**UNIVERSITÀ POLITECNICA DELLE MARCHE**
**FACOLTÀ DI INGEGNERIA**
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione


**TESI DI LAUREA**

# Una soluzione di System Integration di servizi RESTful per HR strategy e job progression in Cloud Academy

# A RESTful Services System Integration solution for HR strategy and job progression in Cloud Academy

Relatore

Prof. Domenico Ursino

Candidato

Giorgia Occhionero

**ANNO ACCADEMICO 2021-2022**

**Sommario**

La System Integration è il processo di unificazione di diverse parti di un sistema per farle funzionare insieme senza problemi. Questo processo implica l'interconnessione di diverse tecnologie, applicazioni, processi, infrastrutture e dati in modo da creare un sistema unificato che funzioni come un tutt'uno. Essa può essere realizzata in diversi modi, tra cui l'utilizzo di API (Application Programming Interface), lo scambio di file in formato standard, l'utilizzo di software middleware o di sistemi di gestione dell'integrazione aziendale (Enterprise Integration Management). Nel caso in esame è stata sviluppata un'applicazione a supporto del processo di HR Strategy attraverso la System Integration, l'idea è di integrare i sistemi Key Partner con API REST messe a disposizione dal portale Cloud Academy, portale utilizzato dall'azienda per fruire la formazione ai propri dipendenti.

**Keyword**: System Integration, API REST, HR Strategy

# Indice

# Elenco delle figure

# Introduzione

La System Integration è un processo che consiste nell'incorporare diverse componenti hardware e software in un'unica soluzione funzionale. Queste componenti, eterogenee per natura funzionale e/o tecnologica, possono provenire da diversi produttori, con diverse piattaforme, linguaggi di programmazione e protocolli di comunicazione. Il suo obiettivo è di creare un sistema integrato che funzioni come un'unica entità coerente, permettendo alle componenti di comunicare e collaborare tra loro senza problemi. Questo sistema integrato può essere utilizzato per una vasta gamma di applicazioni, tra cui l'automazione industriale, la sicurezza, la gestione delle infrastrutture, la sanità e molti altri. Nei sistemi moderni la System Integration è fondamentale per gestire la complessità architetturale e infrastrutturale, la trasformazione dei dati, la flessibilità e la sicurezza del sistema. Senza una buona System Integration, i moderni sistemi aziendali possono diventare inefficienti, vulnerabili e difficili da gestire. La System Integration è importante per diversi motivi:

- *Efficienza*: quando le diverse parti di un sistema lavorano insieme senza problemi, l'efficienza dell'intero sistema aumenta.
- *Consistenza*: l'integrazione dei sistemi aiuta a garantire che le informazioni siano coerenti e uniformi.
- *Automazione*: l'integrazione dei sistemi può aiutare a ridurre il lavoro manuale e migliorare l'automazione dei processi aziendali.
- *Scalabilità*: l'integrazione dei sistemi può aiutare a garantire che i sistemi siano in grado di scalare in base alle esigenze dell'azienda.

Il progetto trattato nella presente tesi nasce dalla necessità di supportare il processo di HR Strategy attraverso la System Integration, integrando i sistemi di Key Partner con API REST messe a disposizione dal portale Cloud Academy, utilizzato per la fruizione della formazione al personale. Il progetto è frutto di una collaborazione tra l'Università Politecnica delle Marche e Key Partner s.r.l., società di consulenza ICT, che lavora in ambito di System Integration. Essa ha una forte partnership con TIBCO, software house e fornitore di servizi cloud, e molte altre collaborazioni con importanti attori dell'ICT come Microsoft, AWS e RedHat. L'obiettivo del progetto è l'ammodernamento, l'aumento dell'efficienza e la resilienza all'attuale architettura utilizzata. L'integrazione di sistemi multipli ed eterogenei ha l'obiettivo di garantire la compatibilità con le reali esigenze dell'azienda e, allo stesso tempo, di soddisfare le esigenze a lungo termine, l'agilità aziendale e la rapidità di risposta ai cambiamenti, nonché la capacità di analizzare i dati in modo efficiente e veloce per individuare e far fronte a possibili scenari

futuri. Come soluzione al requisito di integrazione verso la piattaforma Cloud Academy verrà implementata un'architettura software distribuita su una o più applicazioni basate sullo stack TIBCO BusinessWork, system integrator in grado di comunicare mediante protocolli SOAP over HTTP o API REST. L'applicazione che verrà realizzata consentirà all'HR Manager, figura professionale che si occupa dell'HR Strategy, di manipolare i team all'interno della piattaforma Cloud Academy e di analizzare e gestire l'attività di formazione dei diversi dipendenti. Il processo di System Integration può essere suddiviso in diverse fasi, tra cui la pianificazione, l'analisi dei requisiti, la progettazione, l'implementazione, la verifica e la validazione, nonché la manutenzione continua del sistema integrato.

La tesi descrive tutto il lavoro che è stato condotto per progettare e realizzare tale sistema. In particolare, essa è strutturata come di seguito specificato:

- Nel Capitolo 1 si presenterà l'azienda e verranno esposte le motivazioni che hanno generato la necessità di System Integration a supporto dell'HR Strategy.
- Nel Capitolo 2 saranno introdotti nel dettaglio la System Integration e i diversi Pattern di integrazione.
- Nel Capitolo 3 verrà introdotto il progetto e, successivamente, verranno specificati i requisiti del sistema e i relativi casi d'uso.
- Nel Capitolo 4 verranno descritte la progettazione del sistema e la soluzione tecnologica utilizzata.
- Nel Capitolo 5 verrà presentata l'implementazione del sistema, descrivendo nel dettaglio le varie componenti implementate.
- Nel Capitolo 6 saranno descritti e trattati i diversi test effettuati sull'applicazione, analizzando il suo comportamento sui diversi casi di utilizzo.
- Nel Capitolo 7 verrà analizzata l'applicazione implementata, sottolineando in maniera critica gli aspetti positivi e negativi.
- Nel Capitolo 8 verranno tratte le conclusioni in merito al lavoro svolto e si analizzeranno alcuni possibili sviluppi futuri.

CAPITOLO 1

# Contesto di riferimento

*In questo primo capitolo verrà presentata l'azienda presso cui è stata realizzata la presente tesi; inoltre, verrà descritto l'ambito di riferimento che sarà oggetto di questa tesi, ovvero l'HR Strategy.*

## 1.1 L'azienda Key Partner S.r.l.

L'azienda di riferimento per il progetto connesso alla presente tesi è Key Partner S.r.l. (Figura 1.1), la quale appartiene a Key Partner Group, un gruppo indipendente che si occupa di consulenza IT a 360°.

**Figura 1.1:** Logo dell'azienda Key Partner S.r.l.

Key Partner Group si compone di tre anime:

- *Key Partner S.r.l.*: società che offre servizi di consulenza IT con spiccata natura tecnologica. Sfrutta partnership strategiche con Software Vendor leader dei propri segmenti di mercato. Per supportare aziende enterprise nei loro percorsi di digital transformation, offrendo un'ampia gamma di servizi, dalla consulenza specialistica alla realizzazione di soluzioni integrate o di applicazioni chiavi in mano in ambito ICT. Fortemente orientata all'innovazione e alla formazione continua, Key Partner è in grado di interpretare i rapidi cambiamenti introdotti dalla digital transformation. Facendo leva sulle più moderne tecnologie, Key Partner cavalca i principali trend tecnologici (hyperautomation, modern web development, microservices, cloud, devops, machine learning, etc.) garantendo un'ampia gamma di soluzioni e servizi nonché la creazione di valore in ogni iniziativa.

- *Key Value S.r.l.*: società che offre servizi di consulenza IT orientati al business ed al management consulting. In sinergia con Key Partner, consente un approccio end-to-end proponendo al cliente di affiancarlo nella trasformazione digitale. Competenze

strategiche, metodologiche e know-how tecnologico guidano la trasformazione aziendale dall'individuazione del problema alla soluzione digitale (Process Mining, iBPM; tecnologie: Celonis e Apromore).

- *User Group S.r.l.*: società che oltre ad offrire servizi professionali in ambito API Management, completa l'offerta di Key Partner Group con la componente di prodotto. Ha creato, perciò, API Share, una piattaforma in grado di consentire una governance efficace dell'ecosistema API di una organizzazione.

### 1.1.1 Storia e struttura

La società è nata nel 2010 dal desiderio di creare un'azienda ideale, che si distinguesse dalle altre. Ad oggi si compone di 3 sedi sparse per l'Italia, ovvero:

- Roma;
- Milano;
- Termoli.

La storia di Key Partner è stata un'evoluzione continua, confermando il trend di crescita del 20 % degli ultimi due anni, e arrivando a quota 200 dipendenti dislocati nelle sedi di Roma, Milano e Termoli, con un'età media inferiore a 35 anni.

### 1.1.2 Vision e Mission

L'azienda Key Partner, come detto precedentemente, pone l'attenzione all'innovazione e all'interpretazione dei grandi cambiamenti che la digital trasformation introduce. Si impegna a collaborare per affrontare le sfide più grandi. Utilizzando competenze strategiche, metodologiche e know-how tecnologico guidando la trasformazione aziendale dall'individuazione del problema alla soluzione digitale. La mission aziendale può essere delineata attraverso i seguenti punti:

- definizione del processo IT;
- individuazione della soluzione tecnica;
- messa a terra dell'applicazione e manutenzione.

### 1.1.3 Servizi in Key Partner S.r.l.

Key Partner offre una larga gamma di soluzioni in ambito consulenza IT. Tra i principali servizi offerti dall'azienda vi sono:

- *System Integration*: soluzioni in ambito EAI ed API Management a supporto delle mappe applicative complesse dei propri clienti. Sfrutta le collaborazioni con i vendor meglio posizionati nei Gartner di riferimento (come TIBCO e Talend). Offre servizi professionali ad ampio spettro nell'implementazione di soluzioni di System Integration in ecosistemi complessi di importanti clienti enterprise: Enterprise Service Bus, approccio API-First nell'implementazione di soluzioni a microservizi basati su diverse architetture (SOA, REST, Graphml), API Management, messaging, etc.

- *Digital Application Development*: soluzioni applicative full-stack basate su tecnologia open-source ad ampio spettro ed alta personalizzazione. Si occupa di soluzioni su misura (portali, mobile app, e-commerce, microservice, etc.) adatte a svariati contesti e segmenti di mercato. Vanta la collaborazione con l'agenzia grafica interna Deskey (componente interna del gruppo Key Partner) per le tematiche di UI/UX.
- *Data Management*: soluzioni di ingestion, storage, elaborazione e analisi dei dati che, unite all'utilizzo di tecniche di machine learning, arricchiscono il potere informativo dei dati dei clienti. Sfrutta le collaborazioni con partner tecnologici, come, ad esempio, Elasticsearch, per l'implementazione di soluzioni di monitoraggio attivo implementando dashboard interattive e report basati sui KPI definiti a quattro mani con il cliente. L'implementazione di algoritmi di machine learning consente, inoltre, di evolvere l'observability re-attiva ad un approccio predittivo consentendo una proattività decisionale.
- *Cloud* & *Devops*: architetture e soluzioni cloud-native sfruttando gli strumenti devops per snellire, automatizzare e controllare le procedure operative a supporto delle Operations. Sfrutta le collaborazioni con i maggiori distributori di servizi ed infrastrutture cloud (AWS, Azure, Google, RedHat, Suse, etc.). Implementa soluzioni infrastrutturali iPaaS (EKS, AKS, GKE, OCP) e fornisce servizi e soluzioni di DevSecOps avanzato.
- *Hyperautomation*: AI, ML e RPA unite con diverse piattaforme iBPM per ottimizzare e automatizzare i processi di business dei clienti. Sfrutta le collaborazioni con Appian e Outsystems per l'implementazione di soluzioni low-code in ambito iBPM. Lavora in sinergia con Key Value creando, dal punto di vista tecnico, le soluzioni progettate con il business sul tavolo della reingegnerizzazione dei processi aziendali.
- *Application Maintenance Service*: servizio di supporto applicativo e infrastrutturale cross a tutti gli ambiti tecnologici e a tutti i clienti. È l'unico cluster tecnologico "orizzontale" che offre servizi di supporto (anche 24/7) trasversali sulle tecnologie di competenza degli altri cluster tecnologici.

Key Partner negli ultimi anni si è specializzata nel settore insurtech, offrendo soluzioni innovative a diversi operatori italiani e internazionali. Tra le soluzioni proposte e implementate dalla società figurano: la realizzazione di marketplace digitali che hanno accelerato il processo di digitalizzazione delle compagnie assicurative e il loro accesso alla vendita sui canali digitali, il lancio di data management platform specifiche per il contesto assicurativo; la creazione di chatbot intelligenti a supporto dei processi di gestione dei reclami e/o dei sinistri.

## 1.2 HR strategy

Nel corso degli anni, la Human Resource strategy ha avuto un'evoluzione graduale e continua per far sì che l'azienda possa raggiungere i suoi obiettivi più ampi. In particolare, per quanto riguarda la job progression e la formazione del personale, si è man mano investito un quantitativo di energie e risorse sempre crescente. L'obiettivo aziendale è quello di migliorare costantemente tutti i processi interni in un'ottica di standardizzazione e certificazione degli stessi. L'azienda ha deciso di aderire ed impegnarsi nel rispetto dei 10 Principi delle Nazioni Unite riguardanti i diritti umani e del lavoro, ricevendo la certificazione SA8 000, per il proprio sistema di gestione per la protezione dei diritti dei lavoratori, la certificazione ISO 45 001, per i sistemi di gestione per la salute e sicurezza sul lavoro. Quelli di natura ambientale, la certificazione ISO 14 001, per lo standard di gestione ambientale, e legati all'anticorruzione,

la certificazione ISO 9 001, per il sistema di gestione della qualità, la certificazione ISO 27 001, per il sistema di gestione della sicurezza delle informazioni e il rating di legalità.

In ambito HR, i pilastri su cui si basa il gruppo per il raggiungimento di una sempre maggiore soddisfazione dei dipendenti sono:

- *Meritocrazia - valutazione delle performance.* Il percorso di crescita in KP non è legato a tempistiche vincolanti ma è stabilito in base al merito ed ai risultati ottenuti, oltre che alle proprie competenze trasversali e attitudini caratteriali, valutate dal proprio responsabile e dell'HR ogni fine anno con una scheda di valutazione delle performance condivisa all'inizio di ogni anno. Sono previsti corsi di formazione specifici per integrare ed elevare le soft skill legate al proprio profilo professionale. In generale, per acquisire le conoscenze e competenze relative ad ogni ruolo, si possono calcolare almeno un paio di anni di esperienza per ogni funzione, fermo restando sempre che la crescita è legata al merito.

  Con l'obiettivo di una sempre maggiore standardizzazione del processo di assegnazione degli obiettivi professionali e di valutazione delle performance si è passati da una condivisione quasi informale degli obiettivi e delle prospettive di carriera. Ad una sempre più alta formalizzazione del processo e delle informazioni disponibili sul job grading e sulla job progression. La Job grading seguita in KP identifica i seguenti profili professionali:

  - *Consultant*: è in possesso di competenze metodologiche e di know-how tecnologico utile per gestire attività di progetto e accompagnare i clienti nella risoluzione delle loro esigenze di trasformazione digitale.
  - Senior Consultant: gestisce le attività operative di progetto con un alto grado di specializzazione tecnologica e di autonomia esecutiva; si occupa di coordinare e monitorare le attività delle risorse più junior.
  - *Managing consultant*: funge da riferimento tecnico-funzionale per le risorse senior. Il suo ruolo è quello di gestire le relazioni con il cliente, di impostare le attività progettuali in termini di tempistiche e gestione delle risorse, nonchè di monitorarne lo stato di avanzamento. È responsabile della crescita professionale del proprio team.
  - *Manager*: pianifica le attività progettuali dell'azienda, la gestione del budget e individua le opportunità di crescita per il cliente su cui l'azienda può dare un contributo tangibile. Mantiene le relazioni con i clienti e ne gestisce gli accordi commerciali.
  - *Senior Manager*: Appartiene al vertice della piramide e risolve i problemi di business più difficili per i clienti. Si occupa di ampliare il portafoglio clienti e di stabilire le strategie e le politiche aziendali per migliorarne i processi.

- *Work-life-balance.* Per far fronte alle esigenze di tipo personale e di tipo lavorativo dei dipendenti, l'azienda ha deciso di lasciare libera facoltà di scelta al dipendente su quale modalità di lavoro intraprendere. L'azienda ha fin da subito dato la possibilità ai dipendenti di svolgere il proprio lavoro in smartworking, nel vero senso della parola, perché si lavora per obiettivi e, quindi, non è importante il luogo da cui viene svolto il proprio lavoro. Questa decisione ha portato dei benefici a livello di bilanciamento lavoro – vita privata senza degradare le performance lavorative in termini qualitativi e quantitativi.

- *Formazione continua.* Terzo pilastro, non per importanza, è la formazione continua molto importante all'interno dell'azienda KP. Per incontrare le esigenze del mondo del lavoro che risulta essere in continua evoluzione, c'è bisogno di una formazione continua per i dipendenti aziendali. L'azienda dà ai propri dipendenti la possibilità di acquisire certificazioni importanti e personali per arricchire il proprio bagaglio professionale; allo stesso tempo acquisisce conoscenze, competenze e punteggio per affrontare al meglio le gare con i competitor dei relativi settori. Da una recentissima survey interna sulla employee satisfaction è emerso che il punto maggiormente critico in ambito HR è proprio la formazione del personale. I dipendenti hanno espresso, attraverso il questionario, il riscontro circa la mancanza di sufficiente tempo da dedicare alla formazione, aspetto fondamentale nel mondo della consulenza IT, dove l'obsolescenza delle tecnologie è rapidissima. Per questo l'azienda ha come obiettivo futuro quello di permettere ai propri dipendenti di far fronte a questa criticità.

### 1.2.1 Necessità di System Integration a supporto del processo di HR strategy

Proprio dalla necessità di formazione continua per il personale aziendale, che, per quanto detto precedentemente, è risultato essere un punto critico per l'azienda, nasce la necessità di System Integration a supporto del processo di HR strategy. L'idea è di integrare Cloud Academy, portale utilizzato per la formazione del personale, con i sistemi Key Partner, ad esempio, accedendo alle funzionalità della piattaforma direttamente dalla intranet aziendale, dal sistema gestionale ECOS Agile o dagli strumenti offerti dalla piattaforma Microsoft 365. Per fare ciò è necessaria l'integrazione delle API REST di Cloud Academy, in particolar modo quelle per gestire l'assegnazione dei corsi di formazione ai diversi dipendenti e per il monitoraggio dei progressi da loro ottenuti. Questi sono i due casi d'uso presi in considerazione nello svolgimento del progetto aziendale connesso alla presente tesi.

### 1.2.2 Cloud Academy

Cloud Academy è una piattaforma di gestione delle competenze digitali pronta per l'impresa, che fornisce formazione specifica per ruolo, consapevole del contesto e misurabile su scala. Le aziende si affidano a Cloud Academy per erogare formazione sulle principali piattaforme di cloud pubblico, le migliori pratiche per operare attraverso e tra di esse e le funzionalità che il cloud sblocca. I percorsi di apprendimento (Learning Paths) consentono ai team di imparare a fare e a convalidare attraverso video-lezioni guidate da esperti, brevi quiz e laboratori pratici guidati in console cloud reali.

La creazione di un ciclo di valutazione (Assessment Cycle) consente di identificare facilmente le lacune di competenze e di trovare i talenti giusti per guidare le iniziative; i risultati informano i piani di formazione personalizzati per ogni ruolo lavorativo. Il Content Engine rende semplice personalizzare ed estendere i contenuti formativi di Cloud Academy per fornire il contesto di cui i team hanno bisogno, in base allo stack tecnologico, alle politiche interne e alle implementazioni. La formazione quindi grazie alla piattaforma risulta essere personalizzabile, assegnabile e misurabile, in grado di allineare le capacità dell'organizzazione alle esigenze del mercato. L'azienda KP usufruisce della piattaforma Cloud Academy per erogare la formazione ai propri dipendenti. Ogni dipendente all'ingresso in azienda viene assegnato al proprio team di appartenenza all'interno della piattaforma così da avere la possibilità di seguire la formazione adatta al suo percorso aziendale. Il percorso personale di ogni dipendente all'interno della piattaforma parte dalla formazione prevista per il periodo di stage aziendale, a cui prende parte il dipendente all'ingresso in azienda, e prosegue per tutta la durata della sua carriera aziendale.

# CAPITOLO 2

# System Integration – Stato dell'arte

*Nel capitolo corrente verrà spiegato cos'è la System Integration e ne verranno esposte le diverse tipologie esistenti.*

## 2.1 Introduzione alla System Integration – Pattern di integrazione

Il processo che consente di stabilire una comunicazione, di scambiare informazioni o di far funzionare diversi sistemi o sottosistemi come un unico sistema più complesso è noto come System Integration.

L'evoluzione delle metodologie e dei modelli, dall'integrazione point-to-point ai sistemi di messaggistica, alle API Web, ai servizi REST, sarà discussa in questo capitolo insieme ad alcuni principi importanti della System Integration.

### 2.1.1 Integrazione point-to-point

L'integrazione point-to-point e, più in generale, Spaghetti Integration o Star Integration è stata la strategia di integrazione iniziale. Questa strategia prevede la creazione di un canale di comunicazione improvvisato tra ogni coppia di sistemi (Figura 2.1). La sua implementazione non può essere riutilizzata, deve essere progettata appositamente per lo scopo specifico e non è scalabile con il numero di nodi da collegare perché dipende esclusivamente dalle tecnologie e dalle caratteristiche dei due sistemi.

