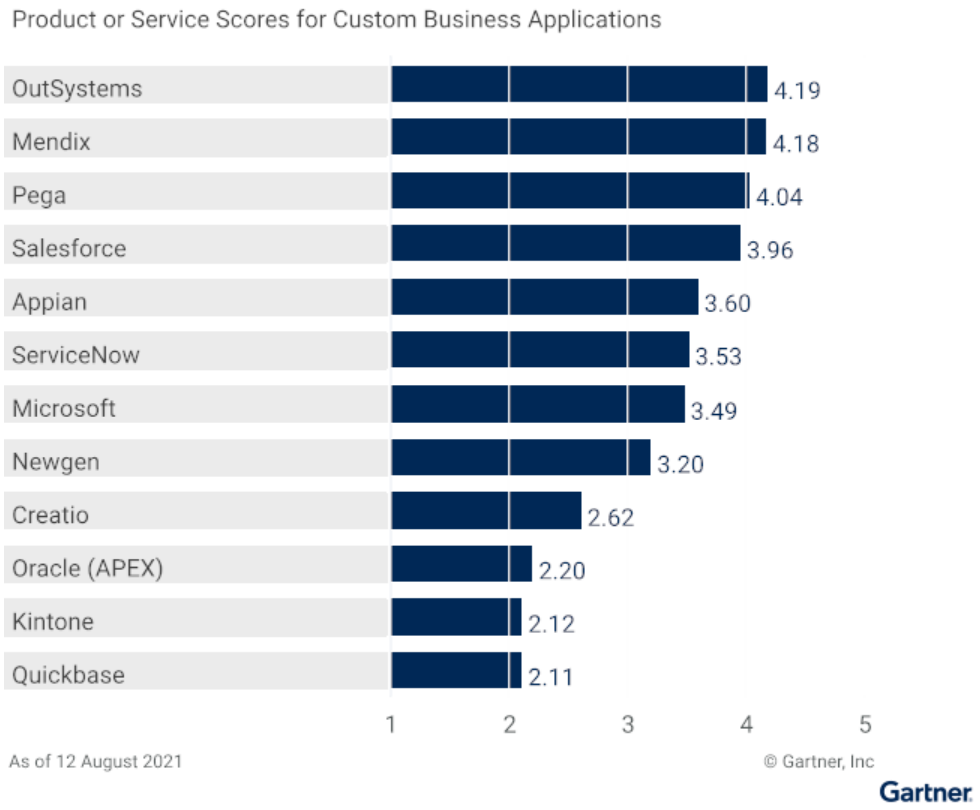

Figura 14 Customer Business Applications

In questo primo caso d'uso, Pega si colloca nei primi tre, con un punteggio di 4.04, confermandosi capace di dare la possibilità allo sviluppatore di realizzare applicativi complessi con un elevato numero di requisiti. In questa analisi, infatti, il punteggio 4 corrisponde a Eccellente, cioè incontra o eccede i requisiti richiesti dal caso. Inoltre, come già detto nel primo capitolo, Pega dimostra con questo risultato di non essere soggetta a problemi di scalabilità.

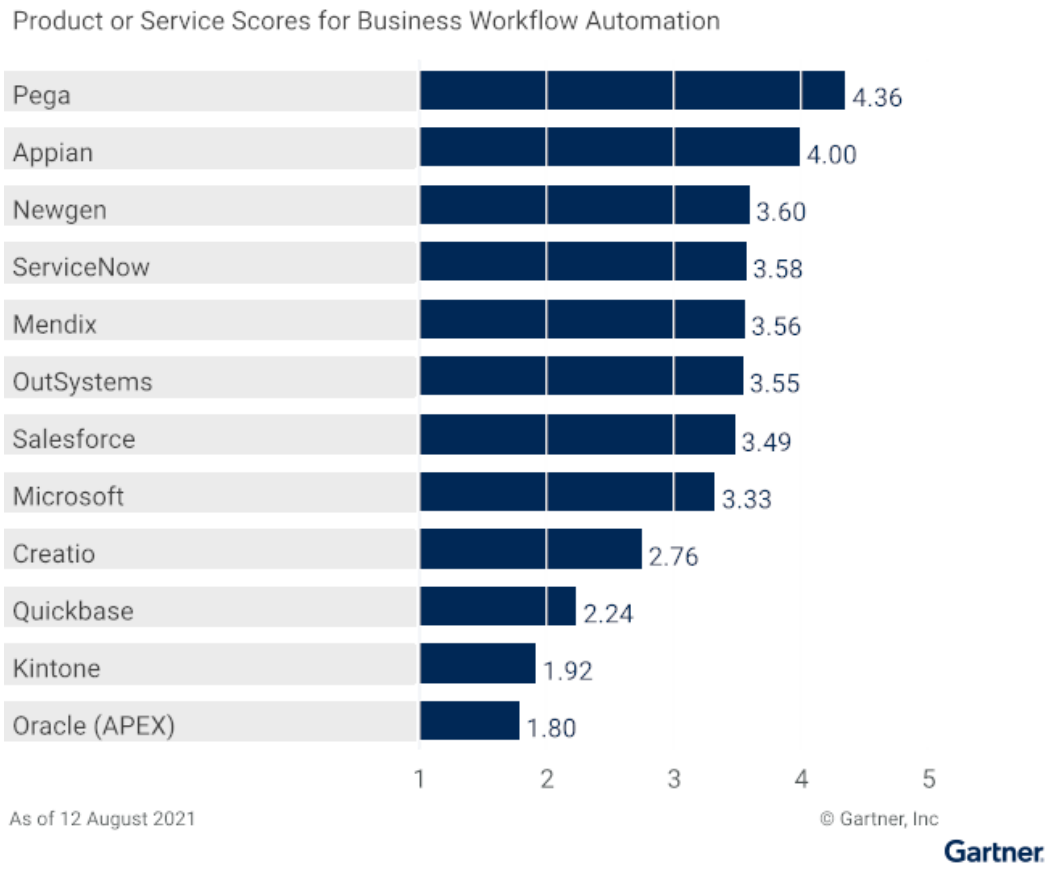

Figura 15 Business Workflow Automation

Nel secondo caso, Pega si colloca prima, facendo dell'automazione dei flussi di lavoro aziendali il suo cavallo di battaglia. Eccelle, dunque, nella gestione di processi complessi offrendo workflow semplici, processi orientati ai documenti, gestione dei task e gestione di casi che richiedono capacità di modellazione dei processi, integrazione e modellazione delle decisioni (24). A confermare l'analisi di Gartner, un'altra società di spicco (Forrester) ha condotto una seconda analisi sempre riguardante le migliori piattaforme per l'automazione dei processi di business. Da essa, Pega è risultata come principale piattaforma adatta a tale scopo, ricoprendo il ruolo di Leader. (25)

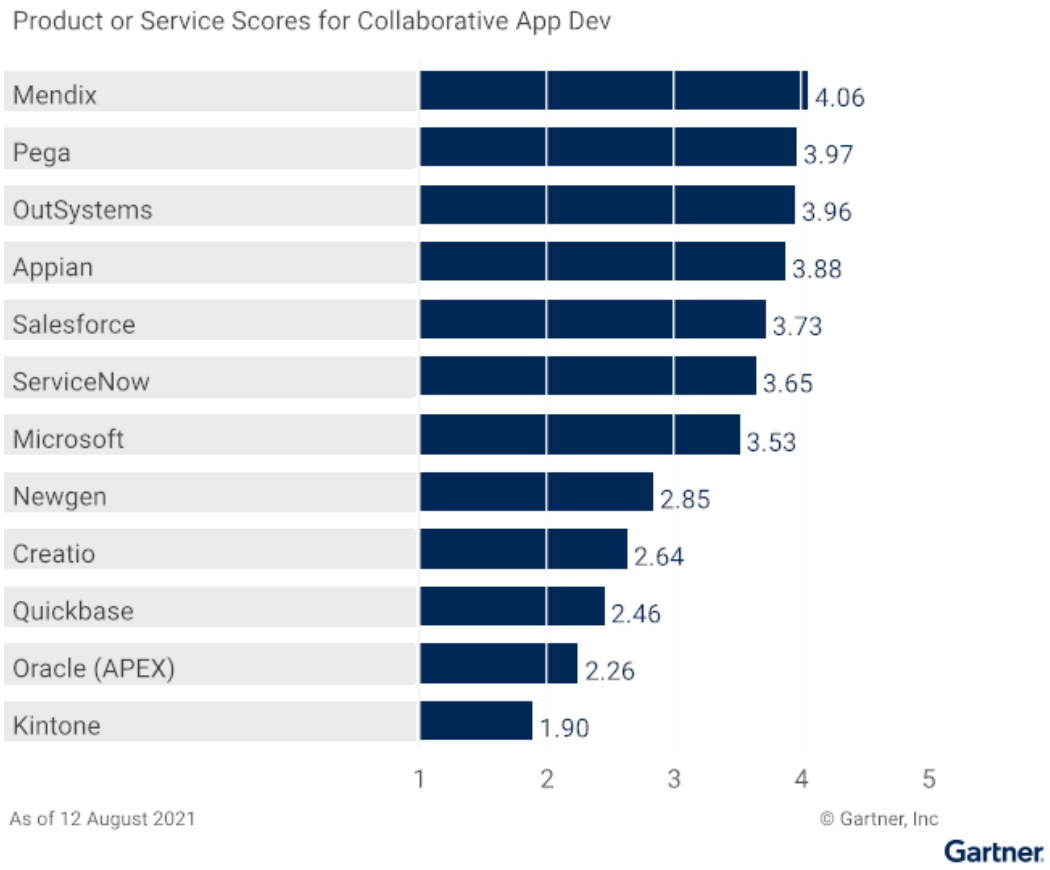

Figura 16 Collaborative App Dev

Nel terzo e ultimo caso, Pega si colloca seconda e dimostra la sua attitudine alla collaborazione di più attori diversi tra loro, offrendo un ecosistema solido ed efficiente in termini di produttività. Offre, infatti, lo sviluppo drag-and-drop, guide per gli sviluppatori, un'integrazione semplificata e una solida governance dello sviluppo delle applicazioni (24).

3.3. Magic Quadrant Gartner per CRM

La pandemia ha richiesto un'azione immediata da parte delle organizzazioni per stabilizzarsi, riprendersi e prepararsi a una nuova crescita. Molti fornitori di tecnologia CEC (customer engagement center), compresi quelli valutati in questo Magic Quadrant, hanno fornito un aiuto vitale ai loro clienti in termini di navigazione nell'interruzione e nell'incertezza. Gartner definisce il mercato del CRM come il mercato delle applicazioni software usate per fornire servizio e supporto ai clienti (CSS) impegnandosi in modo intelligente - sia proattivamente che reattivamente - con i clienti rispondendo alle domande, risolvendo i problemi e dando consigli. In questo rapporto, Gartner analizza i fornitori CEC che hanno concesso l'opportunità di rimodellare il futuro del lavoro all'interno delle aziende, al fine di migliorare la resilienza e di favorire il raggiungimento delle prestazioni ottimali. Sono stati selezionati 15 fornitori in base a numerosi fattori riguardanti sia la notorietà, sia il momento di crescita sia il set di servizi offerti al cliente. Tra i principali requisiti di maggior rilievo in termini di peso, troviamo:

- Prodotto o servizio: Beni e servizi principali offerti dal fornitore per il mercato definito. Questo include le attuali capacità del prodotto e del servizio, la qualità, i vari set di caratteristiche, le competenze, se offerti nativamente o attraverso accordi OEM e partnership;
- Modalità di vendita/Prezzi: Le capacità del fornitore in tutte le attività di prevendita e la struttura che le supporta. Questo include la gestione degli affari, la determinazione dei prezzi e la negoziazione, il supporto prevendita e l'efficacia generale del canale di vendita;
- Reattività/record del mercato: La capacità di rispondere, cambiare direzione, essere flessibile e raggiungere il successo competitivo quando le opportunità si sviluppano, le esigenze dei clienti si evolvono e le dinamiche del mercato cambiano. Questo criterio considera anche la storia di reattività del fornitore.
- Esperienza del cliente: Relazioni, prodotti, servizi e programmi che permettono ai clienti di avere successo con il prodotto valutato. In

particolare, questo include i modi in cui i clienti ricevono il supporto tecnico o il supporto dell'account. Può anche includere strumenti ausiliari, programmi di supporto ai clienti (e la loro qualità), disponibilità di gruppi di utenti, accordi sul livello di servizio e così via.

- Comprensione del mercato: L'abilità del venditore di capire i desideri e i bisogni degli acquirenti e di tradurre tale comprensione in prodotti e servizi.

Questi parametri sono stati fondamentali per poter collocare ogni fornitore all'interno di uno dei quattro quadranti identificati da Gartner, come riportato in figura (26):

Figura 17 Magic Quadrant CRM

Pegasystems è un leader in questo Magic Quadrant. Offre capacità di coinvolgimento dei clienti e automazione del flusso di lavoro all'interno della sua soluzione Pega Customer Service. Le organizzazioni dovrebbero prendere in

considerazione Pega Customer Service quando hanno bisogno di apportare frequenti modifiche a processi e percorsi di assistenza clienti altamente innovativi e proattivi. I punti di forza evidenziati dal report sono:

- Partner di trasformazione digitale: Pegasystems è cresciuta costantemente come azienda pubblica per molti anni e ha un solido record di implementazioni d'impatto. Gli utenti, in particolare le grandi imprese, hanno lodato Pegasystems per il suo coinvolgimento come partner nelle loro iniziative di trasformazione digitale supportate dalle tecnologie CEC;
- Focus sull'integrazione: L'approccio di Pegasystems all'innovazione, supportato dall'approccio Low-Code, rende relativamente facile estendere le proprie soluzioni incorporando tecnologie di terze parti, touchpoint del cliente, flussi di lavoro, AI e RPA per offrire customer experience journey differenziati ed end-to-end;
- Focus sull'automazione: Pegasystems si concentra sull'automazione dell'adempimento delle richieste di servizio e sul miglioramento della soddisfazione del cliente e dell'esperienza dell'agente. Questo è dimostrato, per esempio, dalla sua introduzione di una funzione RPA "start my day", che rende più facile per i consulenti prepararsi per la prima interazione con il cliente della giornata. In generale, l'introduzione di funzionalità Robot Process Automation ha portato un grosso valore aggiunto all'azienda, collocandola tra i Visionari nel Magic Quadrant riguardante proprio le principali aziende che offrono servizi RPA nativi. Il prodotto di Pegasystems è dotato di una tecnologia proprietaria di visione artificiale nota come X-ray Vision: riduce il carico di lavoro iniziale, utilizzando l'IA e il machine learning per l'autoapprendimento per correggere le automazioni man mano che vengono aggiornate le applicazioni (27). La piattaforma Pega Infinity offre, dunque, funzionalità premium, ma a un prezzo premium. Sebbene Pegasystems abbia introdotto diversi aggiornamenti al suo modello di prezzo, continua a rivolgersi alle grandi imprese con budget elevati.

Cosa, invece, lamentano gli analisti riguarda:

- Offerta di prodotti intricata: Gli utenti del servizio di richiesta clienti di Gartner e i potenziali clienti di Pegasystems, con cui Gartner ha avuto contatti, hanno detto di aver avuto difficoltà a capire la piattaforma di Pegasystems, e che questa comprensione era cruciale per identificare il tipo di implementazione più adatto. Inoltre, i clienti di Gartner hanno indicato che le implementazioni iniziali possono essere complesse.
- Accessibilità degli specialisti: Alcuni clienti e potenziali clienti di Pegasystems hanno espresso frustrazione per la difficoltà di trovare specialisti Pegasystems con esperienza tecnica e di settore rilevante. Le organizzazioni che stanno considerando Pegasystems dovrebbero assicurarsi di avere una strategia appropriata per mitigare i potenziali rischi di carenze di personale e competenze. Proprio a tal proposito, Pega ha introdotto negli ultimi anni numerosi percorsi formativi all'interno della propria Academy online (26).

3.4. L'Impatto economico di PEGA

Pega ha commissionato l'azienda Forrester Consulting di condurre uno studio circa l'impatto economico totale in merito all'utilizzo di questa piattaforma per sviluppare applicativi personalizzati e il calcolo del relativo Return on Investment. Per far ciò, ha intervistato numerosi clienti, tra cui cinque aziende, che utilizzano la piattaforma Pega sfruttando il loro framework TEI (Total Economic Impact). L'obiettivo è stato di identificare i costi, i benefici, la flessibilità e i fattori di rischio che influenzano la decisione di investimento. Prima di utilizzare Pega, le aziende intervistate disponevano di svariati sistemi per supportare i processi aziendali che richiedevano dell'hard-coding per sopperire a problemi o solo per aggiornare qualche componente. Spesso queste modifiche venivano realizzate da società di terze parti che richiedevano consistenti somme di denaro per ogni modifica

richiesta. Inoltre, risultava difficile riuscire a modificare il codice di base perché realizzato in maniera contorta o comprendibile solo dall'azienda produttrice. Questo porta a limitare molto la flessibilità del servizio e alla perdita del controllo da parte dell'azienda del componente. Post implementazione Pega, le varie organizzazioni hanno da subito riscontrato numerosi benefici:

- Risparmi di costi grazie alla redistribuzione del lavoro interna ripartito tra IT e citizen developers, riducendo, inoltre, il numero di consulenti coinvolti nei vari progetti;
- Riduzione dei tempi dei cicli di sviluppo e maggiori progetti completati con conseguente aumento della produttività degli utenti business;
- Risparmio dallo smantellamento delle vecchie piattaforme in uso prima dell'adozione di Pega;
- Aumento della produttività degli utenti finali includendo funzionalità aggiuntive di Pega alle applicazioni sviluppate. In particolare, l'implementazione dell'RPA, ha fornito la possibilità di sviluppare bot per supportare le funzioni di vendita.
- Miglioramento dei rapporti tra area IT e Business dell'azienda che adesso riescono a comunicare in maniera più efficace, fornendo numerosi spunti per il miglioramento dei processi.

Forrester è riuscita a quantificare gli effettivi guadagni in termini di risparmio, di Net Present Value (Valore attuale netto), Payback e ROI, sulla base delle informazioni raccolte delle interviste su una base temporale di tre mesi:

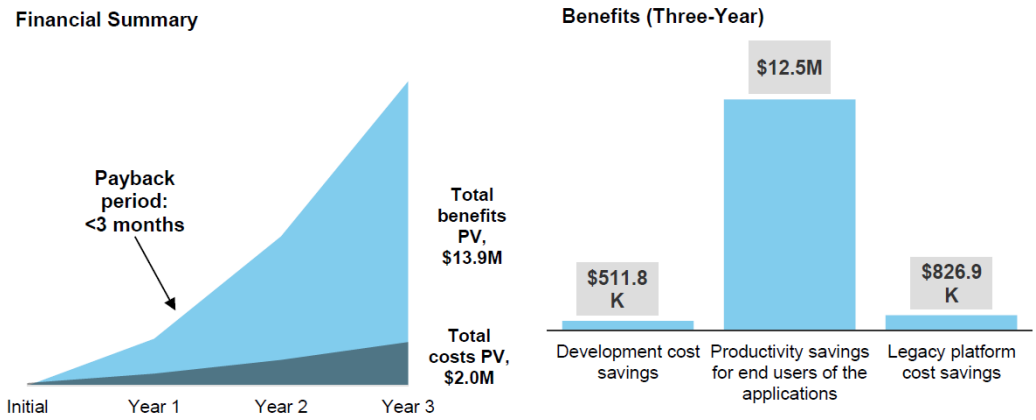

Grafico 3 Payback e Risparmi **(22)**.

Cash Flow Analysis (risk-adjusted estimates)

	INITIAL	YEAR 1	YEAR 2	YEAR 3	TOTAL	PRESENT VALUE
Total costs	($117,490)	($473,400)	($754,521)	($1,084,113)	($2,429,524)	($1,985,935)
Total benefits	$0	$2,349,050	$5,654,167	$9,382,450	$17,385,667	$13,857,539
Net benefits	($117,490)	$1,875,650	$4,899,646	$8,298,337	$14,956,143	$11,871,604
ROI						598%
Payback period (months)						<3

Tabella 4 Cash Flow Analysis **(22)**

I benefici dell'adozione di Pega sono evidenti. Nella media, le aziende intervistate hanno riscontrato un risparmio sulla spesa riguardante i servizi di consulenza del ben 60% in un arco temporale di tre anni. La produttività generata è stata quantificata e stimata pari a 12,5 milioni di dollari con un totale di 20 applicazioni sviluppate che supportano 3.350 utenti. Tale aumento deriva dal reindirizzamento delle ore di lavoro lontano da processi laboriosi che adesso risultano automatizzati. A ciò si aggiungono il risparmio delle spese d'uso delle vecchie piattaforme. Forrester ha anche aggiunto le spese che la piattaforma Pega richiede che riguardano: implementazione, gestione, formazione e costi di licenza. Il NPV illustra come al di là dei costi, i benefit sono fortemente superiori ad essi, il payback è addirittura inferiore ai tre anni e sono riusciti ad ottenere un ROI del 598%. Il

grafico sottostante, oltre a segnalare benefit e costi, mostra il trend del beneficio netto cumulato che sembra destinato ad aumentare ancora nel tempo.

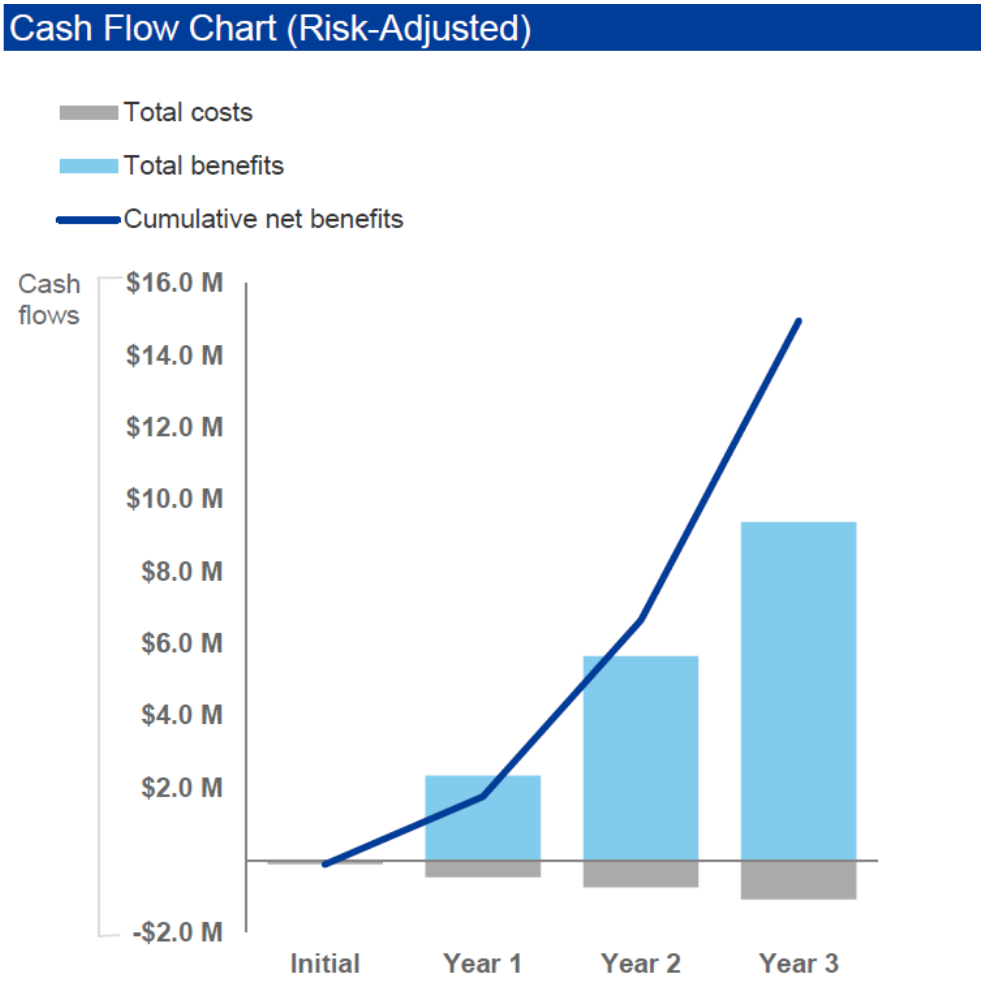

Grafico 4 Cashflow Chart **(22)**

Oltre alla comparazione tra benefici e costi, come detto all'inizio di questo paragrafo, si è analizzata la Flessibilità della piattaforma, intesa come il valore strategico che può essere ottenuto in futuro grazie all'investimento fatto. Man mano che aumenta il livello di integrazione della piattaforma nel contesto aziendale e il numero di persone coinvolte di varia tipologia nel processo di sviluppo, si stima una progressiva accelerazione dei tempi di consegna in merito ai continui miglioramenti alle applicazioni esistenti. Inoltre, grazie al continuo utilizzo della piattaforma, si genereranno sempre più componenti che entrano a far parte del parco