Ciò presenta numerose difficoltà in fase di manutenzione, perché se un sistema subisce una modifica, tutti i connettori che portano e lasciano il sistema devono essere modificati. Poichè il numero di connettori aumenta esponenzialmente con il numero di nodi, essa era appropriata solo per infrastrutture con un numero limitato di nodi. A causa di questi fattori, questa strategia è stata impiegata sempre meno frequentemente nel tempo ed è stata gradualmente sostituita da strategie più scalabili e adattabili.

### 2.1.2 Integrazione verticale

Nell'integrazione verticale, i sistemi sono interconnessi in base alle loro capacità nel tentativo di creare unità funzionali note come silos, spesso isolate l'una dall'altra.

Questi silos, tuttavia, non sono particolarmente efficaci nella gestione dei dati; essi non permettono di estrarre i dati in modo efficiente, limitano la possibilità di vedere come i dati sono correlati, riducendo, così, l'efficacia dell'analisi dei dati; infine, non sono scalabili. I

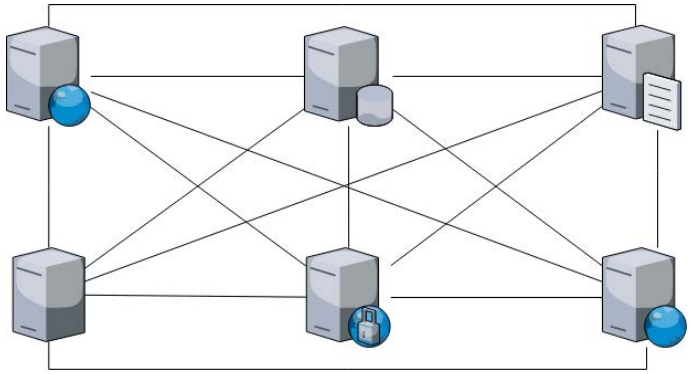

**Figura 2.1:** Spaghetti Integration

sottosistemi non possono essere riutilizzati quando sono necessarie nuove o migliori funzionalità, costringendo la costruzione di nuovi silos, si hanno così, un uso inefficiente e una duplicazione delle risorse.

### 2.1.3 Hub e Spoke

L'architettura hub and spoke è costituita da un hub centrale e da diversi sistemi collegati ad esso tramite spoke (Figura 2.2). Gli spoke espongono particolari adattatori e l'hub si occupa della comunicazione con essi, separando, in questo modo le applicazioni. La richiesta di comunicazione tra le applicazioni è soddisfatta dall'archiviazione permanente dei dati e da tecniche basate sulla messaggistica.

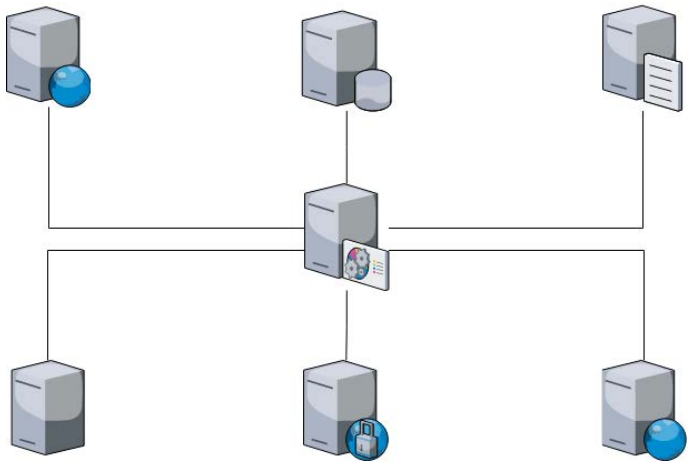

**Figura 2.2:** Architettura Hub e Spoke

Questa architettura presenta diversi vantaggi rispetto alla Spaghetti Integration; tra questi citiamo:

- *Maggiore sicurezza e controllo*: l'hub può implementare meccanismi e politiche per limitare e controllare gli accessi e i flussi di dati.
- *Migliore scalabilità*: se si aggiunge un nuovo spoke, per collegarlo all'hub è sufficiente un adattatore, anziché un connettore per ogni applicazione con cui deve comunicare.
- *Manutenzione più semplice*: nel caso in cui sia necessario modificare l'implementazione di uno spoke, è sufficiente apportare una modifica all'adattatore che lo collega all'hub, lasciando inalterate le altre applicazioni.

- *Monitoraggio più semplice dell'intera infrastruttura e dei dati*, implementato direttamente nell'hub.

## 2.2 Message Oriented Middleware

Una soluzione di comunicazione per i sistemi distribuiti eterogenei e collegati in modo lasco è chiamata Message Oriented Middleware (MOM). Esso aiuta a separare il livello di implementazione da quello di comunicazione, mantenendo un elevato grado di indipendenza da particolari implementazioni, protocolli o sistemi operativi. Questo, insieme alle qualità discusse nelle prossime sezioni, lo rende particolarmente adatto all'implementazione degli ESB.

### 2.2.1 Componenti

La MOM è composta da diverse parti:

- L'oggetto che contiene le informazioni da trasferire viene chiamato messaggio.
- Un'applicazione client è un programma che invia o riceve messaggi.
- Un client che produce e invia messaggi è noto come producer o publisher.
- Un client che si abbona a un servizio viene definito consumer/ subscriber.
- Il provider, noto anche come message broker, è il componente della MOM incaricato di ricevere i messaggi dai producer o publisher, di convalidarli, di eseguire le operazioni necessarie (come la trasformazione, il filtraggio e l'aggregazione) e di inoltrarli ai consumer/ subscriber.
- La coda è una struttura che ospita i messaggi inviati dai produttori ma non ancora letti dai consumatori. Il recupero dei messaggi è possibile in sequenza FIFO (First In First Out).
- I Topic costituiscono un sistema di messaggistica che consente a diversi clienti di scegliere di ricevere messaggi su un determinato argomento.

### 2.2.2 Modelli di comunicazione

Con la messaggistica è possibile sia l'integrazione punto-punto che quella uno a molti. Un produttore trasmette un messaggio su una determinata coda nelle comunicazioni punto-punto. Il messaggio viene successivamente prelevato dalla coda dal consumatore in ordine FIFO. Sulla stessa coda, più produttori possono inviare messaggi, ma solo un consumatore può leggere ogni messaggio. Inoltre, i messaggi in coda possono includere una data di scadenza, dopo la quale il messaggio viene rimosso anche se non è stato letto.

Ci possono essere 0 o più consumatori in attesa in una coda. Il messaggio viene trattenuto finché almeno un cliente non risponde; se nessun cliente risponde entro un tempo massimo il messaggio scade. Invece di utilizzare le code, le comunicazioni uno-a-molti utilizzano l'idea dei topics. Per ogni topic deve esserci almeno un editore, ma ci possono essere zero o più sottoscrittori. I sottoscrittori e gli editori sono mantenuti privati. Ogni abbonato ha due possibilità di sottoscrizione:

- *Sottoscrizione semplice*: per continuare a ricevere le comunicazioni, l'abbonato deve rimanere attivo.

- *Sottoscrizione di lunga durata*: i messaggi vengono conservati finché l'abbonato non si connette e li recupera.

### 2.2.3 Vantaggi e svantaggi

La messaggistica offre una soluzione versatile per un'ampia gamma di casi d'uso, grazie ad alcune caratteristiche. Tra queste citiamo:

- *Asincronicità*: non è necessario che il consumatore e il produttore siano online nello stesso momento. In realtà, MOM offre una serie di opzioni che consentono al produttore di trasmettere e dimenticare il messaggio, avendo, comunque la certezza che quest'ultimo sarà consegnato al consumatore o ai consumatori. Il produttore non subirebbe alcun impatto anche se il consumatore fallisse, e il messaggio verrebbe comunque trasmesso in modo efficace.
- *Trasformazione e instradamento*: un MOM può eseguire operazioni sul messaggio fornito dal produttore per renderlo comprensibile al consumatore. Se combinato con la funzionalità di instradamento, ciò si traduce in un elevato grado di flessibilità; lo stesso messaggio fornito dal produttore può essere modificato in vari modi per soddisfare le esigenze di molti consumatori, garantendo che ciascuno lo riceva correttamente e lo comprenda.

In ogni caso, l'asincronicità non è necessariamente un comportamento preferibile. Molti sistemi operano in sincronia, aspettando la risposta prima di procedere con il loro lavoro. In questi casi è necessario ricorrere ad altre soluzioni in questi casi, poiché il MOM non soddisfa efficacemente alcuni casi d'uso.

Inoltre, la necessità di aggiungere ulteriori componenti al progetto (come il message broker) aumenterebbe la complessità del sistema, rendendolo più difficile da gestire e più vulnerabile a problemi di prestazioni e affidabilità.

## 2.3 SOA – Service Oriented Architecture

### 2.3.1 Introduzione

Un nuovo design, noto come Service Oriented Architecture, è stato creato come risultato della graduale transizione da strutture monolitiche a strutture più modulari e della continua crescita delle reti. Secondo la definizione di OASIS (Organization for the Advancement of Structured Information Standards), la SOA è "un modello per organizzare e sfruttare capacità disperse che possono essere sotto il controllo di più domini di proprietà". Con l'obiettivo di esporre i servizi senza la necessità di creare adattatori o connessioni ad hoc con un hub e di renderli riutilizzabili, SOA è un'architettura eccezionalmente modulare. Il manifesto SOA enumera i principi chiave di questo tipo di architettura; essi sono:

- il valore del business è preferito alla strategia tecnica;
- privilegiare gli obiettivi rispetto ai vantaggi specifici del progetto;
- integrazione naturale piuttosto che integrazione su misura;
- servizi condivisi piuttosto che implementazioni per un determinato scopo;
- flessibilità piuttosto che ottimizzazione;

- il miglioramento evolutivo rispetto alla ricerca del primo ione perfetto.

Questa architettura si basa su protocolli di rete standard comuni, come SOAP e HTTP, e su formati di scambio dei dati comuni, come XML e JSON.

Nella Figura 2.3 illustriamo la tipica architettura SOA.

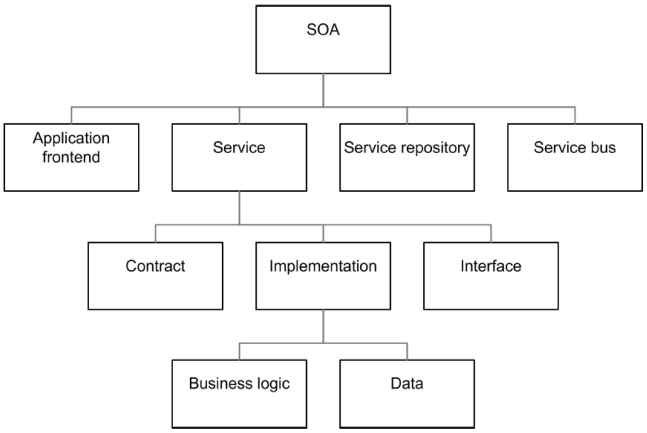


**Figura 2.3:** Architettura SOA

### 2.3.2 Servizi

Una funzione aziendale resa accessibile ed esposta ad altre funzioni o entità è nota come servizio. I servizi devono avere le seguenti qualità:

- *Riutilizzabilità e componibilità*: il servizio può essere utilizzato da più processi, o anche da altri servizi, per costruire servizi più complessi.
- *Contratto di servizio standardizzato*: deve aderire alla descrizione del servizio.
- *Nessuna sovrapposizione funzionale*: se si verifica, i servizi devono essere suddivisi in servizi più piccoli per separare le singole funzionalità, oppure devono essere eliminate le funzionalità duplicate.
- *Accoppiamento libero*: il servizio deve ridurre al minimo la dipendenza da risorse esterne.
- *Incapsulamento*: solo l'interfaccia esterna deve essere accessibile; l'implementazione interna deve essere tenuta nascosta.
- *Autonomia*: i servizi devono poter essere aggiunti, modificati o eliminati senza influenzarsi a vicenda.
- *Scopribilità*: gli archivi di servizi possono essere utilizzati per trovare i servizi.
- *Autodescrittivo*: l'accordo di servizio deve includere una descrizione approfondita dell'interfaccia esposta.
- *Non ha stato*: è necessario evitare il trasferimento di dati tra stati diversi.
- *Agnostico rispetto al luogo e alla lingua*: i servizi devono essere disponibili ovunque e indipendentemente dalla piattaforma o dalla lingua.

Le caratteristiche e le modalità di intervento del servizio sono specificate nel contratto di assistenza. In particolare, si parla di:

- interfaccia del servizio;
- politiche;
- documentazione;
- Qualità del Servizio (QoS);
- prestazioni.

Le operazioni, i formati e i protocolli supportati, gli input accettati e gli output restituiti sono tutti specificati nell'interfaccia del servizio. Un servizio esegue spesso operazioni aggiuntive. Queste azioni vengono eseguite attraverso l'implementazione. Questo componente può essere modificato in qualsiasi momento e non deve essere visibile dall'esterno del servizio. Il cliente del servizio sarà completamente all'oscuro di qualsiasi modifica, purché l'interfaccia rimanga invariata. Ciò offre allo sviluppatore una grande libertà e consente un miglioramento continuo. Inoltre, poiché l'utente vedrà solo una virtualizzazione dei dati reali, i servizi consentono di nascondere le organizzazioni aziendali e l'implementazione fisica dei dati.

### 2.3.3 ESB - Enterprise Service Bus

L'Enterprise Service Bus, o ESB, è un software cruciale nelle SOA. Consente l'esposizione dei servizi e la comunicazione tra diversi sistemi. Simile ai bus fisici dei computer, l'ESB funziona come segue: ogni richiesta viene consegnata al bus, che si occupa di tradurla, convertirla e instradarla opportunamente al destinatario. Dopo essere stata elaborata e trasmessa al destinatario, la risposta viene nuovamente inviata al bus. L'ESB trasforma qualsiasi sistema ad esso collegato in un'interfaccia utilizzabile da tutti gli altri sistemi. Questo metodo è spesso adottato per consentire il riutilizzo di un sistema legacy e per offrire un servizio. Separando le implementazione dall'interfaccia accessibile e utilizzando la filosofia SOA si rende possibile l'integrazione di sistemi che utilizzano protocolli antiquati o formati di dati proprietari. La connessione tra i sistemi collegati all'ESB è sotto il suo controllo. A tal fine, l'ESB mette in atto protocolli di routing e converte i messaggi in formati comprensibili. Di conseguenza, la complessità dei sistemi da collegare è molto ridotta. È anche molto più semplice aggiungere, eliminare o aggiornare i componenti, perché il richiedente e il destinatario non devono concordare in anticipo il formato o altri dettagli della richiesta o della risposta. Inoltre, l'ESB offre molti dei vantaggi dell'architettura hub-and-spoke, tra cui una manutenzione più semplice, una maggiore adattabilità, una maggiore sicurezza e un monitoraggio più semplice. Tuttavia, questa tecnica presenta anche alcune carenze. In particolare:

- L'ESB riceve tutte le richieste e le risposte, il che potrebbe causare un collo di bottiglia.
- La natura centralizzata dell'ESB può renderlo un singolo punto di fallimento in termini di affidabilità.
- L'ESB può essere difficile da installare in molte aziende.

## 2.4 REST – Representational State Transfer

Un approccio architettonico per le applicazioni basate sul Web chiamato REST (Representational State Transfer) è caratterizzato dalle seguenti caratteristiche:

- *semplicità* attraverso la coerenza dell'interfaccia;
- *scalabilità*: deve essere in grado di ospitare un numero elevato di componenti e di garantire un numero elevato di interazioni;
- i componenti devono essere capaci di *adattarsi* alle mutevoli esigenze, devono essere configurabili e personalizzabili;
- *efficacia* e prestazioni della rete;
- *affidabilità* e resilienza ai guasti;
- *visibilità*: capacità di tenere traccia delle interazioni tra i componenti;
- *portabilità*: non ci devono essere dipendenze da piattaforme o linguaggi.

### 2.4.1 Vincoli di implementazione

Per essere considerata RESTful, un'applicazione deve rispettare alcune restrizioni che garantiscono il rispetto degli attributi sopra citati. Tali restrizioni sono le seguenti:

- *Architettura client-server*: scalabilità, portabilità e flessibilità sono rese possibili dalla separazione del livello di presentazione dalla logica di business.
- *Statelessness*: il server non memorizza alcuna informazione sui dati di sessione, il che favorisce le prestazioni grazie alla riduzione dell'onere.
- *Cacheability*: consentendo al client di memorizzare nella cache le risposte, il server può scegliere di ridurre le interazioni e aumentare la velocità.
- *Separazione dei livelli*: un client non dovrebbe essere in grado di sapere se è connesso o meno al server finale. L'uso di bilanciatori di carico, livelli di sicurezza, proxy o altri intermediari è ora possibile senza avere un impatto sui client o sui server.
- *L'interfaccia uniforme*, che si ottiene rispettando quattro vincoli aggiuntivi:
  - L'identificazione delle risorse deve essere univoca (ad esempio, utilizzando URis).
  - Le risorse non vengono acquisite e restituite direttamente, ma vengono manipolate attraverso rappresentazioni. Esse sono rappresentate virtualmente (spesso in JSON o XML).
  - Quando si inviano messaggi autodescrittivi, è necessario fornire i metadati e la semantica del messaggio.
  - HATEOAS (Hypermedia As The Engine Of Application State): ogni risorsa deve avere collegamenti ipertestuali in modo che l'utente possa trovare altre risorse ad essa collegate.

Queste limitazioni portano a un progetto rigorosamente incentrato sulle risorse, portatile, modulare e scalabile, ideale per applicazioni su scala Internet e capace di garantire un legame molto lasco tra implementazione e rappresentazione dei dati.

### 2.4.2 HTTP method e CRUD operation

Le applicazioni che utilizzano l'architettura RESTful si affidano completamente all'HTTP come meccanismo di comunicazione. Le risorse vengono modificate con operazioni CRUD attraverso i meccanismi integrati nel protocollo; questi ultimi sono:

- *GET*: utilizzato per ottenere lo stato di una risorsa
- *PUT*: viene utilizzato per stabilire (o sostituire) lo stato di una risorsa.
- *POST*: viene utilizzato per far sì che la risorsa elabori la rappresentazione fornita in una richiesta.
- *DELETE*: si usa per rimuovere uno stato da una risorsa.

Mentre gli altri metodi sono idempotenti, ovvero in grado di gestire nuovamente richieste identiche, il metodo GET equivale a un accesso in sola lettura. Gli unici metodi memorizzabili nella cache sono GET e POST.

## 2.5 Microservizi

Un'evoluzione della SOA è l'architettura basata sui microservizi. Un'applicazione basata su microservizi è costituita da diversi microservizi collegati in modo lasco, autonomo e a grana fine, che utilizzano protocolli portatili e indipendenti dalla piattaforma. Ogni microservizio è una funzione aziendale indipendente che espone caratteristiche specifiche attraverso un'interfaccia distinta.

### 2.5.1 Vantaggi e svantaggi:

I microservizi offrono una serie di vantaggi; questi sono:

- *Agilità*: diversi servizi possono essere costruiti in modo indipendente e simultaneo da piccoli team che lavorano in contesti ben specificati, accelerando la velocità di sviluppo.
- Poiché le modifiche a un microservizio non richiedono modifiche ad altri elementi, sono possibili *tecniche di sviluppo continuo*, una *gestione più rapida* delle modifiche, degli avanzamenti tecnologici e delle correzioni dei bug.
- *Scalabilità*: se uno dei microservizi ha un picco di domanda, non è necessario scalare l'intero sistema.
- *Affidabilità*: se un microservizio si guasta, non influisce sull'intera architettura, proteggendo gli altri servizi.
- *Flessibilità*: l'aggiunta o l'eliminazione di una funzionalità è semplice come aggiungere o rimuovere un microservizio, senza impattare sugli altri.
- *Flessibilità tecnica*: è possibile utilizzare diverse tecnologie per scrivere una varietà di servizi, consentendo ai team di sviluppo di scegliere quella che meglio si adatta alle loro esigenze.

Tra i difetti di questa architettura ci sono:

- traffico di rete e latenza elevati;

- maggiore impegno in termini di test e implementazione;
- aumento della complessità dell'architettura nel suo complesso;
- necessità di maggiori risorse computazionali;
- necessità di orchestrazione dei microservizi.

### 2.5.2 Microservizi vs SOA

Essendo uno sviluppo della SOA, i microservizi condividono molti concetti legati ad essa, ma si differenziano anche per i seguenti aspetti:

- *Ambito*: i microservizi hanno un ambito applicativo, mentre la SOA ha principalmente un ambito aziendale.
- *Indipendenza*: i microservizi sono completamente indipendenti l'uno dall'altro, mentre la SOA richiede un canale comune (spesso un ESB) per la comunicazione.
- *Interoperabilità*: i microservizi utilizzano semplici protocolli standard, come HTTP e REST, mentre la SOA utilizza una gamma più ampia di protocolli.
- *Governance*: i microservizi offrono una maggiore flessibilità rispetto alla SOA, che ha una governance condivisa tra tutti i suoi servizi.
- *Archiviazione*: mentre i microservizi possono avere uno storage dei dati indipendente, la SOA offre un livello di storage dei dati condiviso.
- *Granularità*: mentre i servizi SOA hanno dimensioni che vanno da quelle modeste a quelle aziendali, i microservizi sono a grana fine.
- *Condivisione delle risorse*: i microservizi non scambiano risorse come fa la SOA.

Grazie a queste distinzioni, SOA e microservizi non sono in opposizione tra loro. Al contrario, dle due architetture si completano a vicenda e si adattano meglio a requisiti diversi.