strumenti messi a disposizione di ogni sviluppatore. Questo porta ad un progressivo incremento della velocità di sviluppo e, allo stesso tempo, a ridurre il numero di persone coinvolte nello sviluppo di una singola applicazione (22).

3.5. Servizi aggiuntivi PEGA

Oltre i principali servizi precedentemente citati, Pega offre una varietà di strumenti aggiuntivi, definibili "premium", che offrono un valore aggiunto al cliente. Tra le varie licenze offerte, verranno analizzati brevemente due strumenti in particolare: SFA e CDH.

3.5.1. SFA

L'acronimo SFA sta per Sales Force Automation e consiste nell'automazione dei processi di vendita in tutti e tre i più comuni casi di vendita: B2B, B2C e Vendite indirette. L'automazione nasce grazie ad una piattaforma in grado di raccogliere al suo interno tutti i vari step del processo di vendita, partendo dal rapporto iniziale con il cliente e terminando con l'evasione dell'ordine. Questa raccolta di informazioni dona una particolare flessibilità e permette all'azienda di poter adeguare le proprie proposte, in termini di timing, offerte, ecc., in base al profilo del cliente. In aggiunta, Pega integra a questo strumento i vari processi di marketing, di servizio clienti e logistici, al fine di creare un ecosistema sempre più produttivo ed efficiente (28). Gartner ha eseguito un'analisi di mercato in merito a questa tematica e ha inizialmente collocato Pega tra i Visionari per poi attribuirle, in un secondo report, punteggi superiore al 4 in tutti e tre i casi d'uso sopra citati, posizionandola prima in tutte e tre le classifiche che mettevano a confronto le varie società. Questo perché nell'ultimo anno ha incrementato e integrato nuove capacità specialmente nel mondo delle vendite B2B. Lato B2C, l'app mobile offerta dal servizio permette di tenere traccia di tutti i processi di vendita, integrando dashboard, servizi di geolocalizzazione e calendarizzazione. Per quanto concerne

le vendite indirette, ne permette l'archiviazione, evitando potenziali conflitti interni e fornisce degli insights utili al monitoraggio in tempo reale di tali contratti di vendita. Cosa lamentano gli utenti è la complessità dell'infrastruttura e, come già detto precedentemente, il principale deficit è la mancanza di risorse specializzata nell'integrazione del servizio (29).

3.5.2. CDH

Sempre più aziende e in particolare le corrispettive aree marketing hanno capito come spesso le classiche campagne marketing risultano poco impattanti perché destinate a target troppo ampi. Inoltre, una singola azienda dispone di vari mezzi di comunicazione che alle volte possono entrare in conflitto tra loro a causa di mancanza di comunicazione interna. Si è giunti, dunque, alla conclusione che quello che impatta veramente è il contenuto e il timing dell'offerta personalizzata per ogni cliente, garantendo così una migliore customer experience dello stesso. A tal proposito, alcune aziende hanno deciso di affidarsi al servizio CDH di Pega (30). Pega Customer Decision Hub è adatto ad aziende molto grandi che devono elaborare elevati volumi di richieste dettagliate dai clienti. Il suo approccio è basato su modelli ed è progettato per servire settori con frequenti cambiamenti nei processi ad alta complessità, come la sanità, le assicurazioni, e i servizi finanziari. Consiste nel raggruppare tutte le informazioni del cliente all'interno dello stesso spazio, utilizzando un approccio che mette il cliente al centro della strategia, allontanandosi dalla classica ideologia che pone l'attenzione al solo prodotto, lasciandosi guidare dai gusti e necessità del cliente (31). Il vero potere del CDH deriva dalla possibilità di elaborare i big data dei clienti generando dei modelli di machine learning capaci di prevedere le necessità di ogni cliente. Grazie anche alla presenza dell'AI, propone intelligentemente offerte e soluzioni ai clienti personalizzate capaci di incrementare le vendite dell'azienda. A testimonianza di ciò, Forrester ha eseguito un'analisi economica intervistando quattro principali clienti di Pega, utilizzanti il CDH, e ha riscontrato i seguenti dati:

Cash Flow Table (Risk-Adjusted)

	INITIAL	YEAR 1	YEAR 2	YEAR 3	TOTAL	PRESENT VALUE
Total costs	($5,913,380)	($13,563,960)	($7,665,160)	($8,045,160)	($35,187,660)	($30,623,543)
Total benefits	$0	$46,440,000	$74,493,000	$101,926,238	$222,859,238	$180,361,336
Net benefits	($5,913,380)	$32,876,040	$66,827,840	$93,881,078	$187,671,578	$149,737,793
ROI						489%
Payback period						<6 months

Tabella 5 Cash Flow table CDH **(30)**

Le interviste di Forrester ai quattro clienti esistenti e la successiva analisi finanziaria hanno rilevato che un'azienda ha ottenuto benefici di 180 milioni di dollari in tre anni a fronte di costi pari a 31 milioni di dollari, raggiungendo un valore attuale netto (NPV) di quasi 150 milioni di dollari e un ROI del 489%. Al di là dei costi di molto maggiori rispetto ai costi per la piattaforma standard, risulta netto il guadagno ottenuto dell'implementazione di questo strumento.

4. PROGETTO

Al fine di fornire una panoramica completa delle piattaforme Low-Code, seguirà la descrizione del progetto svolto in concomitanza della tesi per poter fornire un'esperienza di sviluppo in prima persona, dopo che il tesista ha seguito un percorso di formazione di 50 ore sulla piattaforma di Accademy offerta dall'azienda PegaSystem.

L'azienda Blue Reply S.R.L. consente ai propri dipendenti di personalizzare la propria postazione di lavoro e di favorire il setup tecnologico per i nuovi ingressi. Ciò avviene per mezzo di richieste di acquisto (RdA) tramite cui qualsiasi dipendente può richiedere al dipartimento competente l'acquisto di dispositivi elettronici e periferiche utili alle proprie attività lavorative. Durante la finalizzazione dell'ordine, ciascuna RdA viene sottoposta ad un processo di approvazione da parte dei Manager e dei Partner dell'azienda. A tal fine, si richiede la realizzazione di un portale per la sottomissione di RdA, al fine di agevolare l'inserimento dei campi necessari per elaborare la richiesta. L'interfaccia grafica sarà composta da componenti quali filtri e campi di input, con dipendenze tra input ed interfacciamento a database per il recupero di campi dinamici. Il portale dovrà essere realizzato tramite tecnologia low code attraverso la piattaforma Pega. Il progetto verrà condotto in modalità Agile secondo il framework SCRUM, descritto nel capitolo due di questa tesi. Per l'organizzazione degli Sprint, si è ricorsi al software Jira, uno dei principali utilizzati in ambito Agile.

4.1. Fase di Pianificazione

In fase di pianificazione, il Product Owner ha raccolto le informazioni utili alla definizione dei requisiti di progetto che sono state successivamente tradotte in voci prioritizzate nel backlog, in funzione del valore di business, sotto forma di User Stories e caricate sul software Jira, nell'apposita sezione. Inizialmente, sono stati schedulati tre sprint, da una settimana ciascuno, ove all'inizio di ciascuno, è stata

prevista una Sprint Planning mentre, al termine di ciascuno, in linea con le regole Scrum, sono state previste una Sprint Review e una Sprint Retrospective. Nella versione di Jira utilizzata, le User Stories all'interno di ogni sprint potevano essere esplose in micro-attività e ciascuna User Story poteva essere inserita all'interno di quattro diversi riquadri, ognuno rappresentante un diverso stato, quali: "da completare", "in corso", "in test" e "completato". Questo è possibile grazie all'interfaccia grafica presente all'interno della pagina Board, sotto riportata:

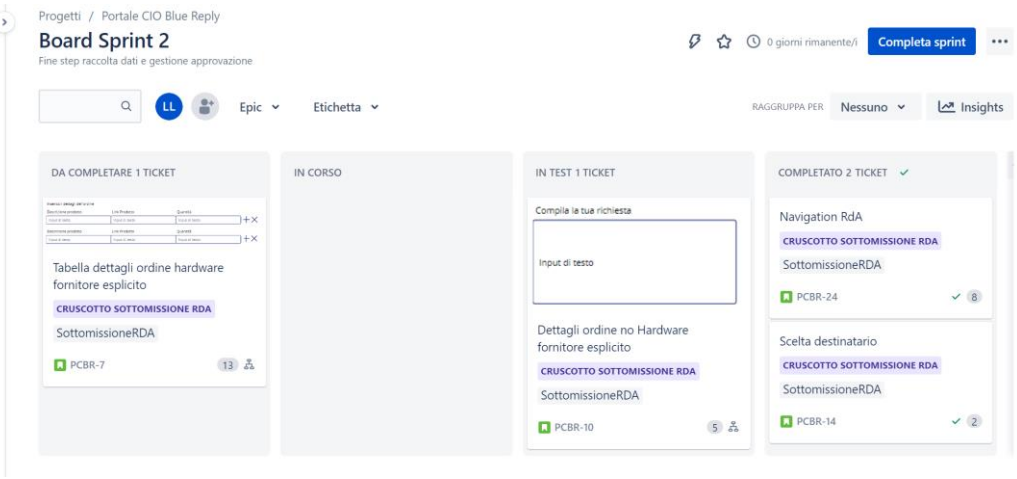

Figura 18 Jira

Fatto ciò, si è svolta la prima Sprint Planning dove sono state selezionate alcune voci del backlog in base alla prioritizzazione e al numero totale di punti storia ipoteticamente raggiungibili dal team.

4.2. Primo Sprint

Il primo Sprint aveva l'obiettivo di creare il macro-processo di raccolta dati per quanto concerne l'inserimento della generica richiesta. Sono state selezionate inizialmente quattro User Stories: PCBR 2, 3, 4, 5 al quale poi sono state aggiunte durante lo stesso sprint le Stories 13 e 6.

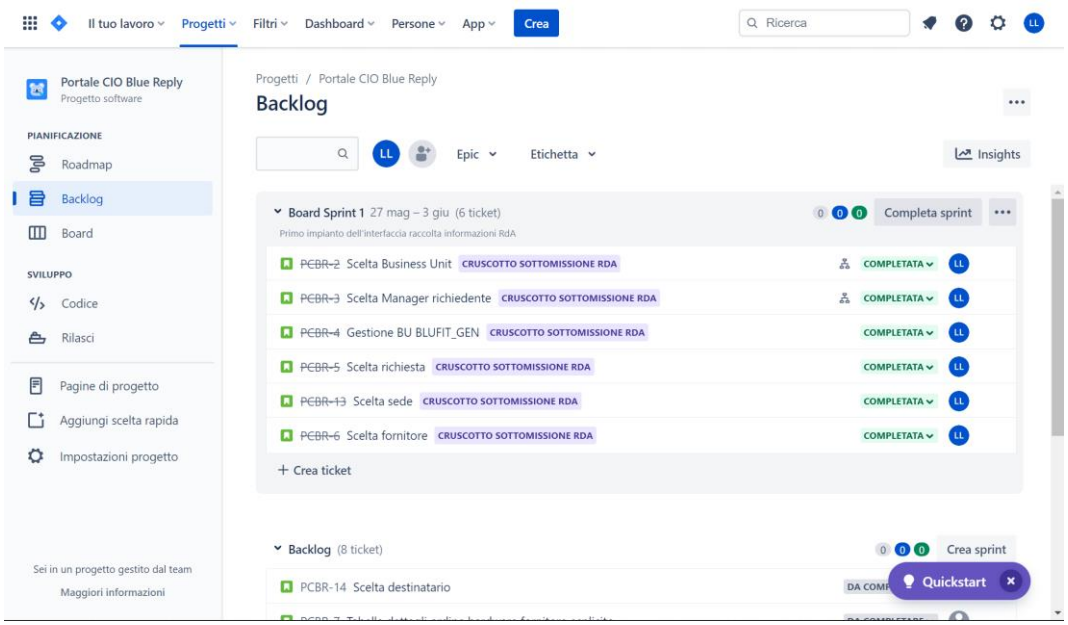

Figura 19 Primo Sprint

La piattaforma organizza i flussi in *microjourney* che consistono in una piccola parte del percorso complessivo del cliente e si concentra sul raggiungimento di un obiettivo specifico. A tale scopo, viene creato il Case Type che contiene una parte del macro-processo generale ed è caratterizzato da una gerarchizzazione delle attività. Troviamo, infatti, lo Stage al livello più elevato che specifica un preciso stato del flusso; all'interno del singolo stage, sono presenti i processi che raccolgono i vari step da seguire al fine del raggiungimento dell'obiettivo finale.

All'interno del primo Stage, sono stati realizzati tre step, contenuti all'interno dello stesso processo ed è stata definita la "Persona" che andrà ad interagire nel processo attraverso l'User Portal.

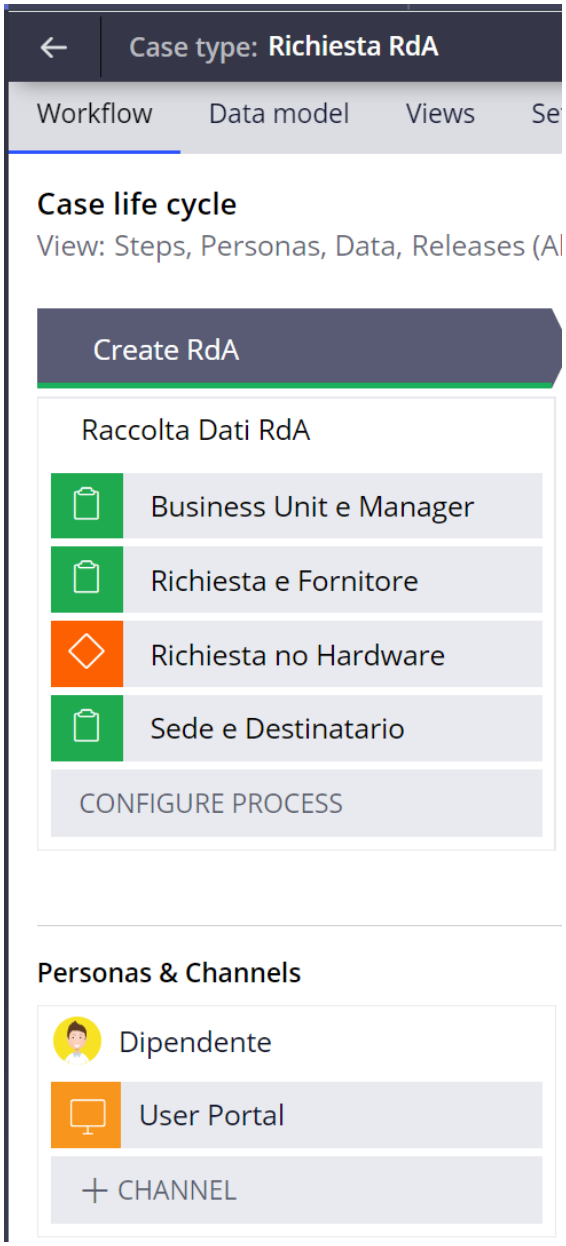

Figura 20 Primo Stage

Il primo step si preoccupava di raccogliere l'informazione circa la scelta della Business Unit e del Manager richiedente tramite due picklist. Per far ciò, è stato creato il data object "Business Unit" contenente il solo campo testuale Nome ed è stato popolato con i nomi delle varie BU (selezione Records).

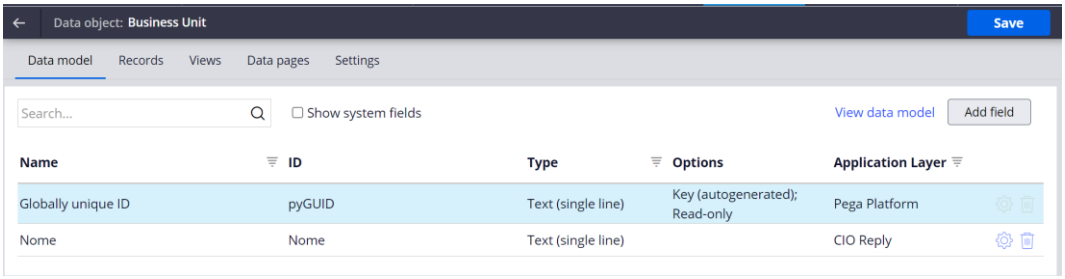

Figura 21 Business Unit Object

Scelta la Business Unit, il progetto prevede di selezionare il Manager richiedente in funzione della BU selezionata allo step precedente. Per tale motivo, è stato creato il data object "Manager" avente i campi: Full name (nome completo) e il campo relativo alla business unit di appartenenza. Successivamente è stata creata una Lista Manager, in forma di Page List, in modo tale da applicare una "data transform" (trasformazione dei dati) alla lista completa al fine di filtrarla solo con i manager appartenenti alla BU selezionata.

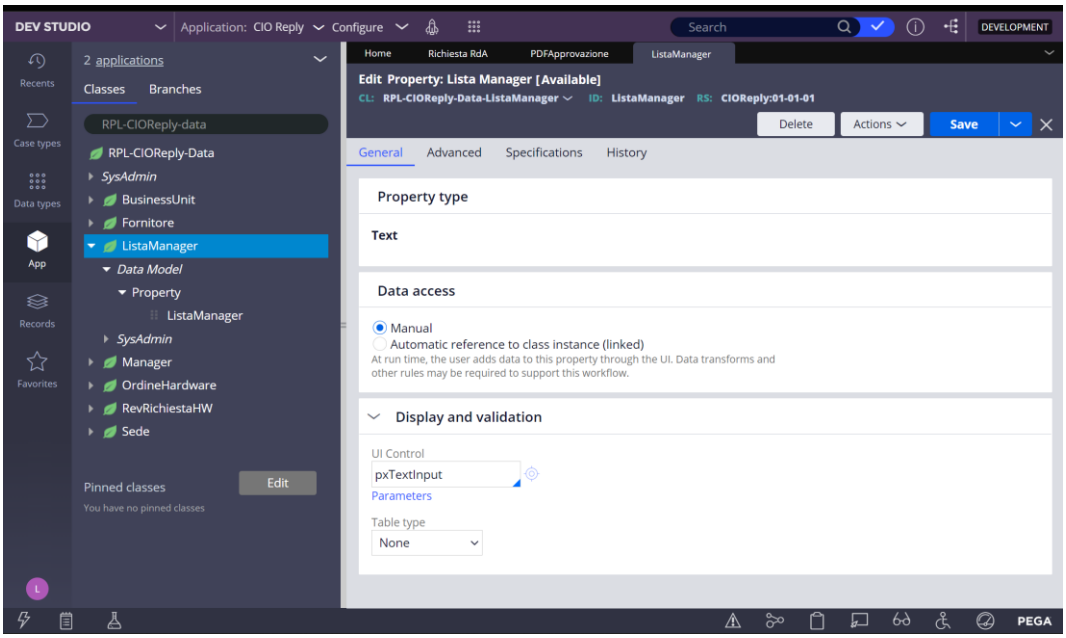

Figura 22 Lista Manager

La data transform è una funzionalità capace di elaborare dati in input e di fornire in output qualsiasi tipologia di dato con le modifiche desiderate. In questo caso, dopo

un reset della lista, è stato inserito un "*for each*" al fine di ciclare tutti gli elementi della lista Manager e selezionare, tramite una condizione di *when*, gli elementi aventi la BU selezionata dall'utente. Le corrispondenze vengono inserite all'interno della lista precedentemente creata per poterla visualizzare nella picklist presente lato interfaccia utente.

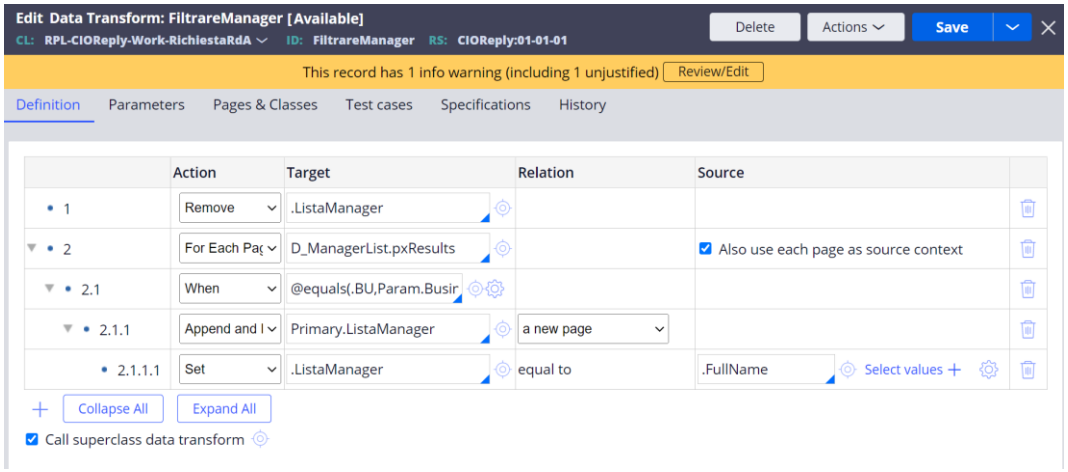

Figura 23 Data Transform

La data transform è stata applicata all'interno della view dello step "Business Unit e Manager":