# CAPITOLO 3

# Soluzione: Integrazione delle API REST

*In questo capitolo, dopo una breve introduzione al progetto, verrà trattata la fase di analisi dei requisiti, fase importante della progettazione di un sistema. Successivamente, verranno definiti i casi d'uso del sistema che si vuole implementare per comprendere al meglio le principali funzionalità.*

## 3.1 Introduzione al progetto

Il progetto in esame nasce dalla necessità di supportare il processo di HR Strategy attraverso la System Integration, integrando i sistemi Key Partner con API REST messe a disposizione dal portale Cloud Academy, portale utilizzato per la fruizione della formazione al personale. L'obiettivo del progetto è l'ammodernamento, l'aumento dell'efficienza e la resilienza dell'attuale architettura utilizzata. L'integrazione gioca un ruolo centrale nel raggiungimento di questi risultati.

L'integrazione di sistemi multipli ed eterogenei ha l'obiettivo di garantire la compatibilità con le reali esigenze dell'azienda e, allo stesso tempo, soddisfare le esigenze a lungo termine, l'agilità aziendale e la rapidità di risposta ai cambiamenti, nonchè la capacità di analizzare i dati in modo efficiente e veloce per individuare e far fronte a possibili scenari futuri.

## 3.2 Analisi dei requisiti

L'analisi e il reperimento dei requisiti sono difficili da standardizzare, poiché dipendono dal sistema da sviluppare. La fase di definizione delle problematiche che il sistema dovrà affrontare e delle caratteristiche che dovrà avere viene definita raccolta dei requisiti. I requisiti sono definiti con gli stakeholder durante la prima fase del progetto e sono espressi in un linguaggio naturale; per questo si può incorrere in ambiguità e incertezze. Di conseguenza, è fondamentale formalizzarli in modo appropriato per evitare le difficoltà sopra descritte.

L'analisi dei requisiti, processo che si pone l'obiettivo di formalizzare i requisiti, ricopre un ruolo fondamentale, essendo la prima fase indispensabile per la progettazione di un sistema software. La metafora più comune per la gestione dei requisiti nei progetti software è l'altalena, rappresentata nella Figura 3.1, dove vengono illustrate le difficoltà che si possono incontrare durante la fase di analisi dei requisiti.

I requisiti possono essere suddivisi in due categorie, ovvero requisiti funzionali e requisiti non funzionali.

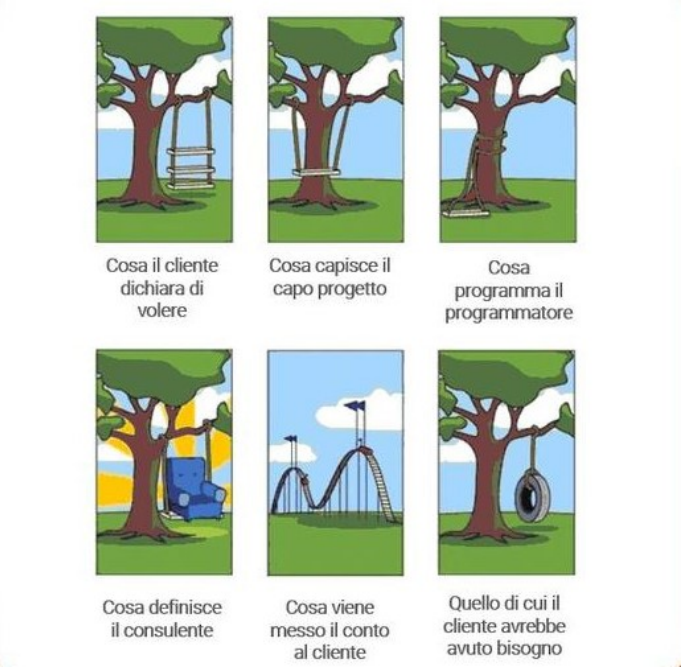


**Figura 3.1:** Metafora dell'altalena

### 3.2.1 Requisiti funzionali

I requisiti funzionali definiscono una o più funzioni di un sistema o i requisiti dei suoi componenti, specificando la tipologia degli ingressi e delle uscite e il comportamento. Si presentano sotto forma di elenchi di funzioni o servizi che il sistema deve svolgere, nonché di descrizioni del comportamento del sistema in risposta a determinati input e in corrispondenza di specifici scenari. La completezza e la coerenza sono due qualità fondamentali delle specifiche dei requisiti.

Un documento di specifica dei requisiti si dice "completo" se tutte le esigenze dell'utente sono dichiarate; si dice, anche, che è coerente e consistente se non ci sono requisiti in competizione. Sebbene questi risultati siano semplici per i sistemi di piccole dimensioni, richiedono chiaramente più lavoro per i sistemi con un numero elevato di criteri. In quest'ultimo caso, la sfida sta proprio nella maggiore frequenza di errori e, soprattutto, di omissioni. Infine, i requisiti funzionali specificano ciò che il sistema che si vuole costruire deve eseguire.

Nel caso preso in esame per l'azienda Key Partner s.r.l., i requisiti funzionali che dovrà avere il sistema finale che verrà realizzato sono i seguenti:

- RF1: il sistema dovrà consentire l'assegnazione di un nuovo utente ad un team.
- RF2: il sistema dovrà consentire di togliere un utente da un team.
- RF3: il sistema dovrà consentire di visualizzare tutti gli utenti di un team.
- RF4: il sistema dovrà consentire di monitorare i progressi dei componenti di uno specifico team.
- RF5: il sistema dovrà consentire di monitorare i progressi di un singolo membro di un team.

### 3.2.2 Requisiti non funzionali

I requisiti non funzionali, a differenza di quelli funzionali visti precedentemente, descrivono vincoli sul processo di sviluppo dell'intero sistema. Non riguardano le specifiche funzioni fornite dal sistema, ma possono riferirsi sia a caratteristiche che si desiderano per esso e sia a vincoli ai quali il sistema deve sottostare. I requisiti non funzionali sono strettamente vincolati alla necessaria modalità di interazione del relativo sistema con altri componenti.

Una possibile classificazione dei requisiti non funzionali è la seguente:

- *requisiti di prodotto*: descrivono le caratteristiche del prodotto, in termini di usabilità, efficienza, affidabilità e portabilità.
- *requisiti di processo*: derivano dalle politiche e dalle procedure organizzative relative al cliente e allo sviluppatore.
- *requisiti esterni*: si riferiscono a fattori esterni al sistema ed al relativo processo di sviluppo, come requisiti legislativi o di interoperabilità.

Nella Figura 3.2 viene mostrata una classificazione gerarchica dei requisiti non funzionali.

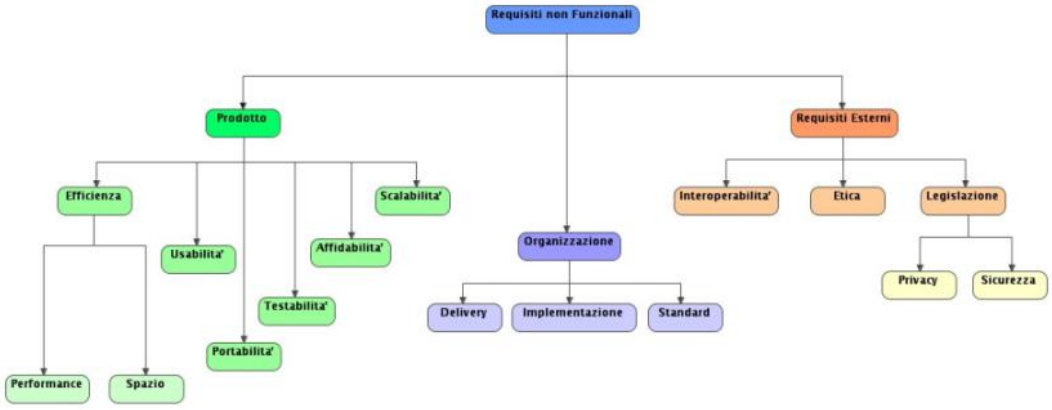


**Figura 3.2:** Classificazione gerarchica dei requisiti non funzionali

Nel caso preso in esame per l'azienda Key Partner s.r.l., i requisiti non funzionali che dovrà avere il sistema finale che verrà realizzato sono i seguenti:

- RNF1: il sistema dovrà essere sempre disponibile per assicurare all'utilizzatore finale la continua attività.
- RNF2: il sistema dovrà essere sufficientemente veloce.
- RNF3: il sistema dovrà essere intuitivo e di facile utilizzo.
- RNF4: il sistema dovrà essere affidabile.

## 3.3 Definizione dei casi d'uso

A partire dai requisiti si possono creare una serie di scenari, ciascuno dei quali individua una sequenza d'uso del sistema da implementare. Gli scenari, detti anche "casi d'uso", descrivono i modi in cui il sistema verrà utilizzato. La mancata definizione di questi ultimi potrebbe causare problemi di programmazione che si rifletteranno sul prodotto finale.

I casi d'uso si possono definire come la specifica di una sequenza di azioni, incluse eventuali sequenze alternative e sequenze di errore, che un sistema può eseguire interagendo con attori esterni. Il diagramma dei casi d'uso in UML (Unified Modeling Language), linguaggio diventato standard per la modellazione dei requisiti, è la rappresentazione di diagrammi che descrivono funzioni o servizi offerti da un sistema così come sono utilizzati dagli attori che interagiscono con esso.

Nel caso in esame, gli attori che interagiscono con il sistema saranno sicuramente persone fisiche e non sistemi automatici; in particolare, saranno persone che si dedicano all'HR Strategy aziendale, che rappresenteremo come attore principale. Nella Figura 3.3 viene mostrato il diagramma dei casi d'uso che specifica come qualunque persona che si occupa di HR Strategy sarà in grado, attraverso il sistema che verrà realizzato, di manipolare i team all'interno della piattaforma Cloud Academy e di analizzare e gestire l'attività di formazione dei diversi dipendenti.

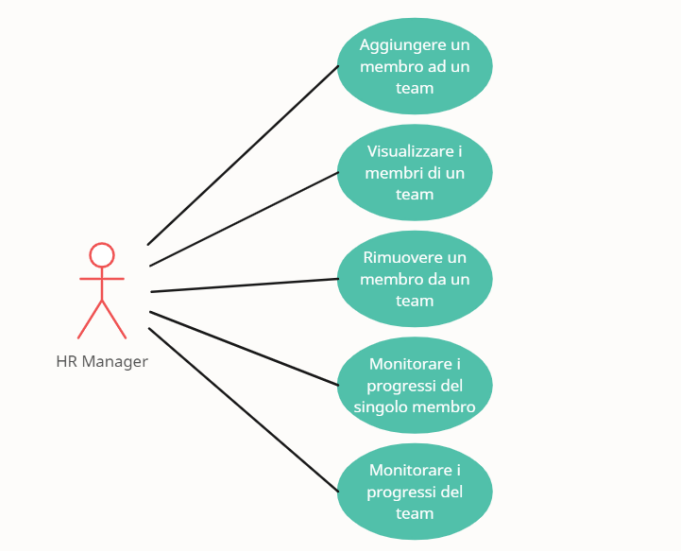


**Figura 3.3:** Diagramma dei casi d'uso del sistema finale relativo all'HR Manager

CAPITOLO 4

# Progettazione del sistema

*La fase di progettazione è una fase importante per lo sviluppo di un sistema software ed è indispensabile per la fase successiva di implementazione. In questo capitolo ci occuperemo della progettazione del sistema oggetto della presente tesi.*

## 4.1 Definizione della soluzione

In questa sezione analizzeremo la conversione dei requisiti funzionali in requisiti tecnici. In particolare, vengono illustrati alcuni modelli di architettura, strumenti e tecnologie identificati per soddisfare tali requisiti. Come soluzione al requisito di integrazione verso la piattaforma cloud academy verrà implementata un'architettura software distribuita su una o più applicazioni basate sullo stack TIBCO BusinessWorks. BusinessWorks è un system integrator in grado di comunicare mediante protocolli SOAP over HTTP o API REST.

Ciò permetterà, quindi, lo scambio di informazioni tra i sistemi aziendali Key Partner e la piattaforma Cloud Academy.

In relazione ai casi d'uso identificati e descritti nel capitolo precedente, questa sezione riporta la traduzione tecnica della soluzione.

- *RF1*: da interfaccia web (ECOS, Microsoft 365, intranet) richiedere l'assegnazione di un utente ad un team. Il manager HR, in modalità on-demand, deve poter richiamare la funzionalità di assegnazione utente, inserendo in input il codice del team e le informazioni personali dell'utente. Il manager HR vedrà sullo schermo la conferma dell'avvenuta assegnazione nella stessa sessione.

- *RF2*: da interfaccia web (ECOS, Microsoft 365, intranet) richiedere la rimozione di un utente ad un team. Il manager HR, in modalità on-demand, deve poter richiamare la funzionalità di rimozione dell'utente, inserendo in input il codice del team e l'indirizzo email dell'utente. Il manager HR vedrà sullo schermo la conferma dell'avvenuta rimozione nella stessa sessione.

- *RF3*: da interfaccia web (ECOS, Microsoft 365, intranet) richiedere la visualizzazione della lista dei membri di un team. Il manager HR, in modalità on-demand, deve poter richiamare la funzionalità di visualizzazione, inserendo in input il codice del team. Il manager HR vedrà sullo schermo le informazioni richieste nella stessa sessione.

- *RF4*: da interfaccia web (ECOS, Microsoft 365, intranet) richiedere la visualizzazione dell'andamento della formazione degli utenti di un team. Il manager HR, in modalità on-demand, deve poter richiamare la funzionalità di rimozione dell'utente, inserendo in input il codice del team. Il manager HR vedrà sullo schermo le informazioni richieste nella stessa sessione.
- *RF5*: da interfaccia web (ECOS, Microsoft 365, intranet) richiedere la visualizzazione dell'andamento della formazione di un utente. Il manager HR, in modalità on-demand, deve poter richiamare la funzionalità di rimozione dell'utente, inserendo in input le informazioni personali richieste dell'utente. Il manager HR vedrà sullo schermo le informazioni richieste nella stessa sessione.

Per poter garantire il requisito di ricezione della conferma sullo schermo dell'avvenuta esecuzione dell'operazione richiesta o delle informazioni richieste dall'HR Manager, l'applicazione TIBCO deve implementare un pattern di integrazione sincrono, in modo che l'esito dell'elaborazione venga restituito all'utilizzatore.

Ai requisiti funzionali precedenti possiamo inoltre aggiungere alcuni requisiti funzionali cross, requisiti che descrivono le funzionalità o le caratteristiche di un sistema che devono essere implementate in modo da interagire con altri sistemi o componenti. I requisiti funzionali cross sono importanti per garantire che un sistema funzioni correttamente e in modo affidabile in un ambiente distribuito complesso.

I requisiti funzionali cross relativi al sistema oggetto della presente tesi sono i seguenti:

- *Scelta architetturale*: API REST, più leggera, moderna, scalabile, estendibile, gestita nativamente dallo strumento.
- Necessità di gestire l'accesso ai servizi di Cloud Academy mediante chiamata a endpoint di generazione *token* secondo lo standard OAuth2 con flusso client credential.
  - Il token restituito ha una validità di 10h, è necessario salvare il token in memoria insieme alla data di scadenza dello stesso per poterlo riutilizzare nelle 10h successive.
- *Gestione dei retry in caso di KO*: effettuare 3 tentativi a distanza di 5 secondi l'uno dall'altro.

Mediante questa fase di traduzione dei requisiti funzionali in requisiti tecnici è stato possibile creare dei diagrammi di sequenza che illustrano le interazioni tra le componenti del sistema. Nella Figura 4.1 possiamo vedere il possibile diagramma di sequenza relativo alla soluzione tecnica ottenuta dalla traduzione dei requisiti funzionali.

I diagrammi di sequenza sono uno strumento potente per comprendere e progettare sistemi complessi, in quanto forniscono una chiara visione delle interazioni tra i componenti coinvolti in uno specifico caso d'uso. Possono essere utilizzati per identificare potenziali problemi e colli di bottiglia in un sistema, nonché per migliorare la comunicazione e la collaborazione tra i membri del team di sviluppo.

Nel diagramma successivo, abbiamo tre oggetti, "Kp", "TIBCO BW6" e "CloudAcademy". L'utente "KP" avvia l'interazione inviando un messaggio di "richiesta dati" all'oggetto "TIBCO BW6", questo farà un controllo sul token posseduto dall'utente e, in caso di non validità dello stesso, effettuerà una chiamata all'oggetto "CloudAcademy" per generarne uno valido. Dopo aver ricevuto il token valido, l'oggetto "TIBCO BW6" interrogherà nuovamente l'oggetto "CloudAcademy" al fine di ottenere le informazioni richieste dall'utente. Infine, ottenuta la risposta dall'oggetto "CloudAcademy", l'oggetto "TIBCO BW6" inoltrerà la stessa all'utente "KP".

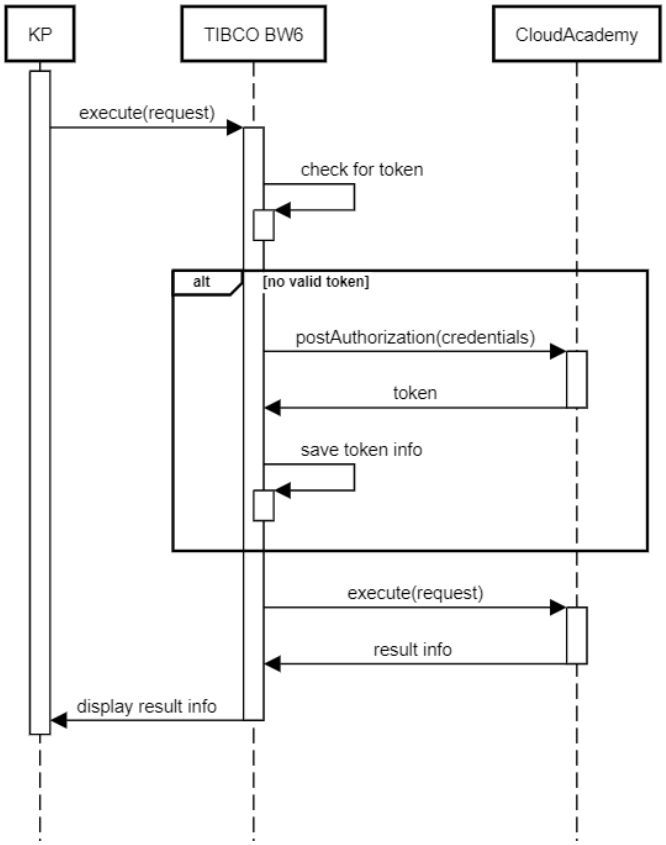


**Figura 4.1:** Diagramma di sequenza relativo al nostro sistema

## 4.2 Soluzione Tecnologica - TIBCO BusinessWorks

TIBCO è una software house statunitense leader del mercato nel settore dell'integrazione di sistemi. TIBCO ActiveMatrix BusinessWorks è una suite di prodotti di integrazione per applicazioni enterprise, web e mobili. Con questo software è possibile creare servizi e integrare applicazioni utilizzando un ambiente di sviluppo grafico ed interattivo di tipo model-driven.

La suite si compone di:

- *Design time*: un IDE grafico chiamato TIBCO BusinessStudio comune a tutte le versioni del software (on premises, BusinessWorks Container Edition e TIBCO Cloud Integration),
- *Run time*: una piattaforma ActiveMatrix BusinessWorks caratterizzata da una process engine potente ed una natura modulare basata su un framework OSGI,
- *Amministrazione*: un'interfaccia grafica di amministrazione TIBCO Enterprise Administrator e delle utility utilizzabili da linea di comando.

Questo strumento permette di implementare pattern di integrazione di tipo batch, sincroni e asincroni, in grado di soddisfare i requisiti in ambito Enterprise Application Integration, tipici di una soluzione di middleware, o Enterprise Service Bus.

Le soluzioni vengono sviluppate utilizzando l'ambiente di sviluppo integrato TIBCO Business Studio for BusinessWorks sotto forma di applicazioni. Una volta terminata la parte di sviluppo, il software consente di testare l'applicazione senza necessità di distribuirlo in un ambiente runtime. Successivamente, sempre in ambiente di design time, è possibile generare

l'artefatto distribuibile dell'applicazione sotto forma di un Enterprise Archive (EAR) che verrà distribuito sull'ambiente di runtime target.

Nella versione on premises gli artefatti possono essere distribuiti ed eseguiti nell'ambiente di runtime ActiveMatrix BusinessWorks tramite un'interfaccia di amministrazione come TIBCO Enterprise Administrator.

Il software, perciò, consente di gestire tutte le fasi del ciclo di vita dell'applicazione (Figura 4.2).

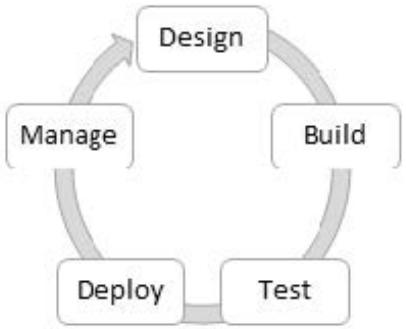


**Figura 4.2:** Ciclo di vita di un'applicazione

### 4.2.1 Architettura del software

ActiveMatrix BusinessWorks™ è costituito da un design-time in cui è possibile sviluppare applicazioni che implementano la logica aziendale, da un runtime in cui si eseguono le applicazioni e da un componente di amministrazione in cui si distribuiscono e si gestiscono le applicazioni nel runtime. Il runtime è un ecosistema di entità che possono essere collocate o distribuite. È possibile distribuire, monitorare e gestire le applicazioni utilizzando l'utility bwadmin e TIBCO Enterprise Administrator.

La Figura 4.3 fornisce una panoramica dei concetti chiave che si incontrano quando si lavora con il software.