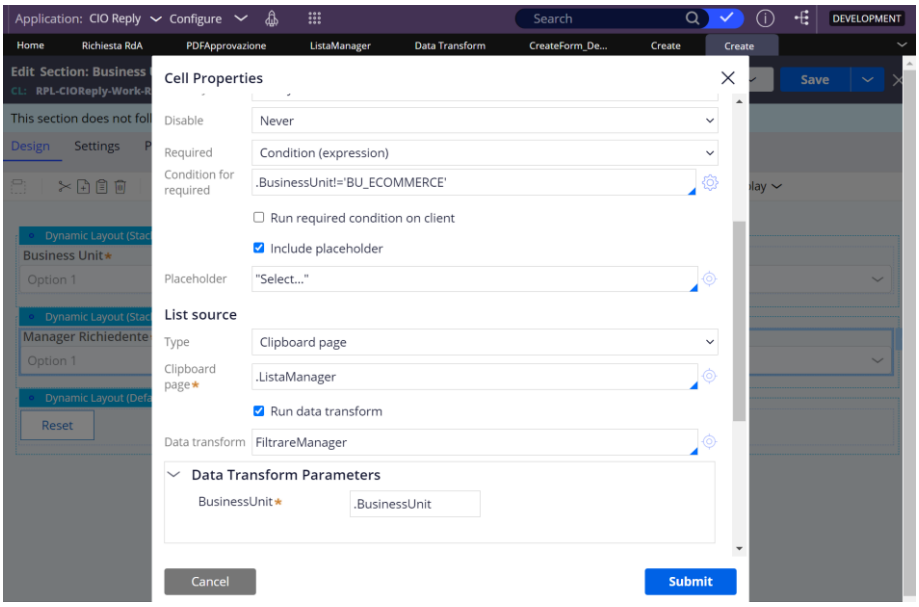

Figura 24 Inserimento Data Transform

Poi, è stato gestito un caso particolare che ha permesso lo skip dello step di selezione del manager andando a modificare l'opzione di visibilità della picklist. Il secondo e terzo Step inseriti, "Richiesta e fornitore" e "Sede e destinatario", sono stati modulati allo stesso modo dello step precedente. Poiché lo Step di scelta sede e destinatario è previsto solo nel caso in cui la tipologia scelta è di tipo "Hardware", è stato inserito uno step "Decision" (rombo arancione) che controlla la tipologia di richiesta e in base a ciò veicola il processo. Durante la Sprint Review sono emersi alcuni problemi in termini di interfaccia grafica e visualizzazione dei vari form. Per questo motivo è stata creata una nuova User Story con il fine di riorganizzare l'interfaccia e la successione di raccolta dati della richiesta. Durante, invece, la Retrospective sono stati rivisti i punti storia assegnati alle varie User Stories nel corso del primo sprint, al fine di pianificare in maniera ottimizzata il prossimo Sprint. I punti storia non sono stati assegnati fin da subito per mancanza di esperienza nello sviluppo e dunque per l'impossibilità di tarare nella giusta maniera le varie attività. Si è deciso, infatti, di identificare una User Story di riferimento da un'unità e da lì assegnare man mano i vari punteggi.

4.3. Secondo Sprint

Come anticipato, durante la Sprint Planning del secondo sprint, è stata aggiunta la nuova user story (PCBR-24) che va a modificare il processo di raccolta dati. A questa user story, dopo aver assegnato i vari punti storia a ciascuna, sono state aggiunte le PCBR 14, 7, 10.

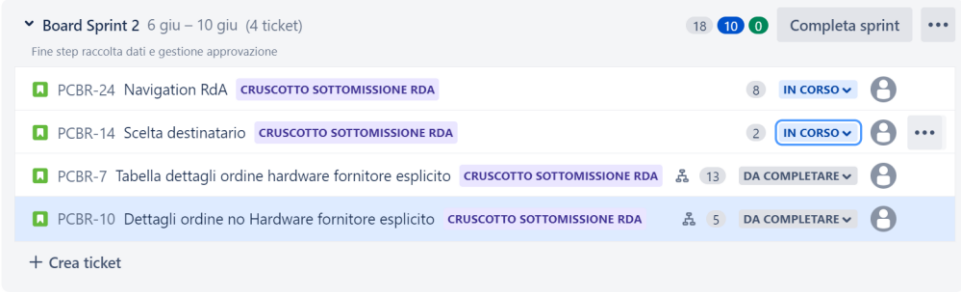

Figura 25 Sprint 2 Backlog

Lo scopo di questo sprint è stato di permettere al dipendente di inserire, in caso di prodotto hardware, il destinatario attraverso un inserimento testuale. A seguito di ciò, il progetto prevedeva una biforcazione in funzione della tipologia di richiesta selezionata e a tal proposito si è deciso di inserire un nuovo Stage al cui inizio è stato inserito un "decision" al fine di filtrare le varie richieste e reindirizzare il processo di richiesta al corretto step successivo. Le opzioni identificate erano due: tabella in caso di richiesta Hardware e Input testuale nei casi di richiesta diversa da Hardware (Software o altra richiesta) o nel caso in cui si selezioni, nella voce "Fornitore", l'opzione "Richiesta Generica". In basso, il secondo Stage ma visualizzato in forma di modellatore di processi.

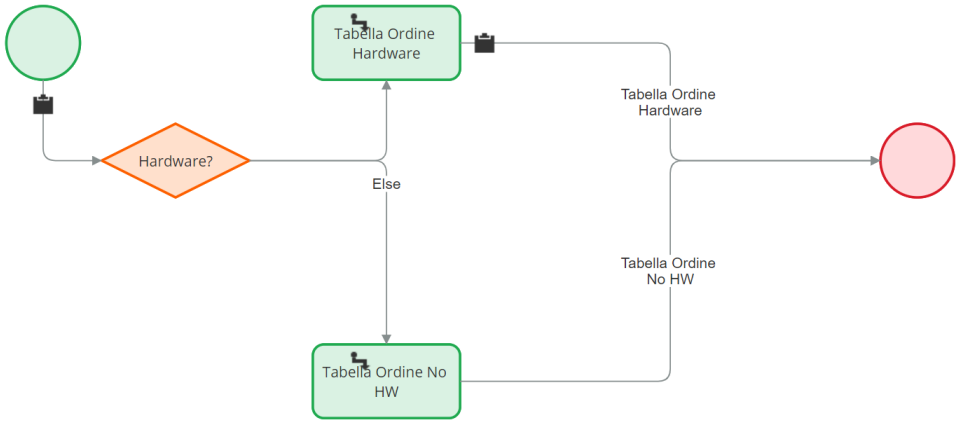

Figura 26 Stage 2

La prima opzione prevedeva una tabella dinamica in grado di visualizzare tre scenari diversi aventi colonne diverse. Per modellare ciò, si è deciso di creare un contenitore unico di dati all'interno della view, ma che contenesse informazioni variabili in base alla tipologia di richiesta scelta visualizzabili in funzione a delle condizioni di visibilità. In termini di interfaccia grafica, il dipendente è in grado di compilare e visualizzare solo quei campi ritenuti necessari ai fini della corretta compilazione della richiesta.

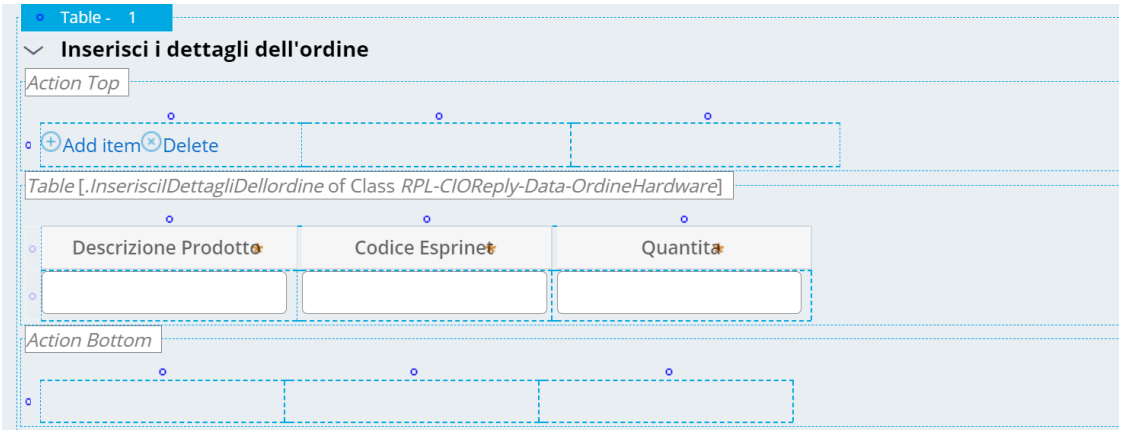

Figura 27 View caso Esprinet

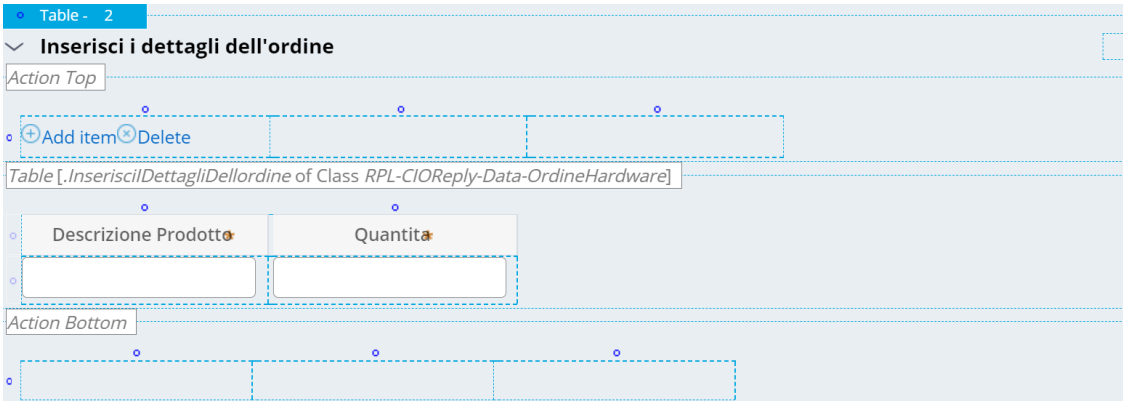

Figura 28 View caso Altro Fornitore

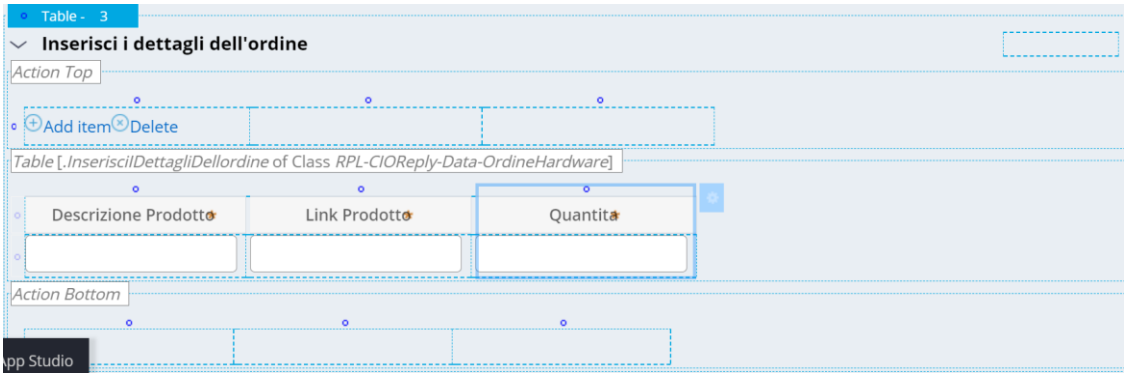

Figura 29 View caso Amazon

Per quanto concerne la schermata per l'inserimento della richiesta generica, è stato creato un campo di testo di tipo "Text Area".

Durante la Sprint Review, non sono state riscontrate particolari criticità ma si è deciso di introdurre, nel solo caso di ordine di tipo Hardware, i campi di costo unitario e Costo Totale al fine di calcolare la spesa finale.

4.4. Terzo Sprint

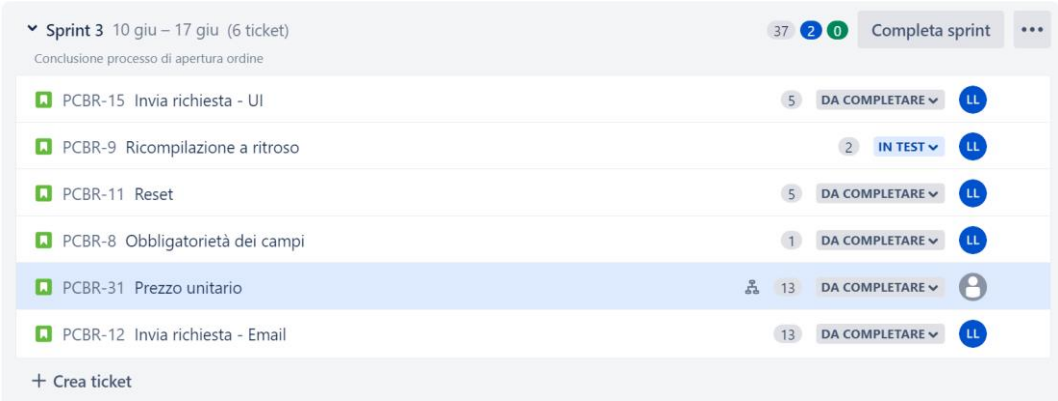

Figura 30 Terzo Sprint backlog

Durante la Sprint Planning del terzo e ultimo Sprint, è stato deciso di introdurre delle User Stories con l'obiettivo di perfezionamento del processo di raccolta dati, andando ad introdurre obbligatorietà dei campi, possibilità di reset campi e possibilità di tornare indietro nei vari step di raccolta dati. A questo, sono state aggiunte tre User Stories di relativo peso: introduzione del prezzo unitario e complessivo relativo ad ogni record della tabella; presenza di un sommario richiesta al momento dell'invio finale; presenza di un messaggio di corretto invio richiesta e invio mail al CIO Manager per approvazione.

Al fine di inserire il prezzo unitario e totale, è stata modificata la view dell'oggetto "OrdineHardware".

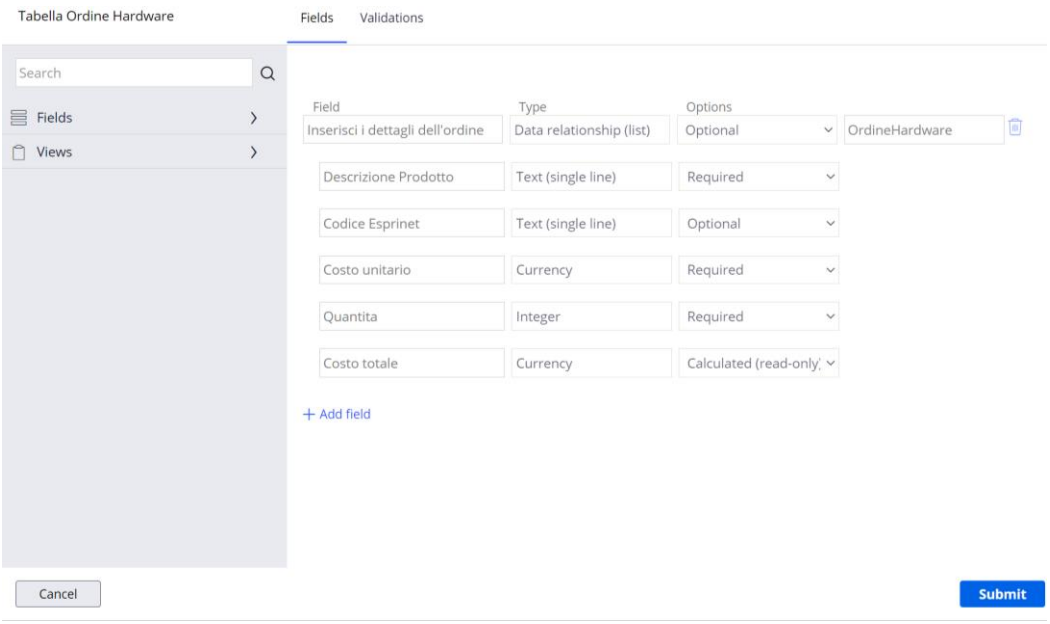

Figura 31 Tabella Ordine Hardware View

Lo step successivo è stato quello di creare uno Step in un diverso Stage capace di visualizzare il sommario della richiesta in caso di tipologia Hardware. Per far ciò, è stata creato una nuova view contenente i campi della richiesta e, in aggiunta, il campo "Spesa Totale RdA", capace di calcolare la somma totale della spesa attraverso la tipologia "calcolated field" in forma di "Sum of" (somma di) dei costi totale di ogni record. A ciò, è seguito una fase di modifica della sezione grafica per poter filtrare la visualizzazione delle corrette colonne in base alle varie tipologie di richiesta hardware, come già fatto per la fase di raccolta.

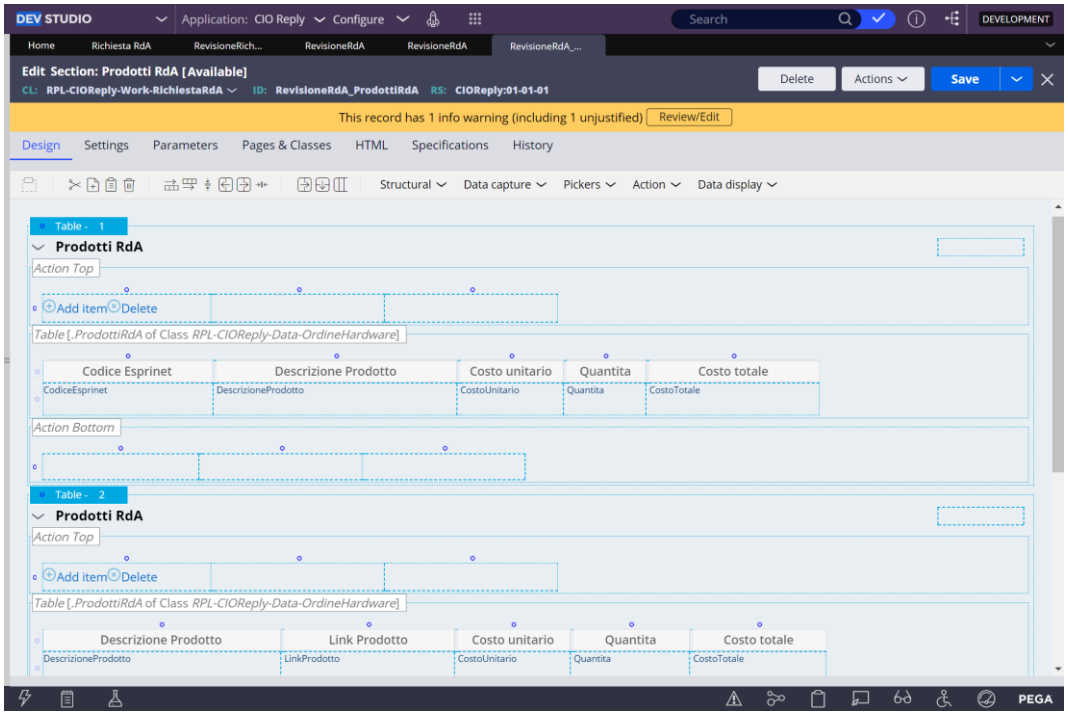

Figura 32 View Revisione Ordine HW

Nel caso in cui la richiesta non è di tipo Hardware oppure lo è ma con Fornitore uguale a “Richiesta generica”, il sistema reindirizza ad un diverso processo che si preoccupa di visualizzare il testo appena riportato dal dipendente durante la compilazione del campo testuale. Questa fase di reindirizzamento è stata comodamente implementata andando ad inserire la condizione sopra riportata nel seguente campo:

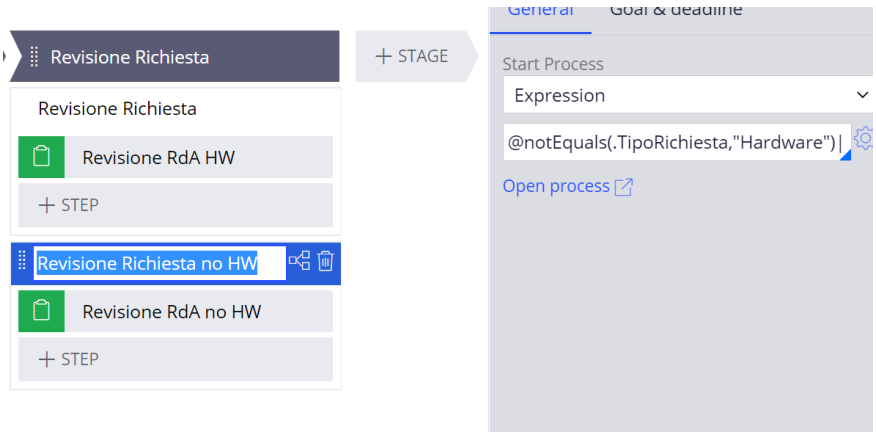

Figura 33 Condizione processo

Terminato lo stage di Revisione, si passa allo stage di Approvazione da parte del CIO manager di tale richiesta. Il Manager ha la possibilità di vedere un riepilogo di tutti i dati che sono stati immessi fino a quel momento dal dipendente. Per far ciò, è stato inserito un processo, al cui interno è presente uno step di tipo "Approval/Rejection" il quale visualizza tramite una singola view, il sommario della richiesta. La view è stata realizzata come negli altri casi, andando a modificare direttamente la section dello step e inserendo una serie di filtri relativi alla visibilità dei singoli campi, in funzione delle precedenti scelte effettuate.

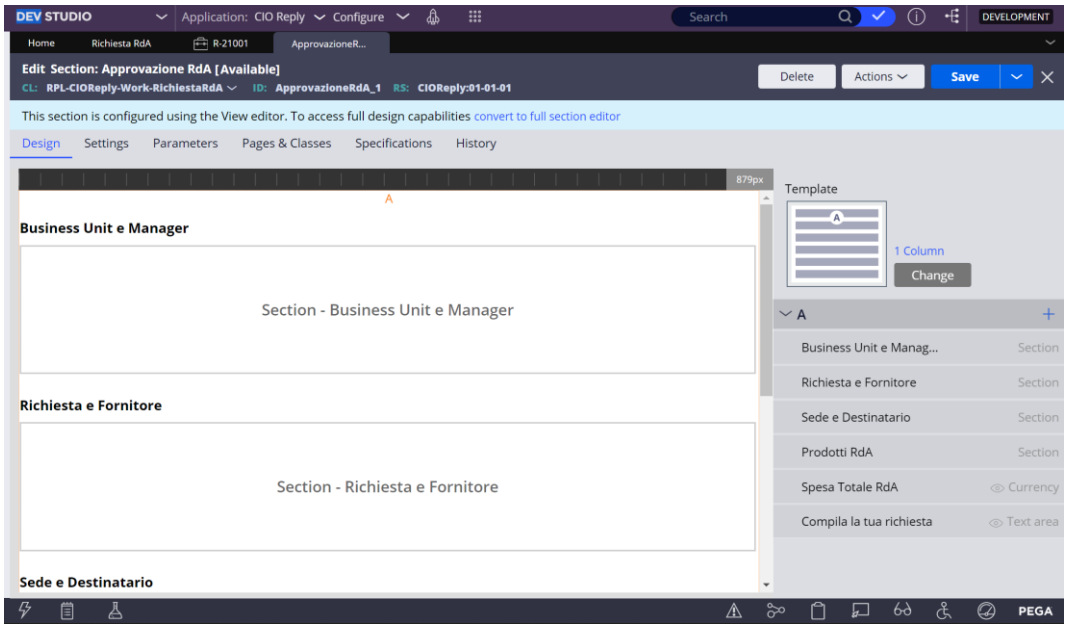

Figura 34 View Approvazione

A questo punto, in base alla decisione del CIO Manager, si delineano due opzioni: Approvata o Rifiutata. Nel primo caso, viene creato uno stage contenente uno step automatizzato che ha lo scopo di inviare la mail di approvazione della richiesta, dove all'interno è possibile modellare il contenuto della mail inserendo varie informazioni già presenti nel database. Tali informazioni sono state raccolte all'interno di un file pdf presente in allegato alla mail. Per creare tale pdf, è stato aggiunto uno Step di automazione "Create PDF" e in seguito ne è stata modificata la view.