#### 4.2.1.1 Componenti di Design-Time

TIBCO Business StudioTM è un IDE basato su Eclipse. Esso consente lo sviluppo di applicazioni con un approccio "low code", pur permettendo all'utente di scrivere codice personalizzato. Offre un workbench ricco di funzionalità con molteplici visualizzazioni. Il core del software è infatti un approccio allo sviluppo basato su modelli, supportato da un ampio set di palette per la progettazione dei processi. Esso consente inoltre, la cooperazione all'interno di un team, includendo strumenti di collaborazione e di versioning come Git e SVN.

L'unità fondamentale e distribuibile è l'applicazione BW. Ogni applicazione ha una struttura ben definita, definita costituita da:

- uno e un solo Application Module;
- zero o più Shared Modules;
- almeno un Component Process;
- zero o più Sub-Processes.

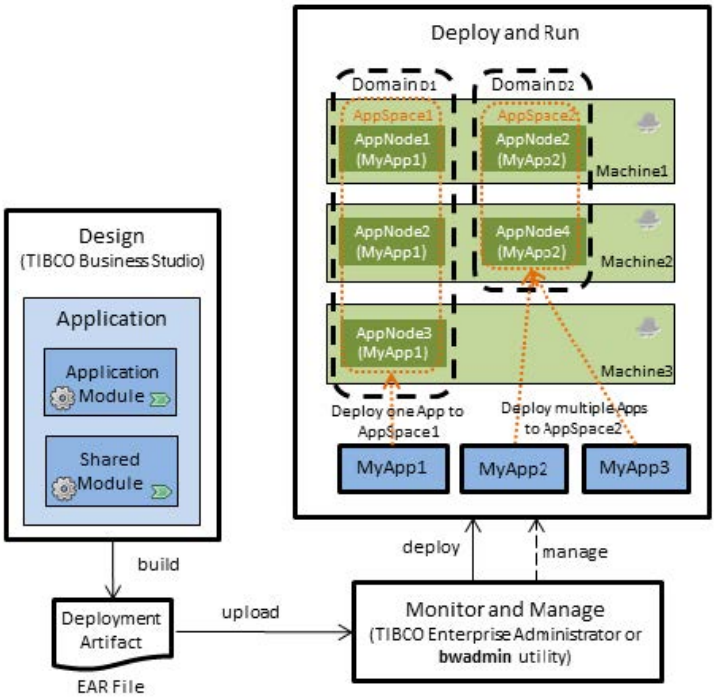


**Figura 4.3:** Componenti BusinessWorks

Ogni Module incapsula alcuni processi e risorse utilizzati, organizzandoli in 'packages'. La differenza tra gli Application Module e i Shared Module è la riutilizzabilità: gli Shared Module sono creati per processi e risorse che possono essere riutilizzati più volte da processi e moduli diversi, mentre gli Application Module non possono essere esposti al di fuori del modulo stesso e hanno un ambito limitato.

Nella Figura 4.4 vediamo la struttura di un'applicazione BW.

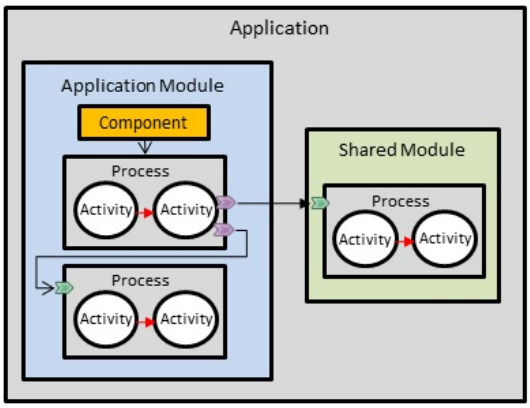


**Figura 4.4:** Struttura di un'applicazione BW

I processi sono pezzi di codice che implementano alcune funzionalità e logiche, disegnati utilizzando le Activity incluse nelle palettes (Figura 4.5) o codificati con codice personalizzato. Esistono tre tipi di processi:

- *Component process*: è un processo che include un'Activity che funge da start del processo. Non può essere invocato da altri processi nella stessa applicazione.

- *Subprocess*: può essere richiamato da ogni processo dello stesso modulo applicativo e non contiene un'attività di start. Deve includere una definizione dei parametri di input e della struttura di output attesi.
- *Parent Process*: è un processo che può chiamare un altro processo o un sottoprocesso.

Le Activity sono rappresentazioni di funzioni specifiche che hanno causato la costruzione di un processo e possono essere collegate tra loro per definire un flusso di processo specifico.

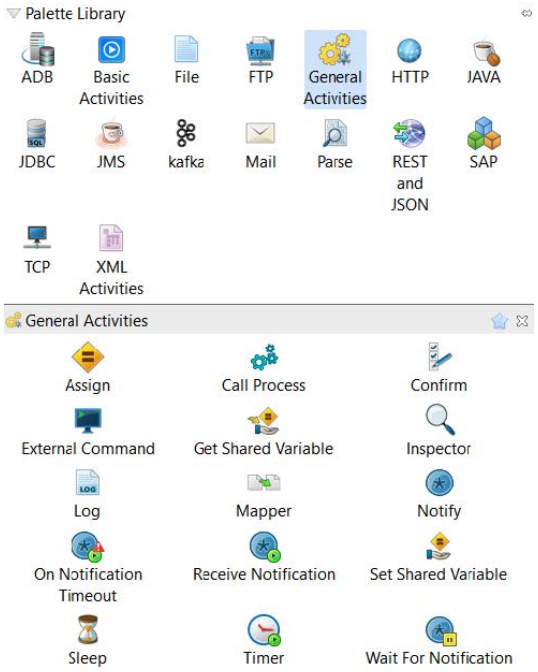


**Figura 4.5:** Palette and activity

Ogni Application Module può anche esporre servizi Web o API REST. Essi forniscono interfacce di esposizione necessarie per rendere il codice all'interno dell'applicazione raggiungibile attraverso la rete. Le interfacce SOAP sono definite attraverso un file WSDL, che può essere codificato manualmente o definito con una procedura guidata WSDL; esse sono esposte attraverso una risorsa HTTP Connector e un binding SOAP collegato a un Component Process.

Nella Figura 4.6 possiamo vedere un esempio di un servizio SOAP utilizzato all'interno di un'applicazione BusinessWorks.

I servizi RESTful, invece, vengono definiti attraverso un file JSON o una specifica procedura che guida l'utente nella loro definizione. Quindi, come per i servizi SOAP, un'origine HTTP Connector e un binding REST collegheranno il processo ed esporranno l'API. Una differenza rispetto ai servizi SOAP è che le API REST possono essere definite solo in applicazioni stateless.

Nella Figura 4.7 possiamo vedere un esempio di un servizio REST utilizzato all'interno di un'applicazione BusinessWorks.

Altre caratteristiche importanti di Business Studio sono le Module Property e i Profile, che consentono di parametrare il codice, definendo variabili e parametri che possono cambiare in base al profilo di runtime scelto, senza la necessità di modificare il codice.

Business Studio offre anche una funzione integrata di debug ed esecuzione per scopi di test che consente la risoluzione di problemi e la correzione di bug direttamente nell'ambiente

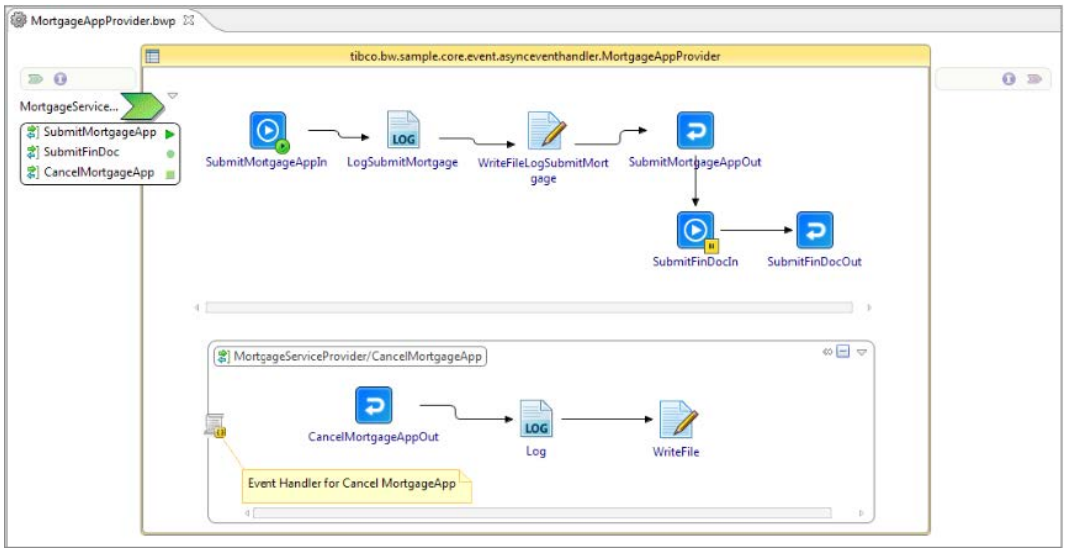


**Figura 4.6:** Esempio di servizio SOAP in BusinessWorks

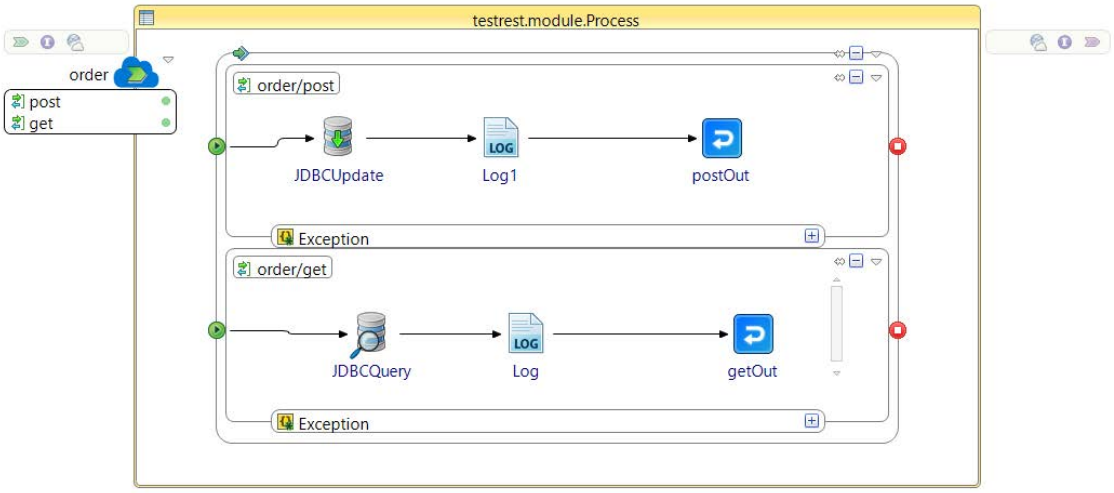


**Figura 4.7:** Esempio di servizio REST in BusinessWorks

di progettazione, fornendo all'utente informazioni utili sull'esecuzione, sia in modo grafico, ad alto livello, sia con viste dettagliate.

Una volta che l'applicazione è pronta, può essere confezionata in un file EAR e di essa si può effettuare il deployment in container, in TIBCO Cloud o in un'installazione on-premise dell'ambiente di runtime.

#### 4.2.1.2 Componenti di Run-Time

Runtime si riferisce all'AppNode e all'ActiveMatrix BusinessWorks Engine che ospitano ed eseguono le applicazioni ActiveMatrix BusinessWorks. Un AppNode, chiamato anche bwappnode, è un processo del sistema operativo che ospita ed esegue le applicazioni ActiveMatrix BusinessWorks. Un AppNode è costituito da due livelli chiave: il framework OSGI e ActiveMatrix BusinessWorks Engine. L'architettura di alto livello di un AppNode è mostrata in Figura 4.8.

Il livello Framework esegue le operazioni del ciclo di vita dell'applicazione, assicura che le dipendenze richieste dall'applicazione siano soddisfatte e interagisce con l'Administrator.

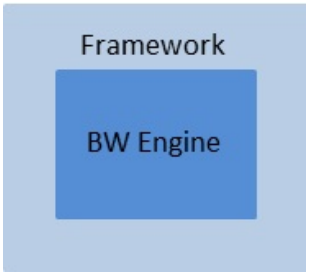


**Figura 4.8:** Architettura Application Node

Il livello Engine è responsabile dell'esecuzione dell'applicazione, è multithread e può eseguire più lavori per la stessa applicazione o per applicazioni diverse in modo simultaneo. In fase di esecuzione, un AppNode lancia il Framework per convalidare e identificare le dipendenze, dopo aver convalidato i moduli; a questo punto l'applicazione viene distribuita e l'ActiveMatrix BusinessWorks Engine avvia i processi sottostanti. Ogni AppNode è associato a un AppSpace, che rappresenta un gruppo di uno o più AppNode. Gli AppSpace sono contenuti in un dominio e al loro interno possono essere distribuite una o più applicazioni. L'esecuzione di un processo crea un ambito di esecuzione per le attività che fanno parte del processo; questo ambito è chiamato istanza di processo, caratterizzata da un id univoco, denominato "ProcessInstanceId".

L'esecuzione di un processo è innescata da vari eventi; ad esempio, gli eventi possono essere generati da un Timer che è programmato per attivarsi a intervalli di tempo specifici, o da modifiche che si verificano nel file system, o da messaggi inviati da un client su un protocollo specifico (ad esempio, HTTP, JMS, etc.), o, semplicemente, da messaggi inviati da altri processi.

L'ActiveMatrix BusinessWorks Engine è un motore multi-thread in grado di innescare l'esecuzione dello stesso processo più volte, in modo concomitante, una volta per ogni evento. Per ogni esecuzione, l'engine crea un'istanza di processo che fornisce un ambito di esecuzione per le attività che fanno parte del processo.

L'esecuzione di un processo di un componente è chiamata Job; ogni job è caratterizzato da un id univoco, denominato JobId. Quando la logica aziendale è distribuita su più processi, vengono create più istanze di processo che vengono eseguite in concomitanza con un determinato evento. Anche se si tratta di istanze di processo separate, esse lavorano insieme e possono essere eseguite come parte dello stesso Job. Un Job può generare più istanze di processo e può fornire il contesto di esecuzione per le attività che fanno parte di più processi. L'engine esegue sempre un lavoro in un unico thread. Tutte le istanze di processo che fanno parte dello stesso lavoro hanno lo stesso JobId. Un'istanza del Component process e tutte le sue istanze di Subprocess in linea sono considerate parte dello stesso Job. I Subprocess non in linea generano un nuovo thread dell'engine e vengono eseguiti su un Job diverso.

#### 4.2.1.3 Componenti di Amministrazione

Le applicazioni vengono distribuite in ambienti runtime e gestite tramite bwadmin utility; quest'ultimo può essere utilizzato anche per gestire e monitorare le applicazioni. TIBCO ActiveMatrix BusinessWorks fornisce un framework flessibile che consente di scalare l'ambiente di runtime in base alle esigenze. Il runtime offre l'opzione di eseguire l'ActiveMatrix BusinessWorks Engine in modo da ridurre il rischio di un singolo punto di guasto durante l'esecuzione di un'applicazione.

I principali componenti di Administration sono i seguenti:

- Un *Application Archive* è l'unità di distribuzione di un'applicazione generata in TIBCO Business Studio for BusinessWorks™.
- Un dominio è un gruppo logico che fornisce un ambiente isolato per le applicazioni e le loro risorse.
- Un *AppSpace* è un gruppo di uno o più AppNode; questi sono entità runtime che ospitano applicazioni ActiveMatrix BusinessWorks.
- Un *AppNode* è un'entità di runtime che ospita applicazioni.
- Il *bwagent* è un processo demone che viene eseguito su ogni installazione di ActiveMatrix BusinessWorks. Quando più installazioni su più macchine sono configurate come una rete, i bwagent interagiscono tra loro utilizzando un datastore e sincronizzano i dati del datastore con il file system locale.

Nella Figura 4.9 possiamo vedere una possibile architettura dell'Administration.

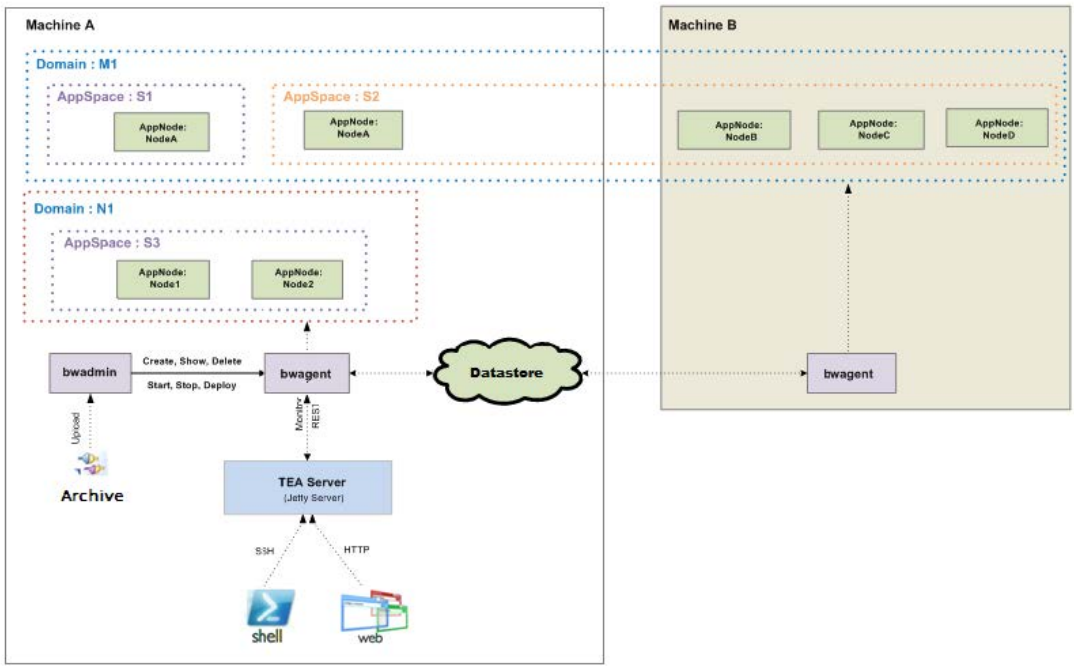


**Figura 4.9:** Architettura dell'Amministrazione

### 4.2.2 Target Platform

TIBCO BusinessWorks è una potente piattaforma di integrazione che offre la possibilità di progettare, sviluppare, distribuire e gestire processi aziendali complessi e integrazioni. La piattaforma è estremamente versatile e può essere distribuita in diversi ambienti, tra cui container, cloud e installazioni on-premise.

Di seguito illustriamo alcuni dettagli sulla distribuzione di BusinessWorks su ciascuna di queste piattaforme:

- *Container*: TIBCO BusinessWorks può essere distribuito in ambienti containerizzati utilizzando strumenti di orchestrazione di container come Kubernetes. La containerizzazione offre un'opzione di distribuzione leggera, scalabile e portatile per le applicazioni. Impacchettando il codice dell'applicazione e le dipendenze in un container, è possibile distribuirlo e gestirlo facilmente in ambienti diversi. TIBCO BusinessWorks può essere distribuito sia in architetture single-container che in architetture multi-container.

- *Cloud*: TIBCO BusinessWorks può essere distribuito su varie piattaforme cloud utilizzando opzioni di distribuzione basate sul cloud, come Amazon Web Services (AWS), Microsoft Azure e Google Cloud Platform (GCP). La distribuzione nel cloud offre un'opzione scalabile e conveniente per le organizzazioni che desiderano distribuire le proprie soluzioni di integrazione. TIBCO BusinessWorks può essere distribuito in ambienti Infrastructure as a Service (IaaS) e Platform as a Service (PaaS). TIBCO Cloud Integration è la soluzione iPaaS di TIBCO, nella quale la piattaforma cloud viene fornita direttamente dal software vendor come un servizio. Gli applicativi vengono rilasciati attraverso un portale web che consente, inoltre, la gestione ed il monitoraggio degli applicativi a runtime.

- *On-Premise*: TIBCO BusinessWorks può essere distribuito anche in installazioni on-premise. Queste ultime offrono alle organizzazioni un maggiore controllo sulle loro soluzioni di integrazione e possono essere ideali per le aziende con requisiti di sicurezza rigorosi. TIBCO BusinessWorks può essere distribuito sia in architetture a singolo nodo che in cluster, garantendo alta disponibilità e scalabilità per le soluzioni di integrazione mission-critical. TIBCO fornisce pacchetti di installazione per BusinessWorks che possono essere installati su vari sistemi operativi, tra cui Windows, Linux e Solaris.

Nel complesso, TIBCO BusinessWorks offre una piattaforma di integrazione versatile e potente, che può essere implementata in una varietà di ambienti e rappresenta la scelta ideale, che meglio soddisfa le esigenze e i requisiti, per le organizzazioni che desiderano sviluppare e gestire integrazioni complesse.

# CAPITOLO 5

# Implementazione del sistema

*Questo capitolo tratterà l'implementazione della soluzione progettata e spiegata nel capitolo precedente. In un primo momento verranno descritte le componenti comuni alle due applicazioni che poi verranno spiegate nel dettaglio successivamente.*

## 5.1 Aspetti comuni delle applicazioni sviluppate

Prima di descrivere lo sviluppo delle applicazioni, si è deciso di standardizzare alcuni loro aspetti comuni, come la struttura dell'applicazione, la gestione del tracciamento dei log e la gestione delle eccezioni.

Per favorire la gestione ed il riutilizzo di queste componenti comuni alle applicazioni in oggetto, è stato creato un modulo condiviso, o *shared module*. Questa tipologia di modulo rappresenta una risorsa riutilizzabile, dotata di un nome identificativo, una versione ed un pacchetto che vanno a comporre, insieme ad altre componenti specifiche, un'applicazione.

Lo *shared module* può essere referenziato da altri che fanno parte della stessa applicazione. Un modulo shared non può essere distribuito da solo; deve essere incluso come parte di un modulo applicativo, o *application module*. Esso esporta le sue funzionalità (processi, risorse condivise e spazi dei nomi dello schema) ai moduli dell'applicazione o ad altri moduli shared. Ciò significa che esiste la possibilità che altri moduli del sistema dipendano da un modulo shared.

### 5.1.1 Struttura a strati

Ogni applicazione è stata strutturata in tre livelli per meglio disaccoppiare interfacce e logica operativa. Più specificamente, i tre livelli sono:

- *Component process*: include tutti i processi BW di ingresso e di arrivo dell'applicazione. Costituisce, di fatto, lo strato che include l'interfaccia inbound dell'applicazione implementandone i suoi processi componenti BW.
- *Parent process*: include tutti i sotto-processi BW che implementano le logiche, funzionali o di business, che l'applicazione deve gestire. In questo livello vengono, ad esempio, gestite logiche di orchestrazione tra più sistemi. Nessuna interfaccia in entrata o in uscita è esposta da questo strato applicativo.

- *Subprocess*: è responsabile dell'integrazione con i sistemi di back-end che l'applicazione deve integrare. Implementa, ad esempio, chiamate ad API REST su canale HTTP, transazioni JDBC verso database, o invocazioni a servizi SOAP, etc. Questo livello applicativo gestisce, inoltre, le logiche di trasformazione e aggregazione del dato, rappresentando il livello di adattatore verso i sistemi di back-end.

Questo approccio favorisce anche la modularità e la riusabilità del codice. Alcuni sotto-processi delle operazioni e del livello di integrazione possono essere utilizzati da più di un processo, ottimizzando lo sviluppo del codice. Inoltre, la suddivisione dei compiti di ogni livello applicativo favorisce le operazioni di manutenzione del codice e la gestione di nuove evolutive o modifiche.

### 5.1.2 Gestione dei log

È stato deciso di adottare un sistema di logging comune a tutte le applicazioni, per effettuare il monitoraggio e l'analisi in modo più efficiente. A tal proposito sono stati implementati alcuni sotto-processi ad-hoc all'interno del modulo shared BW che verrà poi importato in ogni applicazione, fornendo funzionalità di logging standard.

Per prima cosa, è stata identificata una struttura dati comune che raccolga le informazioni utili al processo di tracciamento. Tale struttura dati è stata convertita nello schema XSD (XML Schema Definition) la cui rappresentazione grafica è riportata nella Figura 5.1.

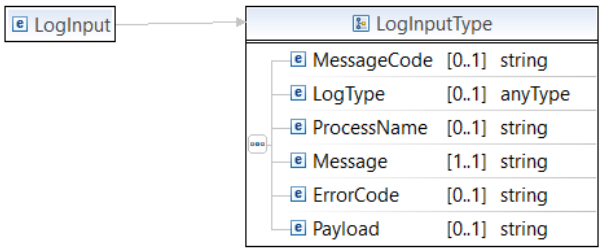


**Figura 5.1:** Struttura XSD dei log

Per la gestione dei log sono stati creati due processi, ovvero SetLog e Log.

Il processo *SetLog*, visibile nella Figura 5.2, è stato creato per avviare il meccanismo di logging inizializzando una variabile di log condivisa sullo stesso job applicativo (job shared variable). Tale variabile avrà come valori assegnati le chiavi di tracciamento che dovranno essere uguali in ogni messaggio di log della stessa applicazione/funzionalità.

Una volta "settate", queste variabili verranno automaticamente copiate in ogni messaggio di log del processo.

Il processo *Log*, visibile nella Figura 5.3, unisce le informazioni di testata, recuperando innanzitutto le informazioni salvate nella variabile condivisa impostata dal processo *SetLog*, alle informazioni inserite in input al sotto-processo stesso nella singola richiesta di tracciamento. Successivamente converte tali informazioni nel formato desiderato, in questo caso in formato JSON. Infine, registra effettivamente le informazioni richieste.

### 5.1.3 Gestione delle eccezioni

Per la gestione delle eccezioni è stata utilizzata la gestione predefinita eseguibile tramite BusinessWorks, implementata utilizzando uno schema creato appositamente per l'applicazione in esame in modo da rendere più facile la lettura delle eccezioni. A tale scopo, quindi, è stato creato un XSD *CommonException* (Figura 5.4).

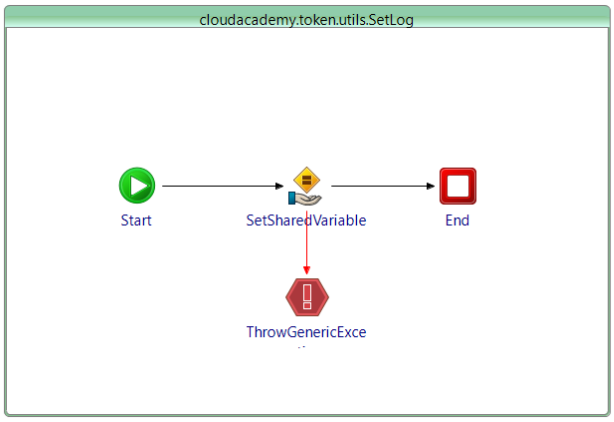


**Figura 5.2:** Processo *setLog*

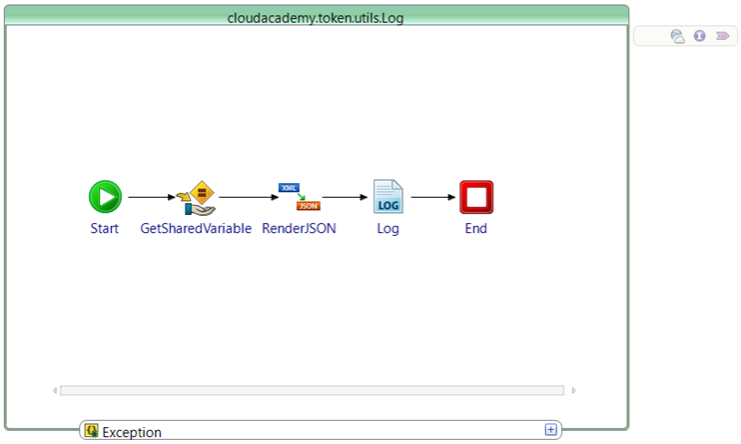


**Figura 5.3:** Processo *Log*

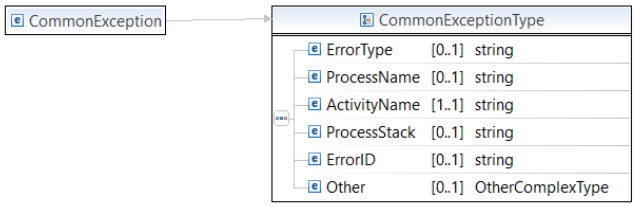


**Figura 5.4:** XSD *CommonException*

Esso verrà utilizzato come schema di riferimento per la gestione dei fault in tutti i processi del sistema. Alcuni codici sono stati standardizzati per una migliore e più rapida identificazione della categoria di errore in base al modello: TIB-BW-"Palette"-"codice" (ad esempio TIB-BW-JSON-0001).

### 5.1.4 API Token

Per gestire l'accesso ai servizi Cloud Academy è necessario generare un token secondo lo standard OAuth2 con flusso client credential. A tal proposito è stato implementato un processo denominato *APIToken* inserito nel modulo Shared dell'applicazione, così da poter permettere l'utilizzo a tutte le applicazioni sviluppate. Come possiamo vedere nella Figura

5.5, il processo ha come punto di partenza *SendHTTPRequest*, activity asincrona che invia una richiesta HTTP ed attende una risposta.

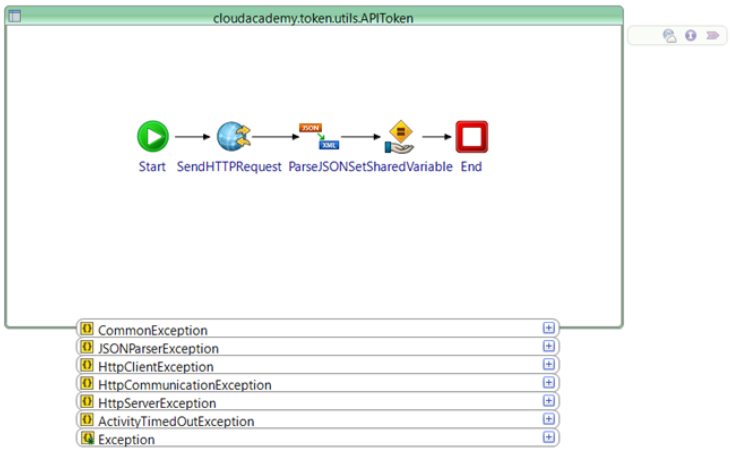


**Figura 5.5:** Processo di integrazione APIToken

Nel caso in esame viene inviata una richiesta al server di Cloud Academy, fornendo le credenziali utili al fine della generazione del token. Per permettere la request http è necessario creare una risorsa HTTP Client, che rappresenta una connessione HTTP in uscita. Nella Figura 5.6 possiamo vedere la definizione della risorsa HTTP Client utilizzata per la generazione del token.

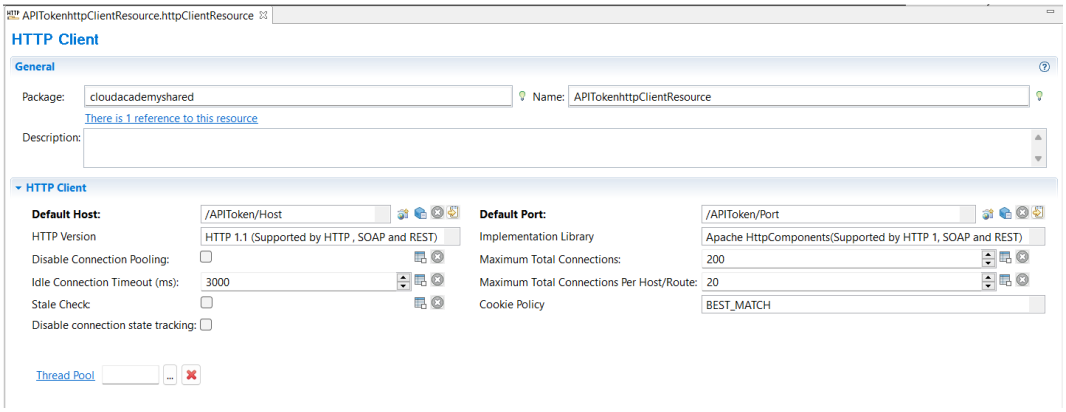


**Figura 5.6:** Risorsa HTTP Client per l'APIToken

Nella risorsa sono state definiti i valori di Host e Porta relativi al server Web di Cloud Academy, server da interrogare per la generazione del token da utilizzare per l'interrogazione delle API messe a disposizione dalla piattaforma stessa.

Dopo aver creato la risorsa HTTP necessaria si potrà utilizzarla nell'activity *SendHTTPRequest* impiegata nel processo APIToken; nella Figura 5.7 possiamo vedere come viene richiamata la risorsa all'interno dell'activity.

Nella Figura 5.8 possiamo vedere il mapping relativo all'input dell'activity *SendHTTPRequest* precedentemente descritta.

Come possiamo vedere, per effettuare la chiamata sono statecreate precedentemente delle module properties utilizzabili all'interno del processo; in questo caso esse sono:

- *URI_Token*: rappresenta il path del server a cui inviare la request.
- *client_id* e *client secret*: rappresentano le credenziali di accesso richieste per la generazione del token.

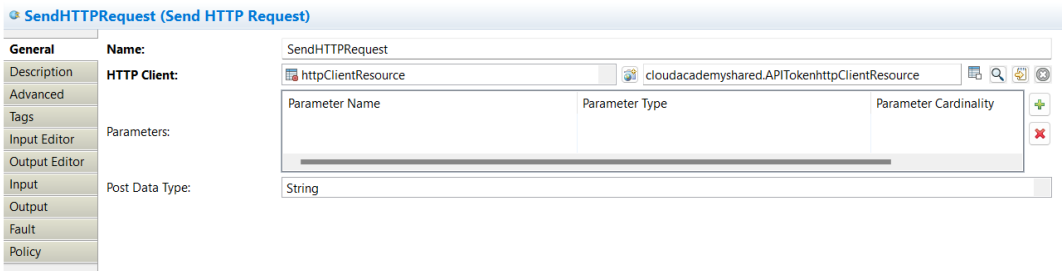


**Figura 5.7:** Configurazione dell'activity *SendHTTPRequest*

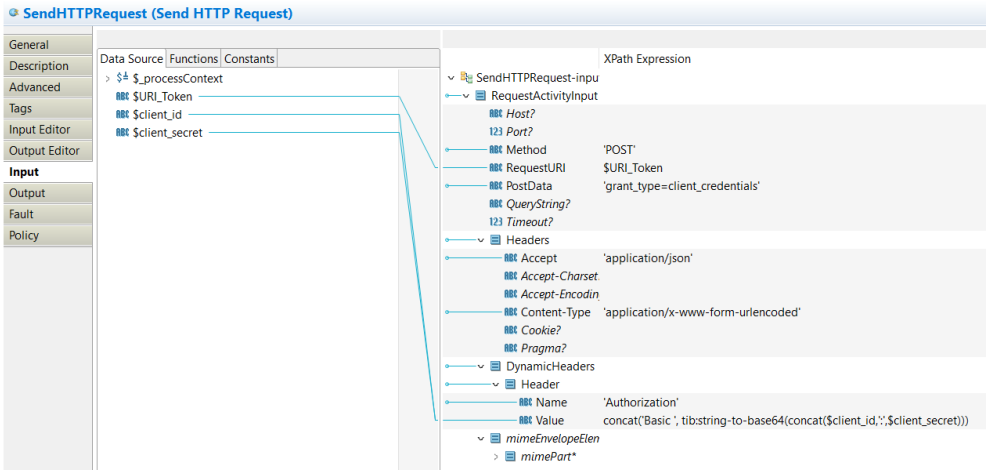


**Figura 5.8:** Mapping dell'activity *SendHTTPRequest*

Nella Figura 5.9 vediamo la definizione delle properties precedentemente descritte.

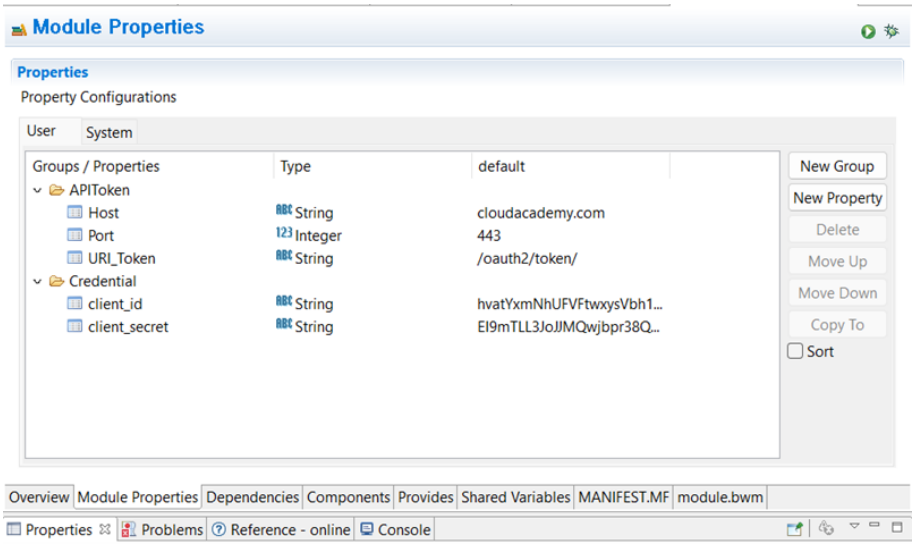


**Figura 5.9:** Module properties del modulo shared

Dopo aver interrogato il server web CloudAcademy, verrà restituito il token da utilizzare per richiamare i vari servizi messi a disposizione. In particolare, verrà effettuato un parse della response ricevuta, il dato ricevuto in formato JSON verrà trasformato in formato XML, e poi memorizzato in una variabile condivisa di modulo (in questo caso una module shared variable) che sarà, poi, utilizzata per richiamare i servizi necessari.

Utilizzando le variabili condivise, è possibile condividere i dati tra le istanze di processo associate a un modulo o a un job. Un'istanza di processo può leggere o aggiornare i dati memorizzati in una variabile condivisa; tale dati saranno accessibili alle altre istanze di processo di un modulo o di un job. Esistono due tipi di variabili condivise, ovvero le variabili condivise del modulo e le variabili condivise del job. Sia le variabili condivise del modulo che quelle del job sono definite a livello di modulo e sono accessibili in un processo tramite le attività Set *Shared Variable* e *Get Shared Variable*.

- *Le variabili condivise di modulo* sono utilizzate per condividere lo stato a livello di modulo e sono visibili a tutte le istanze di processo create dai processi che si trovano all'interno di un modulo. Una volta aggiornato il valore, quest'ultimo è disponibile per tutte le istanze create dai processi che si trovano all'interno del modulo.
- *Le variabili condivise del job* sono utilizzate per condividere lo stato a livello di job, ovvero per la durata di un job. Per ogni nuovo job viene creata una copia della variabile condivisa del job, accessibile a tutte le istanze di processo associate a quest'ultimo.

## 5.2 Applicazioni del sistema

In questa sezione verranno riportate, come della soluzione, le due applicazioni sviluppate per far fronte ai requisiti funzionali trattati precedentemente.

### 5.2.1 Team Manager

L'applicazione denominata *Team Manager* è stata sviluppata per permettere di integrare i sistemi Key Partner con le API messe a disposizione da Cloud Academy per la gestione dei team. Essa rende possibile l'aggiunta di un membro ad un team, la rimozione di un membro da un team e la visualizzazione dei membri di un team. L'applicazione Team Manager prevede una struttura a strati. Essa è, quindi, formata dai processi descritti nel seguito.

#### 5.2.1.1 Processo main Team Manager

All'interno di questo processo (Figura 5.10) viene esposto un servizio RESTful (o risorsa REST) che implementa il metodo HTTP POST. All'interno di questo processo, oltre ad essere presente la gestione del tracciamento delle informazioni di log, viene richiamato il processo core dell'applicazione.

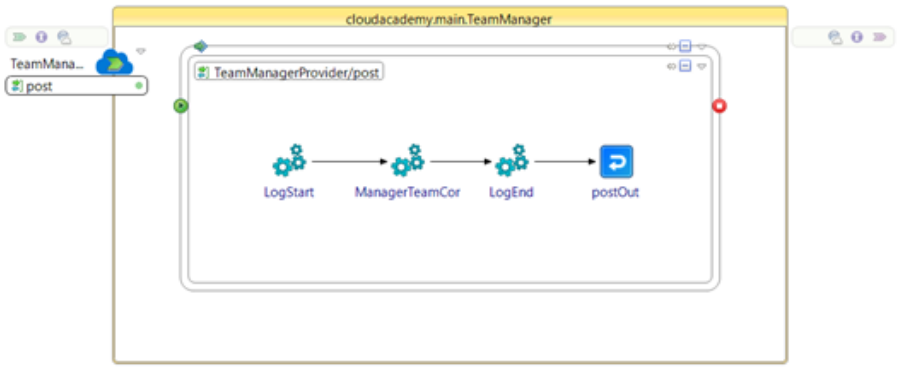


**Figura 5.10:** Architettura del processo *Team Manager*

Nella definizione della risorsa REST (Figura 5.11) è necessario indicare quali strutture dati verranno utilizzate per la request e la response. Per fare ciò, indipendentemente dal formato atteso (in questo caso JSON) è necessario referenziare le strutture dati contenute in schema XSD dedicati, creati in precedenza (Figure 5.12 e 5.13).