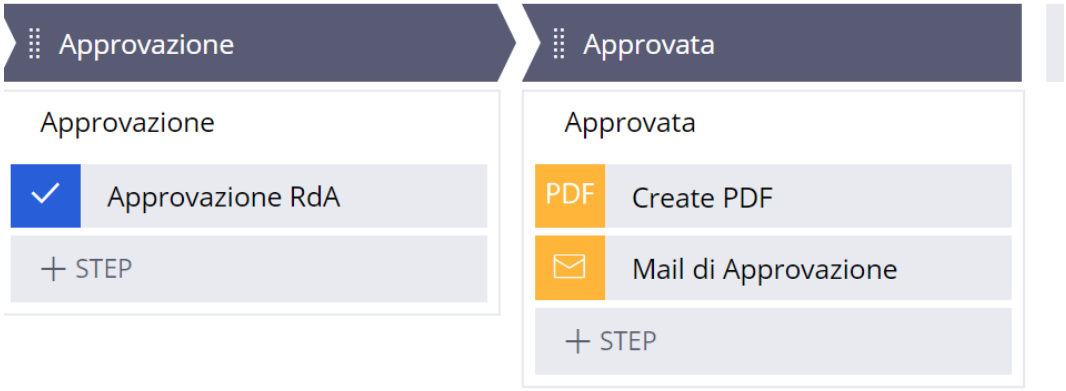

Figura 35 Step Create PDF

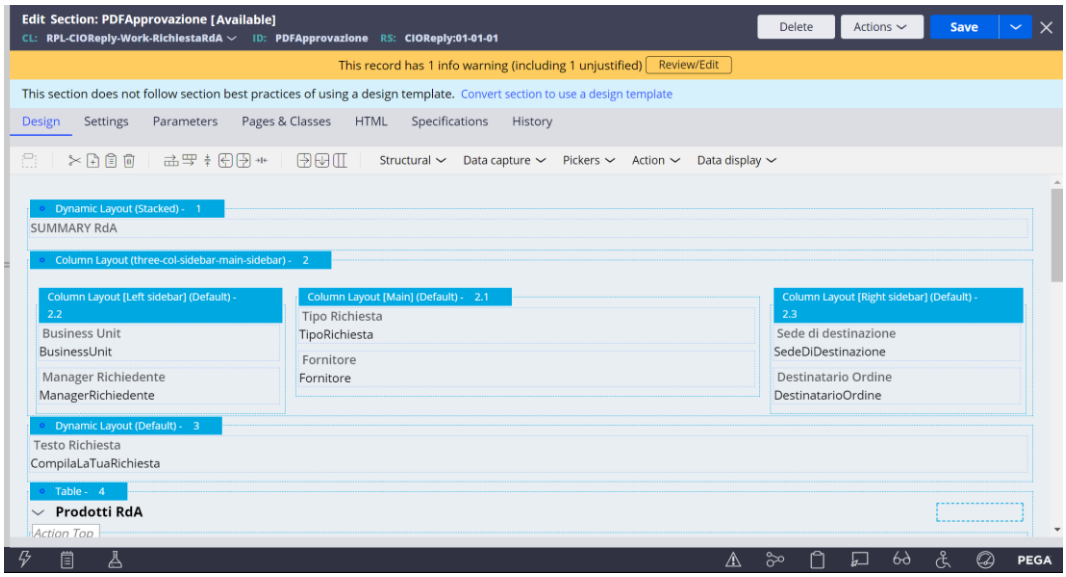

Figura 36 View PDF

Nel secondo caso, viene automaticamente creato uno stage di “Approval Rejection” che verrà percorso durante il processo solo nel caso in cui la richiesta verrà rifiutata. Al suo interno, è presente uno step che invia automaticamente una mail di rifiuto del quale è possibile modificarne il contenuto e specificare come destinatario il proprietario (Owner) della richiesta.

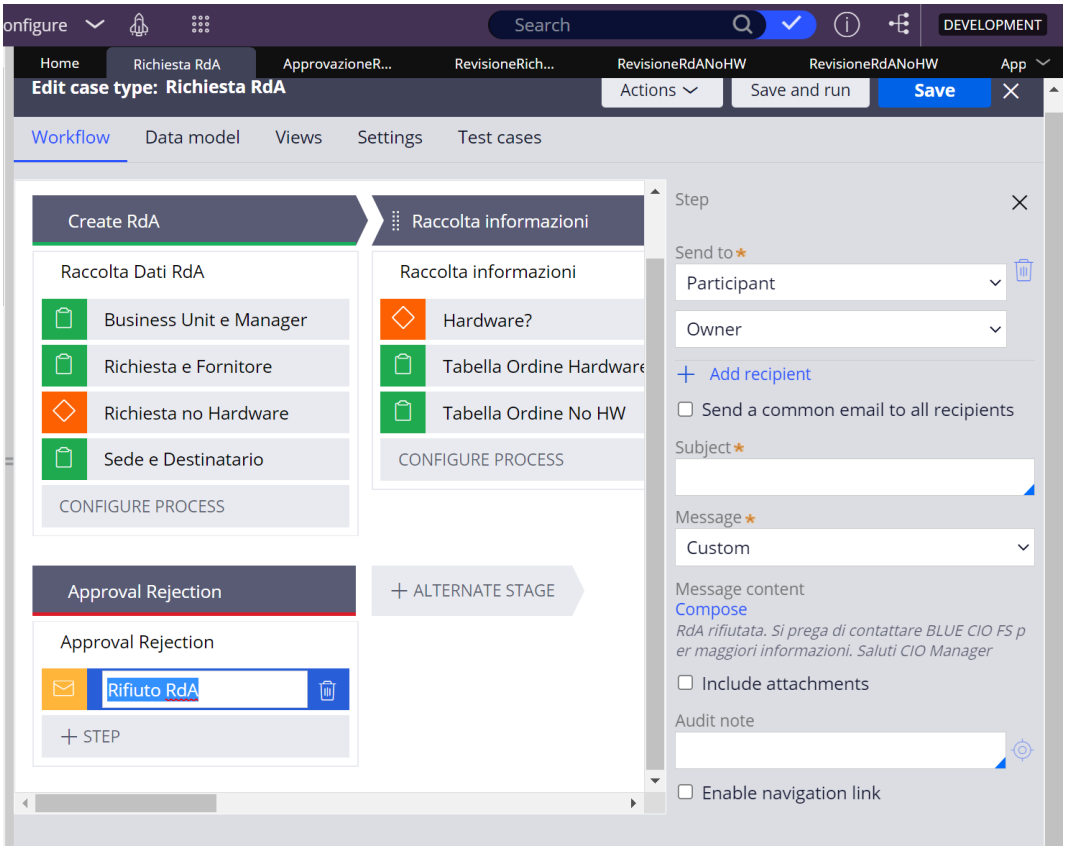

Figura 37 Approval Rejection Stage

Terminate tali User Stories, si è passati all'introduzione di un pulsante di reset all'interno di ogni schermata di Input dati. Per far ciò, si è inserito un button all'interno di ogni view ed è stata creata una diversa data transform per ogni schermata di input. In basso è riportato l'esempio della schermata di "Tabella Ordine Hardware" e della relativa data transform:

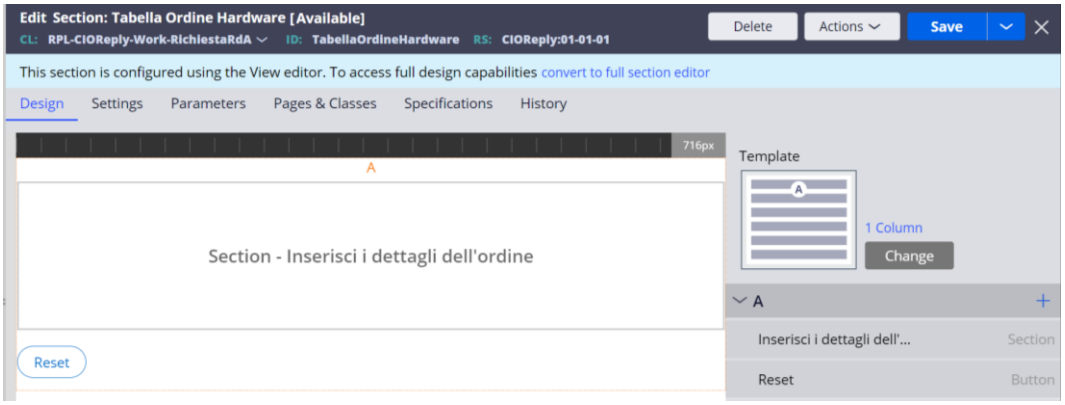

Figura 38 Tasto Reset

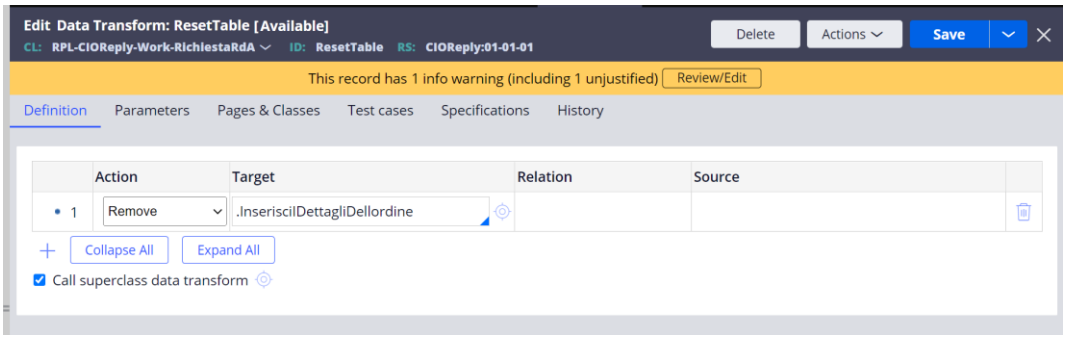

Figura 39 Data Transform Reset

La PCBR-15 richiedeva di inserire un messaggio di pop-up dal momento in cui il dipendente, dopo aver ricontrollato il riepilogo della richiesta, ne confermava l'invio per l'approvazione. Per far ciò, si è risaliti alla section relativa ai pulsanti di default e da qui, è stato inserito un tasto nuovo al set in sostituzione dell'attuale "Submit".