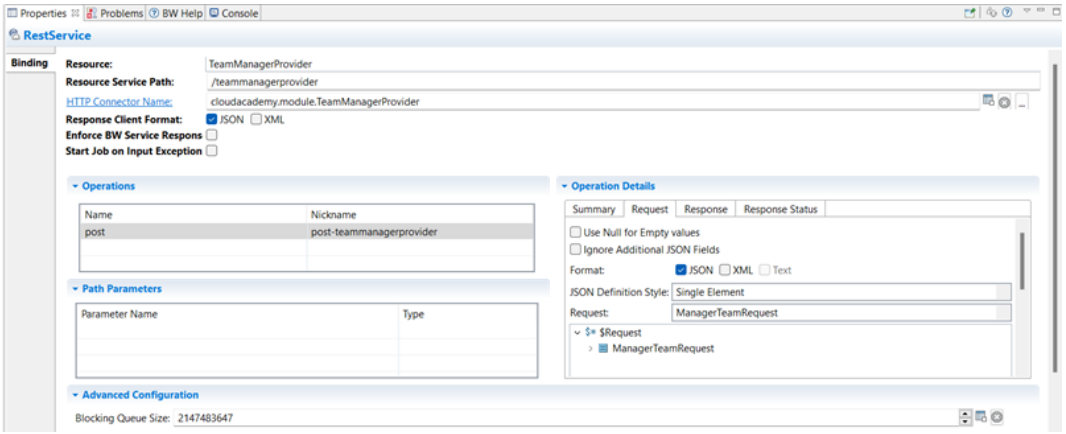


**Figura 5.11:** Definizione del servizio REST

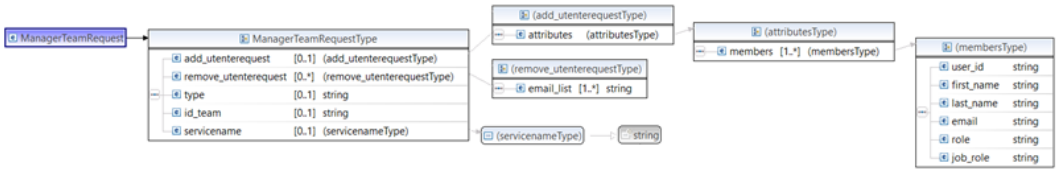


**Figura 5.12:** XSD schema del *ManagerTeamRequest*

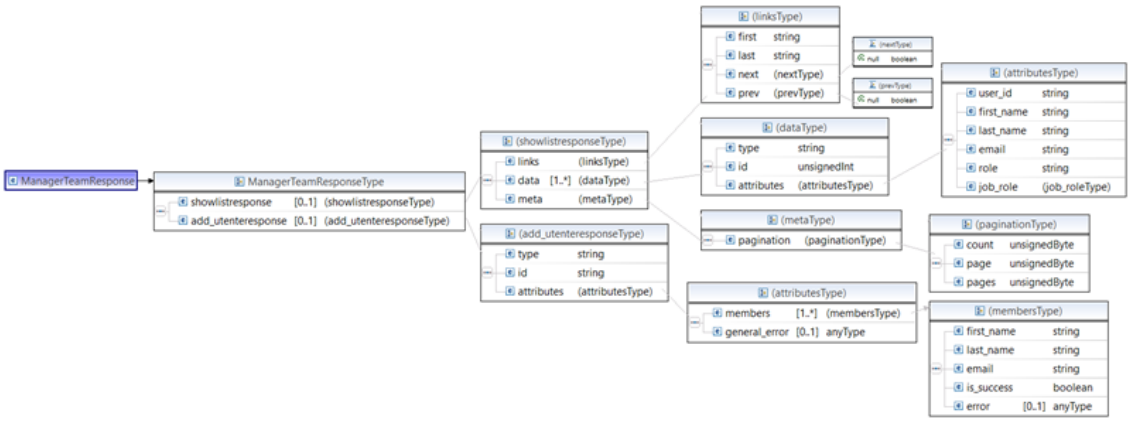


**Figura 5.13:** XSD schema del *ManagerTeamResponse*

La request al servizio RESTful esposto verrà "mappata" in input al servizio core dell'applicazione; nella Figura 5.14 possiamo vedere come è stato definito il mapping.

Per esporre un servizio su canale HTTP all'interno di un'applicazione TIBCO BW, è necessario specificare le informazioni di connessione in un apposito oggetto, che prende il nome di *HTTP Connector Shared Resource* (mostrato in Figura 5.15).

Questa componente descrive le caratteristiche della connessione utilizzata per ricevere le richieste inbound sul canale HTTP. Essa prevede una serie di parametrizzazioni possibili che vanno ad agire sulla configurazione del canale di trasporto HTTP (ad esempio parametri di timeout, queue size, etc.). In particolare, su questa risorsa vengono specificati l'hostname e

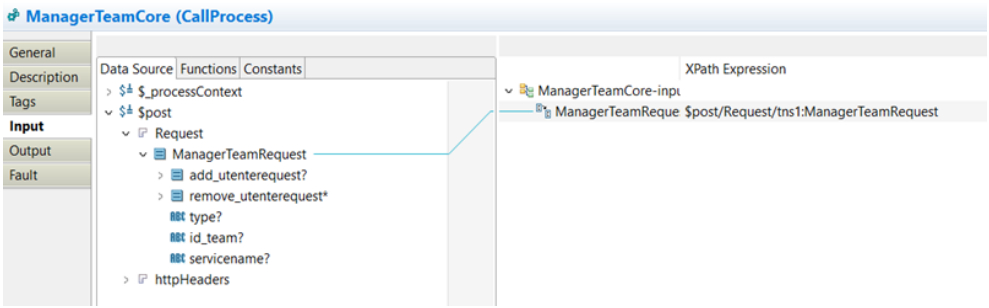


**Figura 5.14:** Mapping request nel processo core

la porta su cui il servizio sarà in ascolto. Per consentire una configurazione a runtime delle parametrizzazioni di cui sopra, è possibile introdurre delle variabili globali o proprietà di modulo (dette "Module Properties" Figura 5.16), configurate a design-time e riutilizzate a runtime dai diversi oggetti contenuti all'interno dell'applicazione.

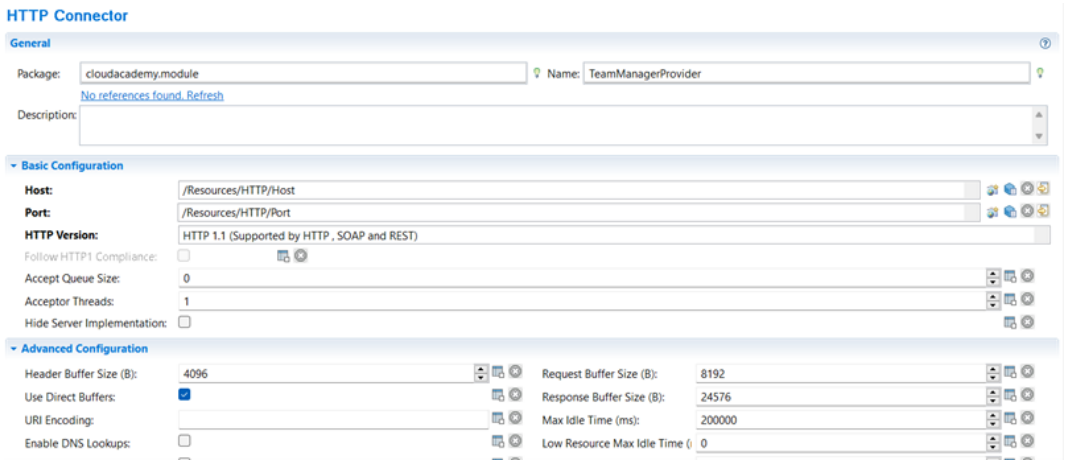


**Figura 5.15:** Definzione della risorsa *HTTP Connector*

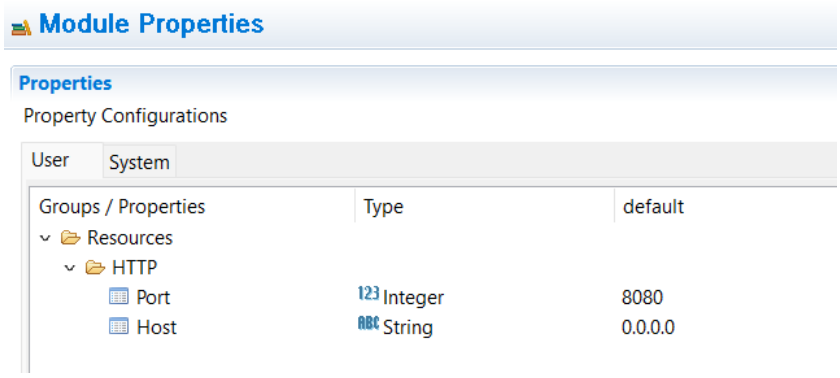


**Figura 5.16:** Module Properties del processo

### 5.2.1.2 Processo Team Manager Core

Questo processo (Figura 5.17) al suo interno richiama il processo *APIToken*, visto precedentemente, utilizzato per la generazione del token. Nell'input dell'activity *Mapper* (Figura 5.18) viene effettuata una logica sul campo "servicename". Ricevuto nelle request, attraverso

il quale viene selezionata la funzionalità che si desidera richiamare tra quelle esposte da Cloud Academy.

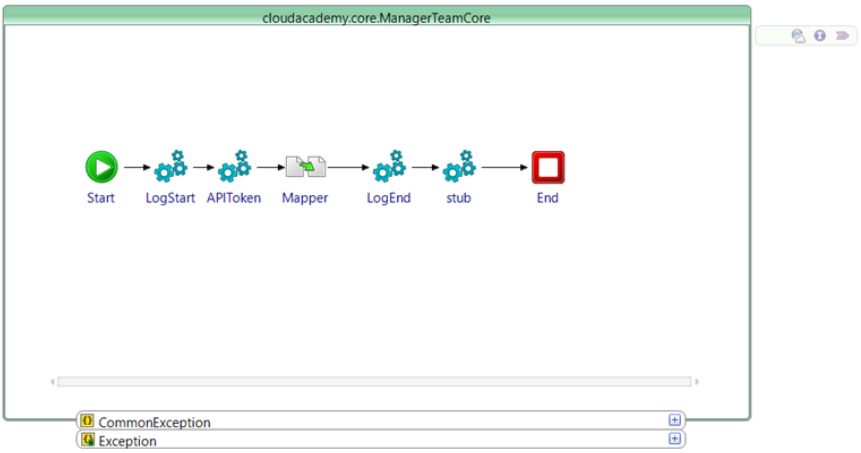


**Figura 5.17:** Processo *Team Manager Core*

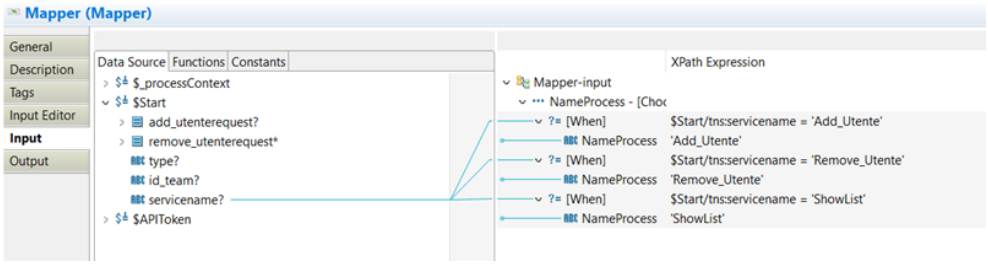


**Figura 5.18:** Mapping dell'activity *Mapper*

Da qui verrà innescata la chiamata al sotto processo specifico, che permette di effettuare l'interrogazione dell'API di Cloud Academy.

#### 5.2.1.3 Sotto processi (Add_Utente, Remove_Utente e ShowList)

All'interno dei sottoprocessi vengono richiamate le API messe a disposizione da Cloud Academy attraverso le activity *InvokeRESTAPI* e *SendHTTPRequest*, in base al servizio che si vuole richiamare, inviando una request contenente le informazioni necessarie al servizio da utilizzare. Entrambe le activity permettono di invocare servizi web e di ricevere risposte dal fornitore del servizio.

Nel caso del sottoprocesso *Add_Utente* (Figura 5.19) vengono derivate dalla request, proveniente dal servizio *main*, le informazioni necessarie per effettuare l'aggiunta di un membro al team di riferimento.

Nella Figura 5.20 possiamo vedere il mapping relativo della request al servizio in esame.

Nell'input dell'activity *InvokeRESTAPI* (Figura 5.21) possiamo vedere come viene "mappato", dalla request ricevuta, l'id del team nel quale effettuare l'aggiunta del membro all'interno del *ResourcePath* utilizzato dall'activity per effettuare la chiamata al servizio. All'interno del *MessageBody* viene mappato, invece, il contenuto della request ricevuta.

La response ottenuta dal servizio verrà restituita nella response del servizio all'interno del processo *main*; nella Figura 5.22 possiamo vedere il mapping dell'output.

La response così sarà visibile all'utente che chiama il servizio RESTful esposto dall'applicazione *Team Manager*.

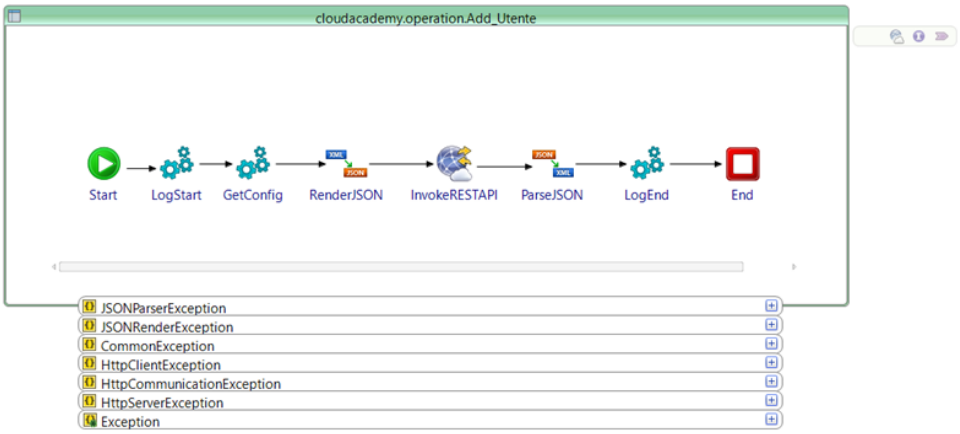


**Figura 5.19:** Processo *Add_Utente*

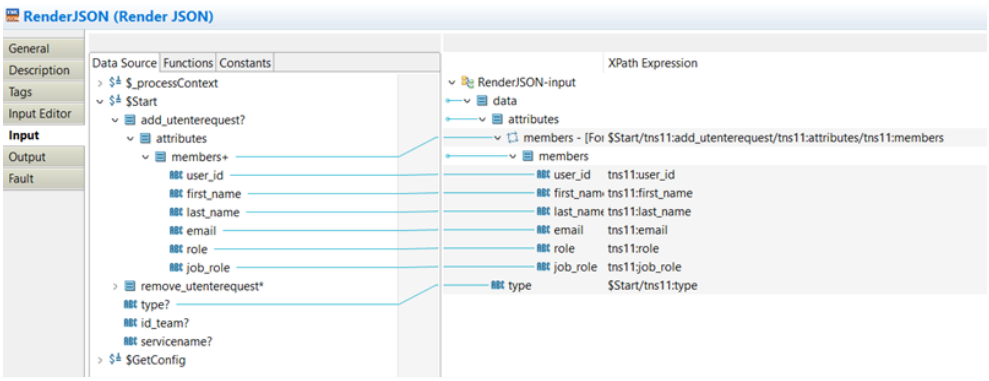


**Figura 5.20:** Mapping request service *Add_Utente*

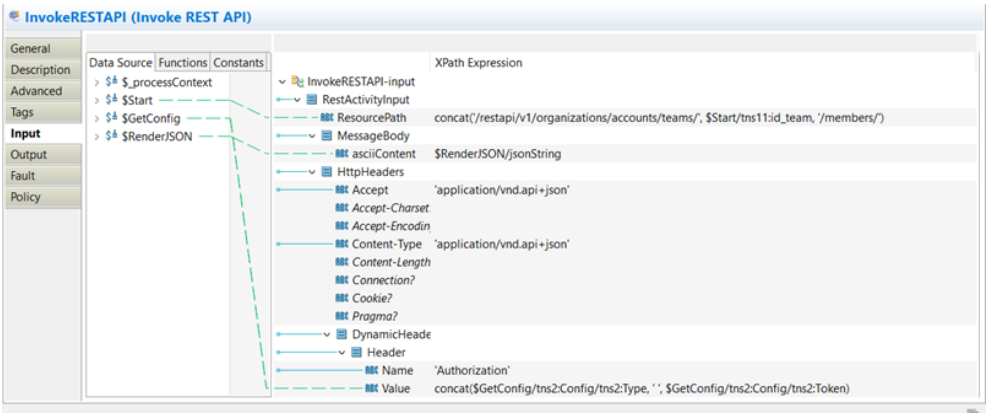


**Figura 5.21:** Mapping dell'activity *RESTAPI*

Allo stesso modo sono stati sviluppati gli altri due sottoprocessi *Remove_Utente* (Figura 5.23) e *ShowList* (Figura 5.24), che, rispettivamente, permettono all'utente di rimuovere un membro da un team specifico e di visualizzare la lista di tutti i membri di un team.

Questi sottoprocessi variano dal processo precedentemente descritto nella request che l'utente fa richiamando il servizio RESTful esposto dall'applicazione. Nel caso del sottoprocesso *Remove_Utente* l'utente dovrà specificare nella request l'indirizzo email dell'utente da rimuovere e l'id del team dal quale deve essere rimosso, inserito successivamente all'interno del *ResourcePath* utilizzato dall'activity per effettuare la chiamata al servizio. Nella Figura

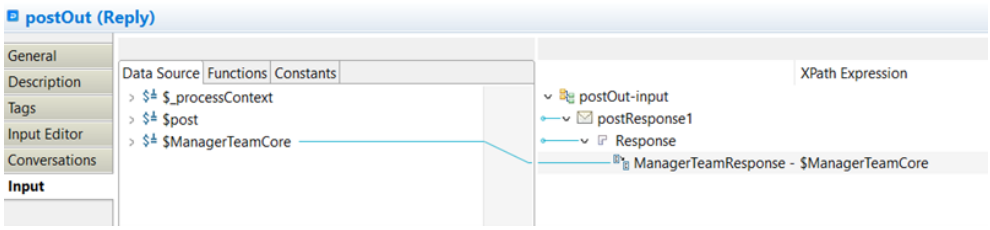


**Figura 5.22:** Mapping response del processo *core*

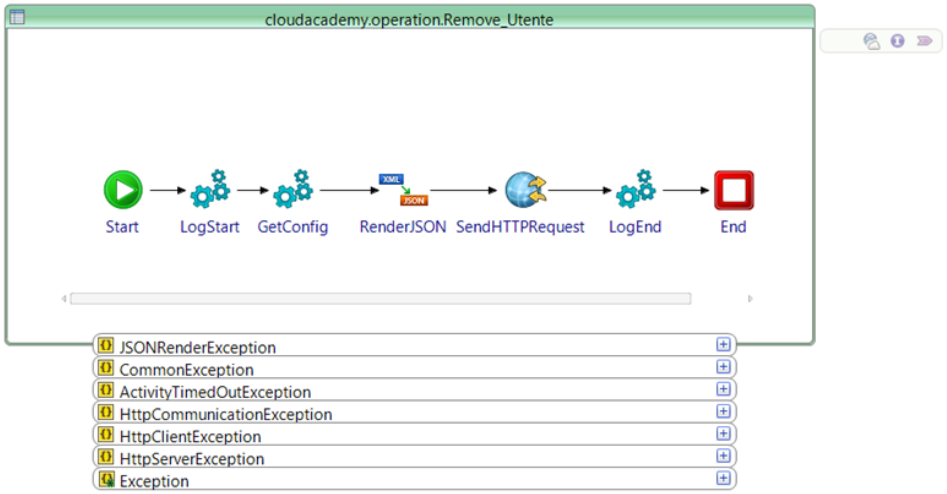


**Figura 5.23:** Processo *Remove_Utente*

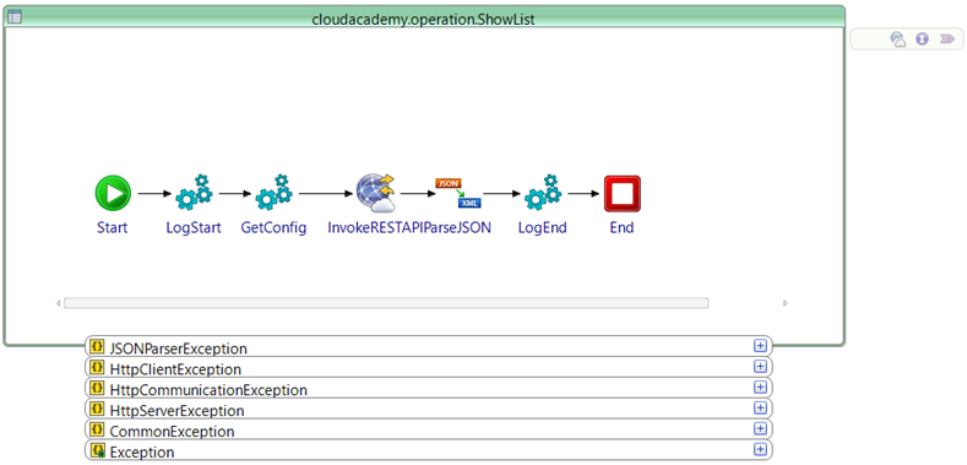


**Figura 5.24:** Processo *ShowList*

5.25 possiamo vedere il mapping della request con la quale viene effettuata la chiamata al servizio.

Nel caso del sottoprocesso *ShowList* l'utente dovrà specificare nella request l'id del team del quale vuole visualizzare la lista dei membri. Nella Figura 5.26 possiamo vedere il mapping della request con la quale viene effettuata la chiamata al servizio.

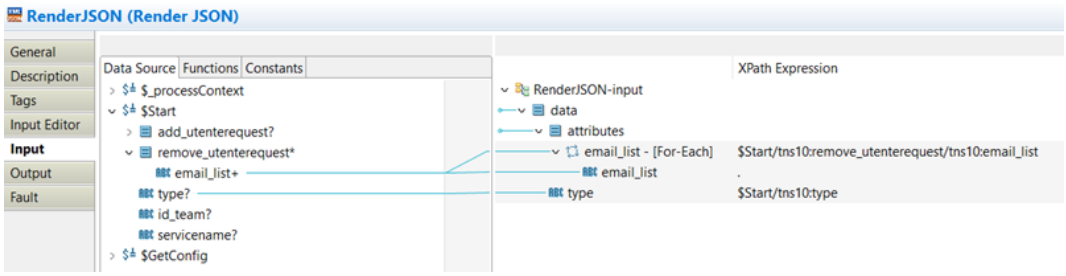


**Figura 5.25:** Mapping request servizio *Remove_Utente*

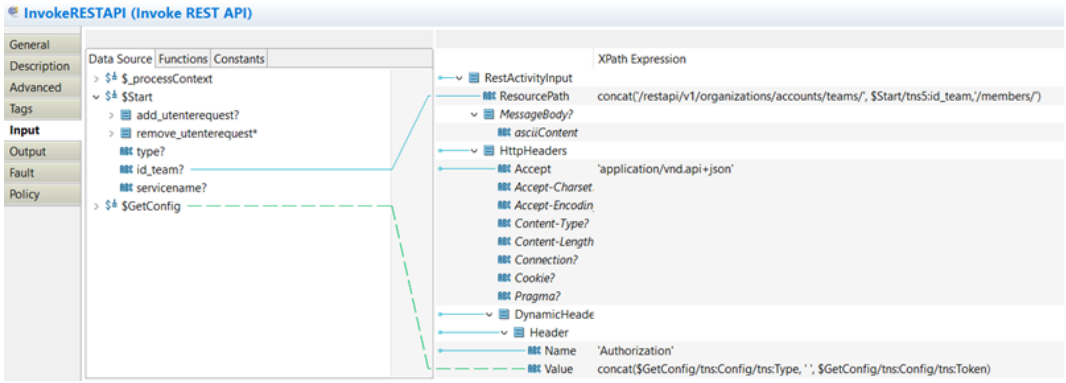


**Figura 5.26:** Mapping request servizio *ShowList*

### 5.2.2 Progress Manager

L'applicazione denominata Progress Manager è stata sviluppata per permettere di integrare i sistemi Key Partner con le API messe a disposizione da Cloud Academy per il monitoraggio del progresso della formazione dei membri dei vari team. Essa rende possibile monitorare il progresso dei membri di tutta l'azienda nonchè quello di un singolo membro specifico. L'applicazione Progress Manager prevede una struttura a strati, quindi, è formata da:

#### 5.2.2.1 Processo main Progress Manager

All'interno di questo processo (Figura 5.27) viene esposta una risorsa REST che prevede l'implementazione del metodo POST, al cui interno viene richiamato il processo core dell'applicazione.

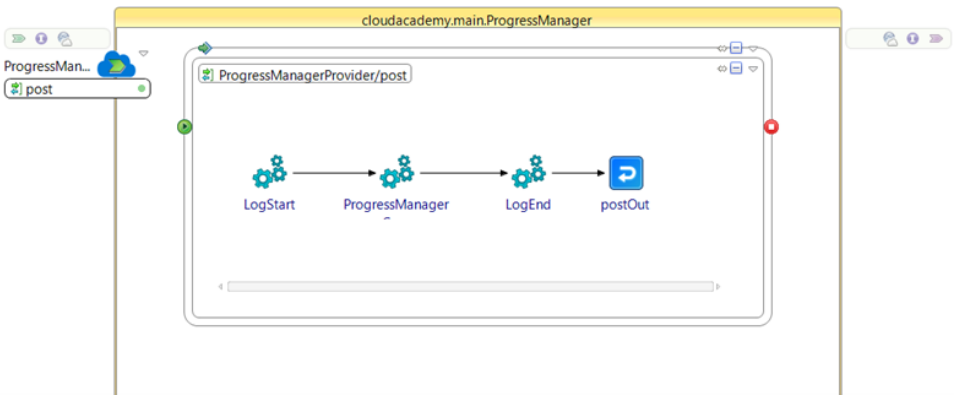


**Figura 5.27:** Architettura del processo *Progress Manager*

Nella definizione del servizio RESTful (Figura 5.28) vengono specificati lo schema di request e di response del servizio; a tal proposito è stato creato l'XSD di riferimento (Figure 5.29 e 5.30).

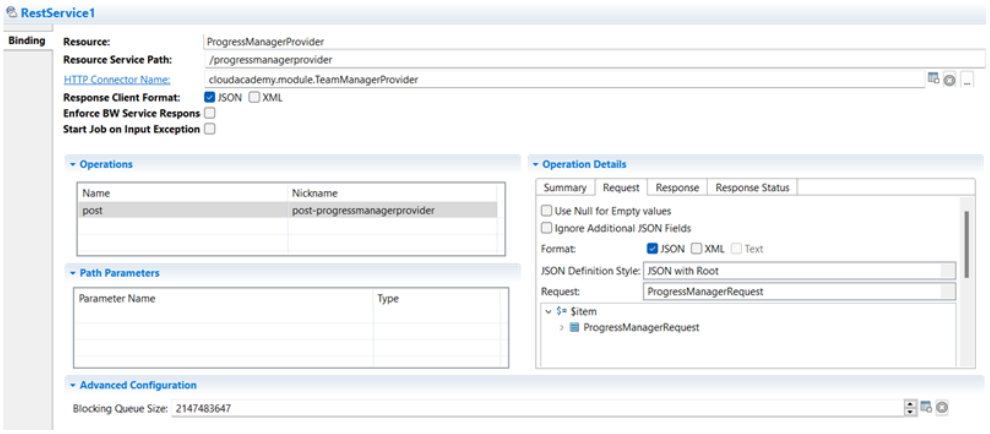


**Figura 5.28:** Definizione del servizio REST

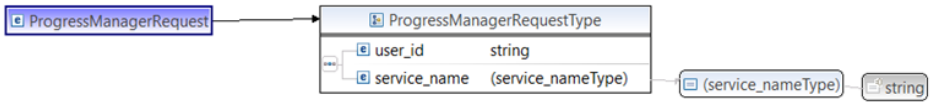


**Figura 5.29:** XSD schema di *ProgressManagerRequest*

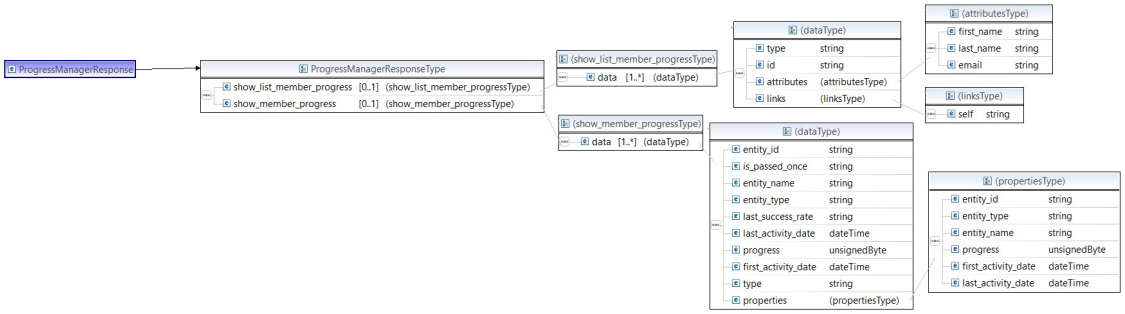


**Figura 5.30:** XSD schema di *ProgressManagerResponse*

La request alla API REST esposta verrà "mappata" in input al servizio core dell'applicazione; nella Figura 5.31 possiamo vedere come è stato definito il mapping.

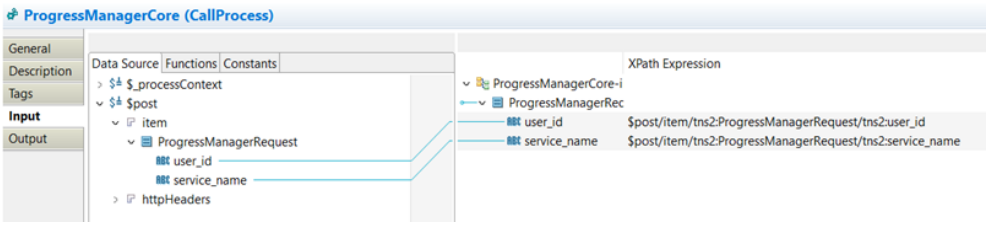


**Figura 5.31:** Mapping request nel processo core

Per esporre il servizio RESTful è necessaria una risorsa HTTP Connector; in questo caso viene utilizzata la risorsa creata per l'applicazione precedentemente descritta.

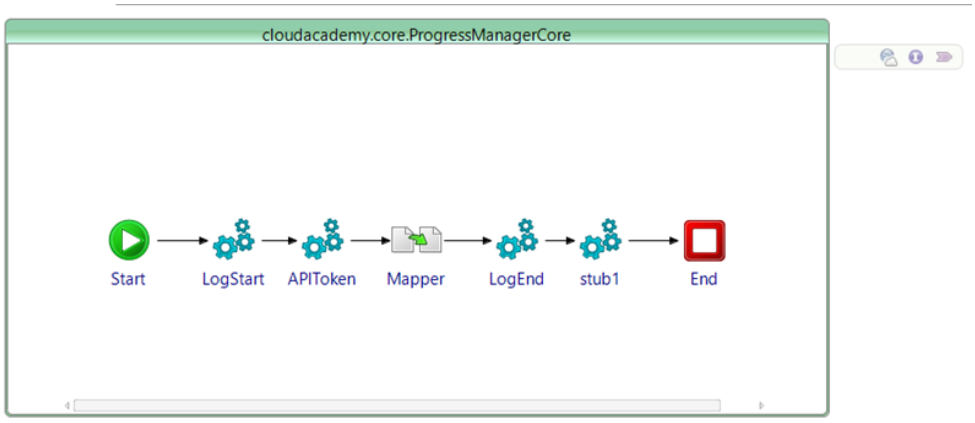


**Figura 5.32:** Architettura del processo core

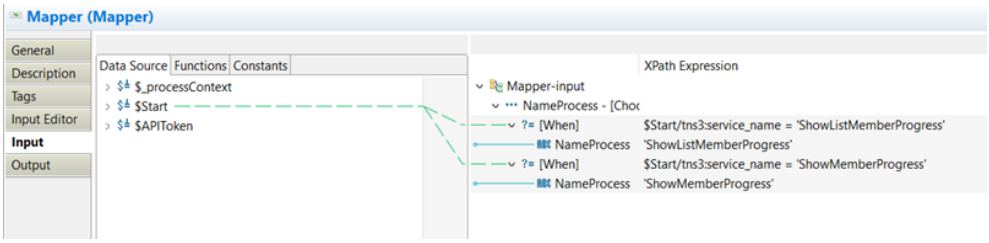


**Figura 5.33:** Logica dell'activity *Mapper*

Da qui verrà innescata la chiamata al sotto processo specifico, che permette di effettuare l'interrogazione dell'API di Cloud Academy.

#### 5.2.2.2 Sotto$_p$*rocessi*(*ShowListMemberProgresseShowMemberProgress*)

All'interno dei sottoprocessi vengono richiamate le API messe a disposizione da Cloud Academy attraverso l'activity *InvokeRESTAPI*, inviando una request contenente le informazioni necessarie al servizio da utilizzare.

L'architettura dei due sottoprocessi è la stessa dei sottoprocessi descritti precedentemente; in questo caso, però, il sottoprocesso *ShowListMemberProgress* (Figura 5.34) permetterà di visualizzare il progresso della formazione di tutti i membri dell'azienda, mentre il sottoprocesso ShowMemberProgress (Figura 5.35) permetterà di visualizzare il progresso della formazione di un singolo membro specifico.

Questi sottoprocessi variano dai processi precedentemente descritti nella request che l'utente fa richiamando il servizio RESTful esposto dall'applicazione. Nel caso del sottoprocesso *ShowListMemberProgress* l'utente dovrà specificare nella request solo il nome del servizio in questione per permettere la chiamata.

Nel caso del sottoprocesso *ShowMemberProgress* l'utente dovrà specificare nella request l'id del membro del quale vuole visualizzare il progresso della formazione, che sarà inserito, successivamente, all'interno del ResourcePath utilizzato dall'activity per effettuare la chiamata al servizio. Nella Figura 5.36 possiamo vedere il mapping della request con la quale viene effettuata la chiamata al servizio.

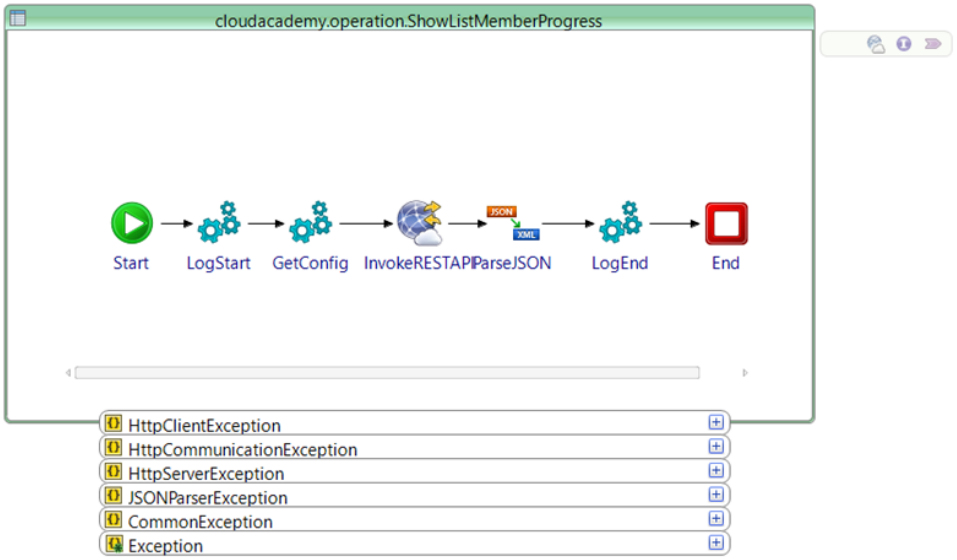


**Figura 5.34:** Architettura processo *ShowListMemberProgress*

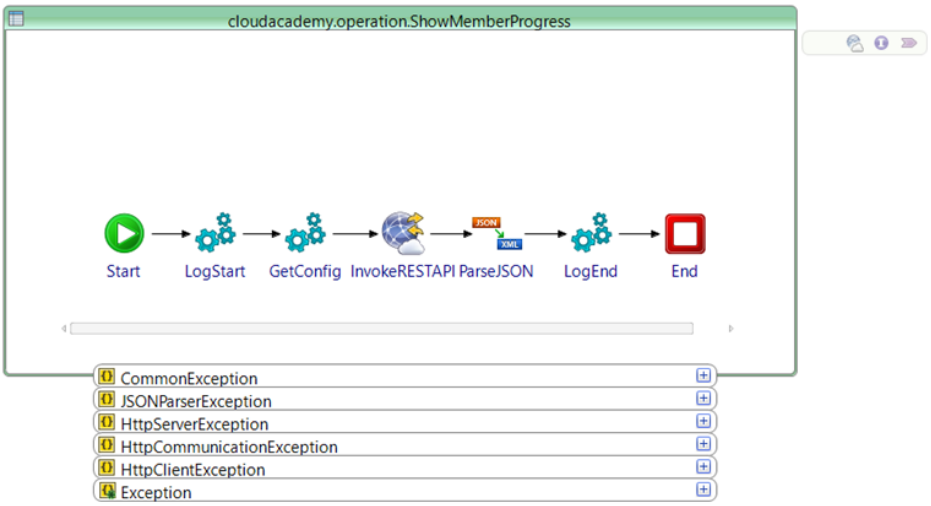


**Figura 5.35:** Architettura processo *ShowMemberProgress*

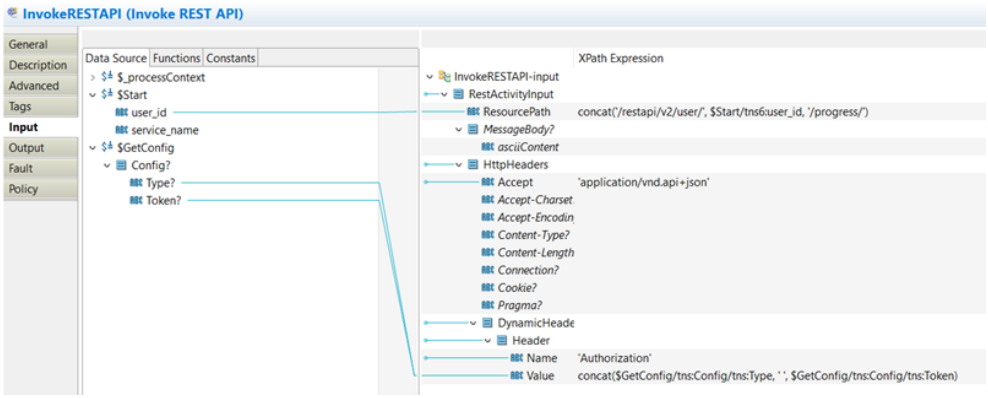


**Figura 5.36:** Mapping request del processo *ShowMemberProgress*

CAPITOLO 6

# Test e validazione del sistema

*In questo capitolo verrà effettuato il test del sistema realizzato e verrà descritto il suo funzionamento con l'obiettivo di dimostrare l'efficienza e la correttezza. Si verificherà, quindi, che quanto è stato realizzato sia conforme con le specifiche definite e che l'applicazione si comporti come previsto.*

## 6.1 Definizione dei casi di test

Nei capitoli precedenti sono state svolte le fasi di analisi dei requisiti, progettazione ed implementazione del sistema finale che rappresenterà la base del progetto per il supporto all'HR Strategy dell'azienda Key Partner. A questo punto si analizzerà la fase di test, fase fondamentale della progettazione di un sistema in ambito IT, poiché precede l'effettivo rilascio ed ha lo scopo di individuare eventuali anomalie ed errori. Se vengono rilevati in questa fase degli errori dovranno essere corretti in modo da rilasciare il prodotto privo di imperfezioni.

Esistono differenti modi per testare un software, i principali sono il testing automatico e quello manuale. Più specificamente:

- *Il test manuale*: prevede un tester, o un team, che controlla manualmente che tutte le funzionalità essenziali del sistema realizzato siano fornite come previsto. In genere, il test del software dovrebbe essere effettuato da persone esperte, terze rispetto al team di sviluppo, in modo da avere una visione imparziale, più vicina all'utente finale che alla macchina e al linguaggio di programmazione.

- *Il test automatico*: prevede che i tester codifichino uno script per testare e convalidare le funzionalità del sistema in modo automatico. Questa tipologia di test prevede uno sforzo maggiore nella fase iniziale, per la produzione dello script; tuttavia, ma in seguito, il test può essere svolto agevolmente risparmiando tempo nell'esecuzione.

Per quanto riguarda il sistema realizzato per il corrente lavoro di tesi sono state testate manualmente tutte le componenti realizzate. È stata utilizzata una request contenente i campi richiesti in input da ogni servizio da richiamare e sono stati eseguiti diversi test per verificare l'effettivo stato di attività dei servizi che vengono richiamati al suo interno, controllando se il sistema risponda o meno correttamente alle fasi di interazione.

## 6.2 Strumenti a supporto del test

Per testare le varie funzionali sviluppate nell'applicazione è stato utilizzato il software Postman ed il Debugger del TIBCO Business Studio. Più specificamente:

- *Postman* è un ambiente di sviluppo API completo. Consente agli sviluppatori di effettuare richieste API, ispezionare i dati di richiesta e risposta ed eseguire il debug degli endpoint API.

  Postman permette di salvare e organizzare richieste API, rendendo facile tenere traccia di diversi endpoint API e passare rapidamente tra loro, oltre a condividerle con altri; ciò lo rende uno strumento molto utile per la collaborazione. Esso supporta vari tipi di richieste, come REST, SOAP, GraphQL, e consente di aggiungere intestazioni personalizzate, autenticazione e parametri alle richieste. Postman gestisce diversi tipi di formati di risposta come JSON, XML, HTML e testo semplice. Esso fornisce inoltre, una funzionalità di test integrato, che consente agli sviluppatori di scrivere test e convalidare la risposta tramite il linguaggio JavaScript. Inoltre, Postman fornisce una funzione chiamata "Server simulato", che consente di simulare le risposte dell'API e testare le richieste senza la necessità di un server reale.

  Per effettuare la chiamata al servizio bisognerà essenzialmente configurare in maniera corretta tutti i parametri necessari. Bisognerà, innanzitutto, settare i valori essenziali di una qualsiasi chiamata a un servizio REST; tali valori sono:

  – la tipologia di chiamata (GET, POST, PUT, DELETE etc.).
  – l'URL da testare.

  Per ogni chiamata sarà possibile impostare la parte relativa all'autorizzazione. Per applicazioni reali, infatti, ad ogni chiamata di un servizio è necessaria un'autorizzazione per poter accedere al servizio stesso. Altra cosa fondamentale quando si espone un servizio REST è la tipologia di dati che viene inviata dall'utente finale e i vari parametri che vengono inviati al server per effettuare la chiamata al servizio; quindi, è importante impostare correttamente la request da inviare.

- *Debugger del TIBCO Business Studio* è uno strumento utilizzato per individuare e risolvere problemi durante lo sviluppo di applicazioni di integrazione con TIBCO BusinessWorks.

  Esso consente agli sviluppatori di interrompere l'esecuzione del codice in un punto specifico e di esaminare lo stato dell'applicazione in quel momento. Ciò consente di identificare rapidamente problemi come errori di logica o di flusso di dati. Il debugger supporta anche la visualizzazione dei dati di debug, la valutazione delle espressioni e la modifica dei valori delle variabili in tempo reale.

  Esso è particolarmente utile durante lo sviluppo di soluzioni di integrazione complesse che coinvolgono molteplici servizi, sistemi e tecnologie. Con questo strumento, gli sviluppatori possono individuare rapidamente e risolvere i problemi prima di distribuire le applicazioni in ambienti di produzione.

  In sintesi, il debugger di TIBCO BusinessWorks è uno strumento essenziale per gli sviluppatori che utilizzano la piattaforma TIBCO BusinessWorks per sviluppare applicazioni di integrazione complesse. Esso consente di identificare rapidamente e di risolvere i problemi durante lo sviluppo, migliorando la qualità delle applicazioni e riducendo i tempi di sviluppo complessivi.