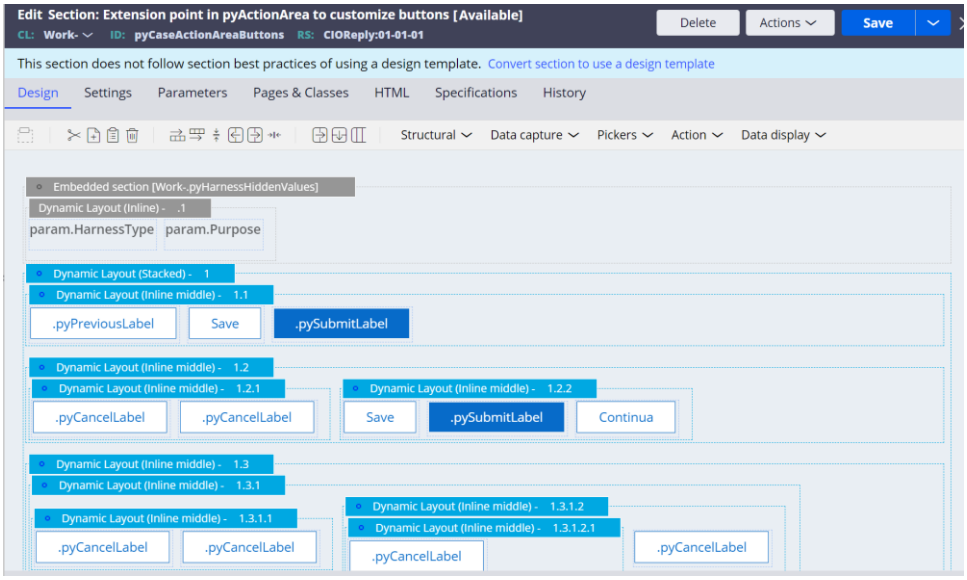

Figura 40 Inserimento tasto 'Continua'

Inserito il tasto “Continua”, questo è stato configurato con una “Local Action” e inserito come Target il “Modal Dialog” che fornisce la schermata di pop-up. Per definire il contenuto del messaggio, è stato creato un template apposito contenente un visualizzatore di tipo Paragraph (paragrafo) per consentire l’inserimento di un messaggio di lunghezza superiore ai 64 caratteri.

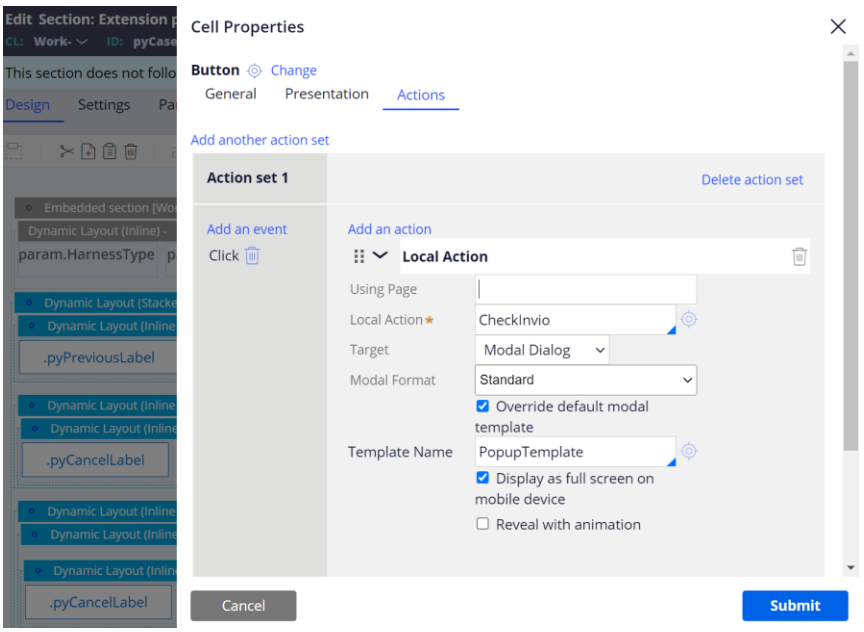

Figura 41 Local Action su tasto

Riguardo le ultime due User Stories, Ricompilazione a ritroso e Obbligatorietà dei campi, la prima è stata resa possibile grazie all'inserimento delle attività all'interno dello stesso processo e inserendo delle azioni di refresh in caso di modifica delle opzioni selezionate; per la seconda è bastato specificare la condizione per cui rendere obbligatorio il campo.

Al termine di questo ultimo Sprint, è stata organizzata, oltre alla Sprint Review, una Sprint Retrospective dove si è andato ad analizzare cosa è andato bene e cosa meno bere e gli eventuali miglioramenti apportabili. Tali dettagli sono presenti nella figura sottostante.

Figura 42 Terza Sprint Review

4.5. Conclusioni

L'esperienza maturata durante tutto il progetto permette di confermare quanto detto nei precedenti capitoli di questa tesi riguardo la tematica Low-Code Platforms. I primi approcci alla piattaforma sono sicuramente i più ardui, come normale che sia, poiché non si conosce bene la piattaforma e le posizioni in cui si trovano tutti i componenti. Pur avendo seguito la formazione offerta dall'Accademy Pega, alcuni casi sono risultati del tutto nuovi e hanno richiesto un supporto da parte di un esperto della piattaforma. Ottenuti i giusti consigli, la realizzazione è risultata abbastanza fluente e gratificante. Vedere delle funzionalità complesse implementate con una semplice spunta ad un'opzione, gratifica lo sviluppatore che

coglie a pieno il risparmio di tempo dato da quel componente preesistente. Già durante il primo Sprint, la velocità di sviluppo è aumentata in parallelo con il progressivo utilizzo della piattaforma, permettendo allo sviluppatore di inserire nuove User Stories nello stesso Sprint. Parlando delle modifiche alla fine di ogni Sprint Review, queste sono risultate abbastanza semplici da effettuare, soprattutto in aspetti riguardanti l'interfaccia Utente. La piattaforma, infatti, permette di modificare abilmente con un puntatore l'oggetto/view con dei semplici click senza dover andare a modificare grandi porzioni di codice. Altro aspetto da non sottovalutare è la possibilità di modificare e salvare come nuove regole alcune impostazioni di base offerte dalla piattaforma. Questo è il caso dei button delle varie schermate che sono stati facilmente modificati partendo dal modello nativo offerto da Pega, con la possibilità di tornare anche al punto di partenze ed eliminare tutte le modifiche appena fatte.

In conclusione, la piattaforma ha performato molto bene per questo progetto e ha permesso di sviluppare un applicativo in sole tre settimane (possibilità di ridurre anche a sole due settimane o meno se considerato un orario full-time) curando sia l'aspetto funzionale sia quello estetico.

INDICE DELLE FIGURE

INDICE DEI GRAFICI

INDICE DELLE TABELLE

BIBLIOGRAFIA

1. **Dahlberg, Daniel.** *Developer Experience of a Low-Code Platform: An exploratory study.* s.l. : Umea Uvinersity, 2020.

2. **Paul, Vincent Jason Wong Kimihiko Iijima Adrian Leow Akash Jain.** Gartner, Magic Quadrant for Enterprise Low-Code Application Platforms. [Online] 2021.

3. **Vailshery, Lionel Sujay.** *Global low-code development platform market revenue 2018-2025.* 12 Febbraio 2022.

4. **Alexander C. Bock, Ulrich Frank.** *Low-Code Platform.* s.l. : University of Duisburg-Essen, 2021.

5. **Hana A. ALSAADI, Dhefaf T. RADAIN, Maysoon M. ALZAHRANI, Wahj F. ALSHAMMARI,.** *Factors that affect the utilization of low-code development platforms: survey study.* Settembre 2021.

6. **Groden-Morrison, Amy.** *Best Practices for Adopting Low-Code and No-Code Platforms. How to Select the Right Platform for Your Team and Manage for Success.* s.l. : Alpha Software Corporation.

7. **RedHat.** Intelligent Process Automation and the Emergence of Digital Automation Platforms. [Online] 2018.

8. **Prinz, Niculin, Rentrop, Christopher and and Huber, Melanie,.** *Low-Code Development Platforms – A Literature Review.* s.l. : AMCIS, 2021.

9. **monday.com/blog/project-management/low-code-platforms/.** [Online] 02 17, 2022.

10. **(ESEM), ACM / IEEE International Symposium on Empirical Software Engineering and Measurement.** *Characteristics and Challenges of Low-Code Development: The Practitioners' Perspective.* 15 ottobre 2021.

11. **Yan, Zhaohang.** *The Impacts of Low/No-Code Development on Digital Trasformation and Software Development.* s.l. : University of Toronto, Canada.

12. **Thinkmoney Powers Customer-Centric Digital Transformation with OutSystems.** ***https://www.outsystems.com/case-studies/thinkmoney-customer-centric-digital-transformation/.*** **[Online]**

13. https://www.prnewswire.com/news-releases/new-york-city-has-deployed-a-covid-19-engagement-portal-built-in-partnership-with-unqork-to-manage-growing-crisis-301033065.html?tc=eml_cleartime. [Online]

14. www.pega.com/it/insights/resources/demo-covid-19-employee-safety-and-business-continuity-tracker. [Online]

15. ***Slide Corso Gestione Progetti.*** **Marco, Alberto De. s.l. : Politecnico di Torino, 2021.**

16. al, C. Fagarasan et. ***Agile, waterfall and iterative approach in information technology projects.*** **s.l. : IOP Conf. Ser.: Mater. Sci. Eng. 1169 012025, 2021.**

17. Senarath, Udesh S. ***Waterfall Methodology, Prototyping and Agile Development.*** **Giugno 2021.**

18. Sutherland, Ken Schwaber and Jeff. ***The Scrum Guide".*** **2017.**

19. https://www.agileway.it/planning-poker-stima-agile-requisiti/. [Online]

20. Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde. ***Una Guida Snella alla Teoria ed alla Pratica di Scrum.*** **2012.**

21. it.wikipedia.org/wiki/Metodo_MoSCoW. [Online]

22. Sirotnak, Casey. ***The Total Economic Impact™ Of Pega Platform for Low Code.*** **s.l. : Forrester, Marzo 2020.**

23. Accademy, Pega. [Online]

24. Akash Jain, Kimihiko Iijima, Adrian Leow, Jason Wong, Paul Vincent. ***Critical Capabilities for Enterprise Low-Code Application Platforms.*** **September 2021.**

25. Koplowitz, Rob. ***The Forrester Wave™: Digital Process Automation Software.*** **14 Dicembre 2021.**

26. Nadine LeBlanc, Jim Davies, Varun Agarwal. ***Magic Quadrant for the CRM Customer Engagement Center.*** **Giugno 2021.**

27. Saikat Ray, Arthur Villa, Naved Rashid, Paul Vincent, Keith Guttridge, Melanie Alexander. ***Magic Quadrant for Robotic Process Automation.*** **26 Luglio 2021.**

28. www.pega.com/it/insights/resources/pega-sales-automation. [Online]

29. Adnan Zijadic, Ilona Hansen, Melissa Hilbert, Steve Rietberg. ***Critical Capabilities for Sales Force Automation.*** **9 Agosto 2021.**

30. Forrester. ***The Total Economic Impact™ Of Pega Customer Decision Hub.*** **febbraio 2020.**

31. Noah Elkin, Julian Poulter, Christy Ferguson, Ilona Hansen, Jeffrey Cohen. ***Critical Capabilities for B2B Marketing Automation Platforms.*** **21 Settembre 2021.**

GLOSSARIO

- **Citizen Developer**: utenti finali che sono in grado di sviluppare applicazioni utilizzando ambienti di sviluppo del dipartimento IT dell'azienda pur operando al di fuori di esso;
- **LCPD**: Piattaforme di sviluppo low-code;
- **RPA**: automazione robotica dei processi
- **Shareholder**: azionista;
- **Stakeholder**: ciascuno dei soggetti direttamente o indirettamente coinvolti in un progetto o nell'attività di un'azienda;
- **Onboarding**: indica il processo d'inserimento di nuovo personale all'interno di una realtà lavorativa, riferendosi ad una delle best practice fondamentali per un'ottimale accoglienza in azienda;
- **Multitenant**: un'architettura in cui una singola istanza di un'applicazione software serve più utenti o, più precisamente, viene usata da più fruitori (chiamati appunto tenant);
- **Framework**: esprime il concetto di una modalità strutturata, pianificata e permanente, che supporta una prassi, una metodologia, un progetto, un sistema di gestione;
- **Business Unit**: divisione aziendale;
- **Picklist**: lista di input con elementi selezionabili;
- **View**: Vista