## 6.3 Esito dei test

Per verificare che tutti i requisiti funzionali, precedentemente descritti, siano stati implementati correttamente sono state testate singolarmente le chiamate ai diversi servizi. Di seguito verranno analizzati singolarmente i test effettuati.

- *Esecuzione della chiamata al servizio Add_Utente.*

  Utilizzando l'applicativo Postman è stata inviata una request al servizio esposto dall'applicazione Manager Team per richiedere l'inserimento di un nuovo membro ad un team specifico. Nella Figura 6.1 possiamo vedere una request di esempio che è stata mandata al servizio.

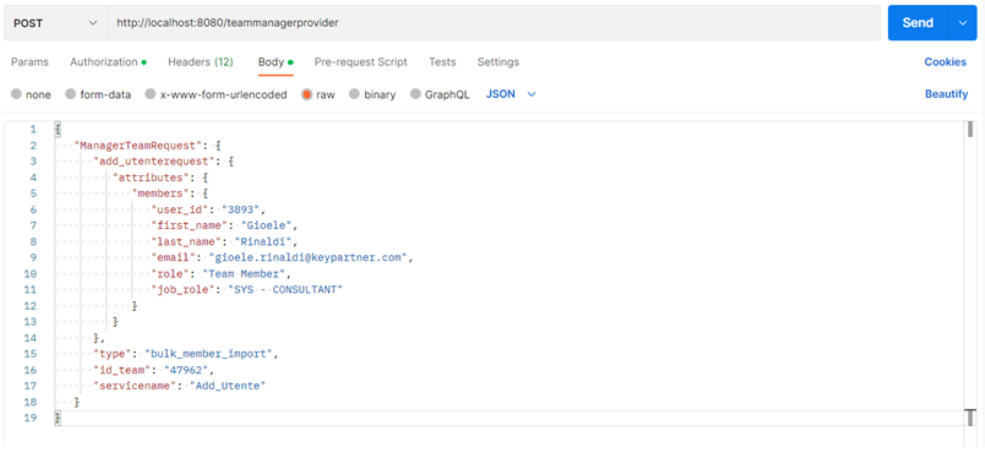


**Figura 6.1:** Request del servizio per l'aggiunta di un membro

Come possiamo vedere, è stata mandata una request al servizio con un'operazione di tipo POST, contenente le informazioni personali del membro da inserire nel team, il tipo di operazione da effettuare sulla piattaforma Cloud Academy, l'id del team sul quale aggiungere il membro e il nome del servizio esposto da Cloud Academy da richiamare. Come response (Figura 6.2) si ottengono le informazioni del membro aggiunto e il risultato dell'operazione; nel caso riportato si ha un'operazione avvenuta con successo e, quindi, il campo *is_success* viene posto a *true*.

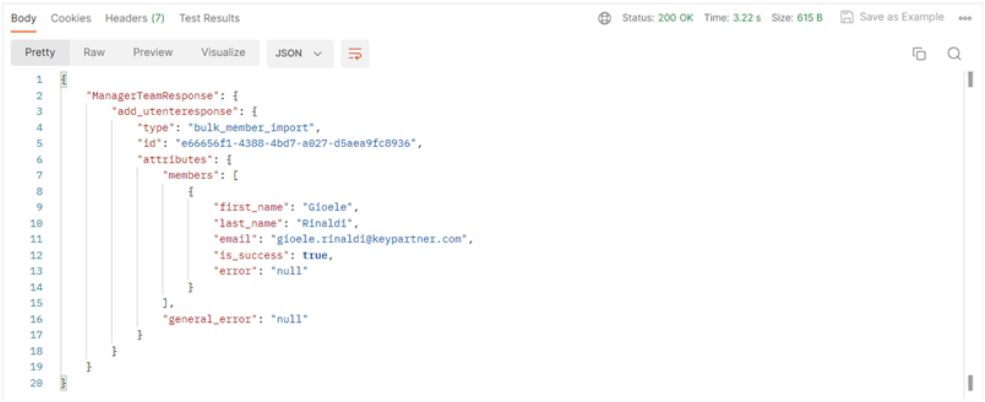


**Figura 6.2:** Response del servizio per l'aggiunta di un membro

Nel caso in cui nella request ci fosse un campo non "valorizzato", la chiamata restituirà un messaggio di errore (Figura 6.3).

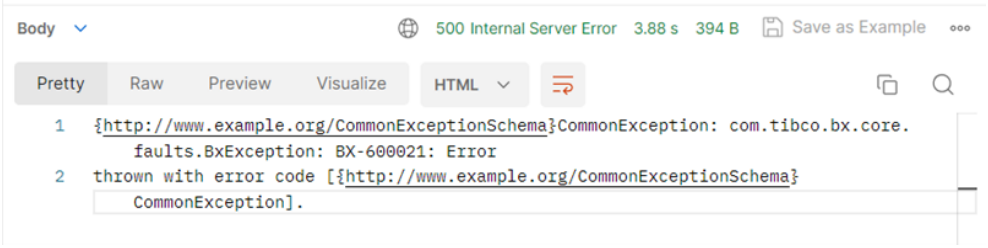


**Figura 6.3:** Response con eccezione su Postman

Si ottiene, invece, un messaggio di errore più esplicativo all'interno dei Job Data contenuti in Business Work; in particolare, il risultato ottenuto è quello mostrato nella Figura 6.4.

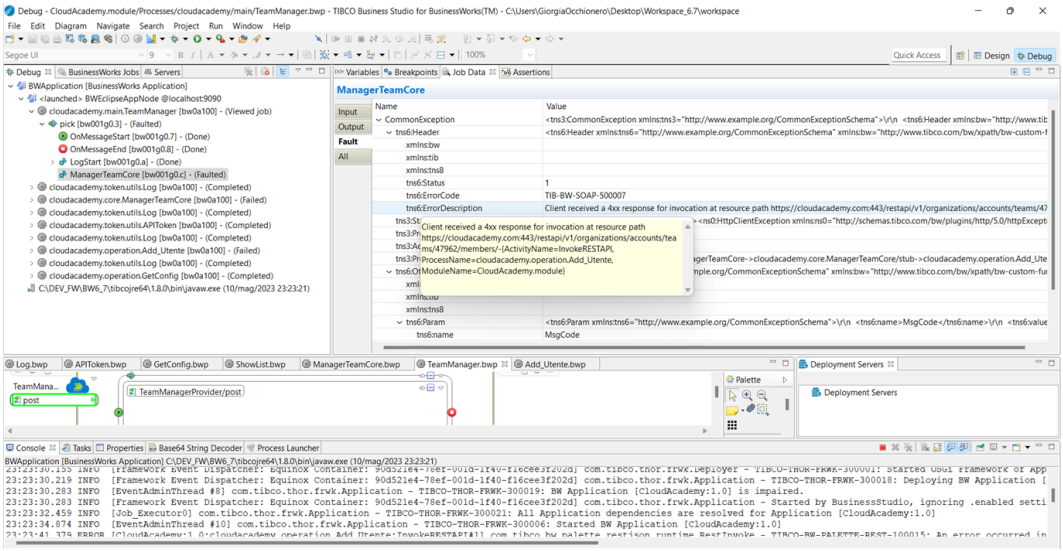


**Figura 6.4:** Messaggio di errore su Business Work

- *Esecuzione della chiamata al servizio Remove_Utente.*

  Per testare il servizio che permette la rimozione di un utente da un team specifico è stata inviata una request al servizio (Figura 6.5), con un'operazione di tipo POST, contenente l'indirizzo email del membro da rimuovere dal team, il tipo di operazione da effettuare sulla piattaforma Cloud Academy, l'id del team dal quale rimuovere il membro e il nome del servizio esposto dalla Cloud Academy da richiamare.

  In response per questo servizio non è stata previsto nessun messaggio; tuttavia è possbile verificare l'avvenuta rimozione dallo status ottenuto dall'operazione, (Figura 6.6).

  In questo caso di test si potrebbe ricevere un messaggio di eccezione nel caso in cui si sia inserito un indirizzo email o un id di un team non presente.

- *Esecuzione della chiamata al servizio ShowList.*

  Per il test del servizio che permette la visualizzazione dei componenti di un team specifico è stata inviata una request al servizio analoga a quella riportata in Figura 6.7.

  Come possiamo vedere, è stata inviata una request al servizio, con un'operazione di tipo POST. (il tipo di operazione da effettuare sulla piattaforma Cloud Academy), Contenente l'id del team del quale si vogliono visualizzare i membri e il nome del servizio

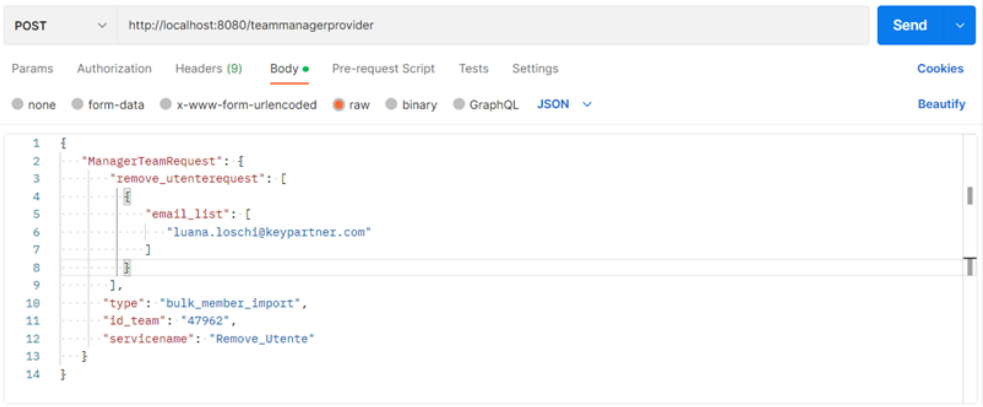


**Figura 6.5:** Request del servizio per la rimozione di un membro

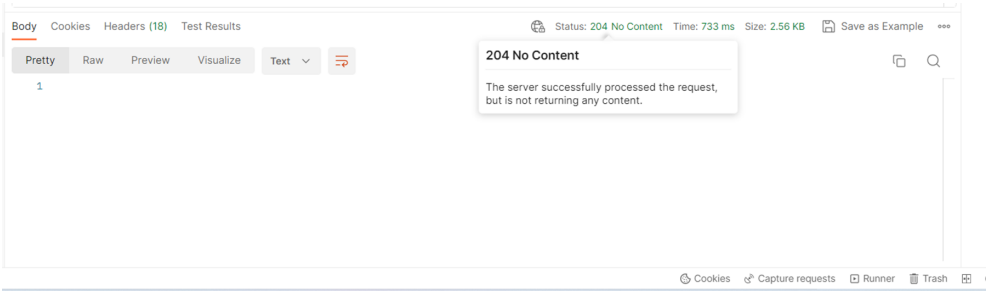


**Figura 6.6:** Response del servizio per la rimozione di un membro

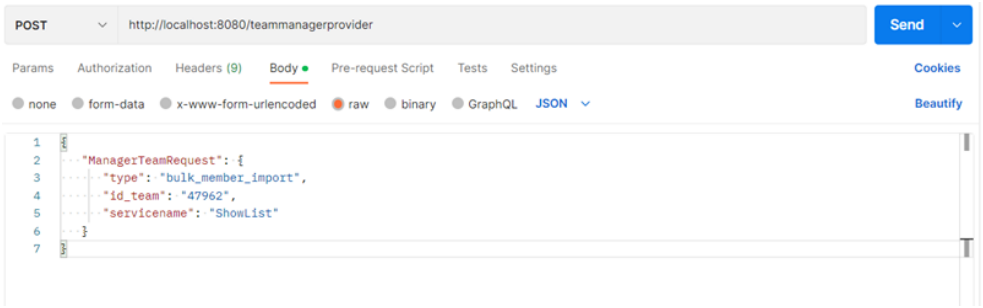


**Figura 6.7:** Request del servizio per la visualizzazione dei membri di un team

esposto dalla Cloud Academy da richiamare. In questo caso, si ottiene come response (Figura 6.8) la lista dei membri appartenenti e le relative informazioni personali.

Nel caso, invece, venga inserito un id di un team non esistente otterremmo una response (Figura 6.9) che segnala l'eccezione ottenuta.

Visualizzando i Job Data (Figura 6.10) relativi all'esecuzione della chiamata in Business Work, grazie alla gestione delle eccezioni implementata nello sviluppo dell'applicazione, possiamo leggere un errore più esplicito e comprendere meglio cosa non ha funzionato correttamente.

- *Esecuzione della chiamata al servizio ShowListMemberProgress*.

  Per il test del servizio ShowListMemberProgress, che permette la visualizzazione del progresso di tutti i membri dell'azienda, è stata inviata una request del tipo riportato in Figura 6.11 al servizio esposto. In questo caso la request non necessita di nessun campo particolare, ma contiene solo il nome del servizio da richiamare. La response ottenuta

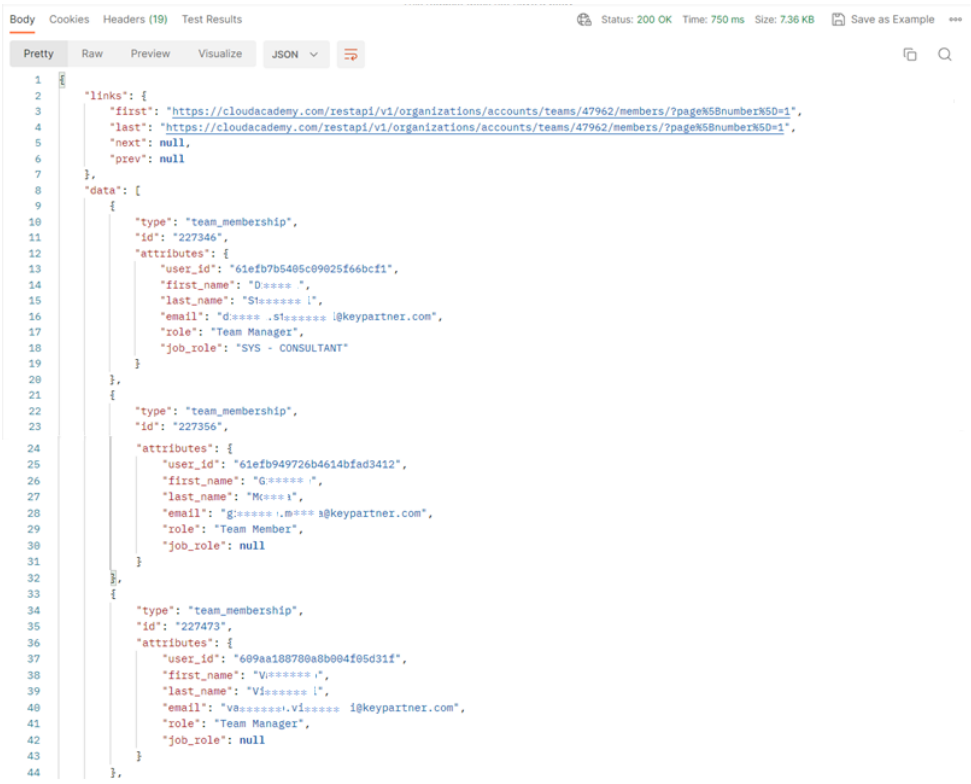


**Figura 6.8:** Response del servizio per la visualizzazione dei membri di un team

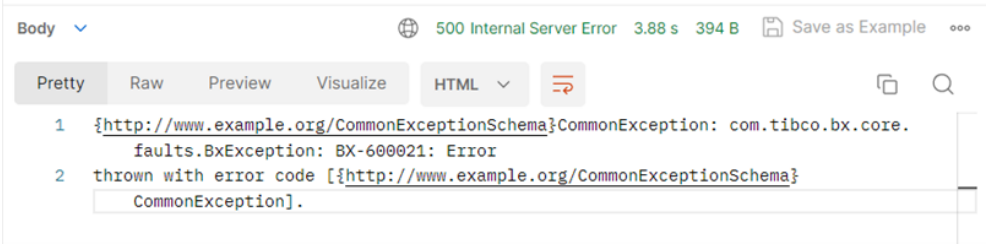


**Figura 6.9:** Response dell'eccezione in Postman

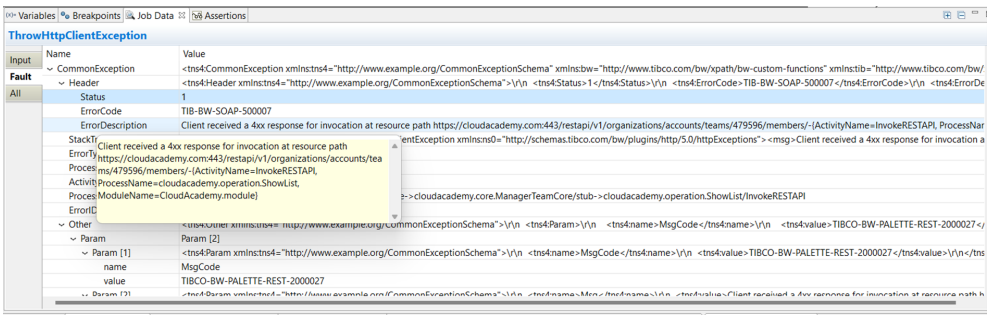


**Figura 6.10:** Job Data relativi all'eccezione in Business Work

dal servizio (Figura 6.12) contiene la lista di tutti i membri dell'azienda; per ognuno vengono riportati, oltre alle informazioni personali, il path per effettuare un'operazione di tipo GET al fine di ottenere il progresso sulla formazione dello specifico membro.

- *Esecuzione della chiamata al servizio ShowMemberProgress*.

  Per il test del servizio *ShowMemberProgress*, che permette la visualizzazione del progres-

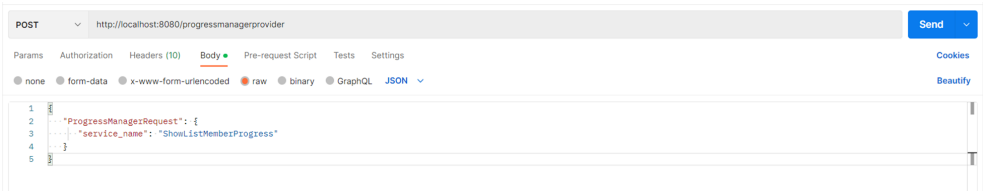


**Figura 6.11:** Request del servizio per la visualizzazione dell'andamento di tutti i membri dell'azienda

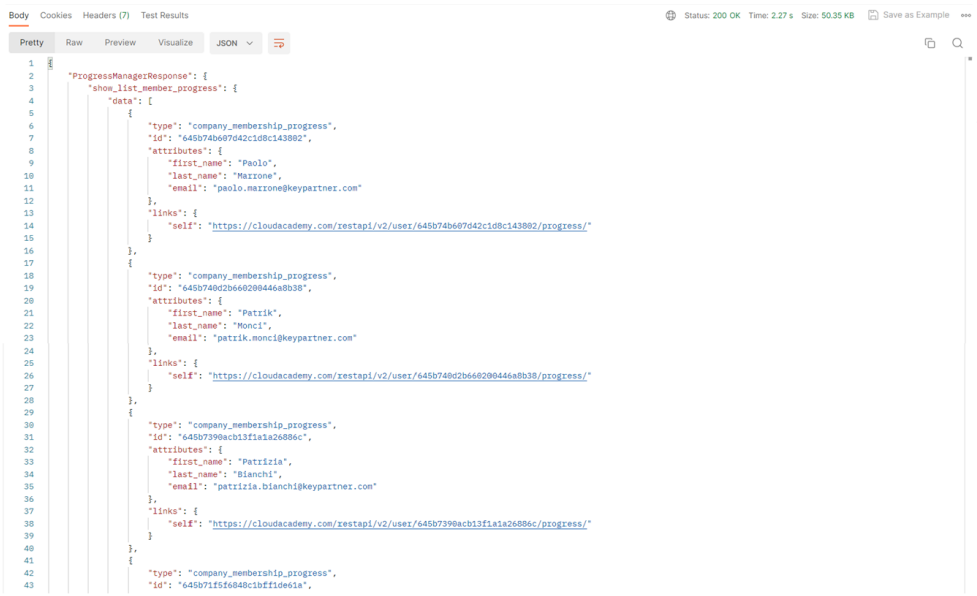


**Figura 6.12:** Response del servizio per la visualizzazione dell'andamento di tutti i membri dell'azienda

so di uno specifico membro dell'azienda, è stata inviata una request del tipo riportato in Figura 6.13 al servizio esposto. Nella request è stato specificato l'id dell'utente del quale si vuole visualizzare il progresso e il nome del servizio da richiamare.

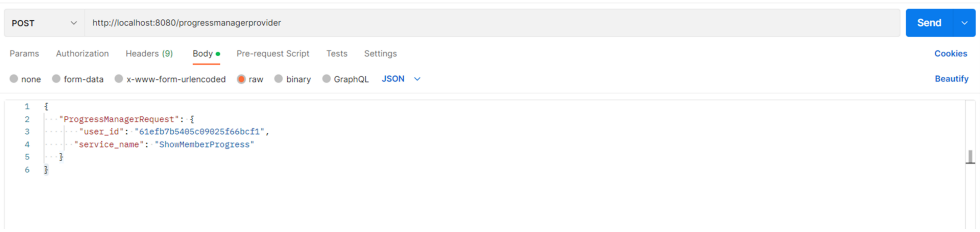


**Figura 6.13:** Request del servizio per la visualizzazione dell'andamento di uno specifico membro dell'azienda

La response ottenuta dal servizio conterrà la lista di tutti i corsi di formazione a cui ha preso parte il membro preso in considerazione. Per ogni corso a cui ha partecipato il membro, sono specificate le informazioni rilevanti, il nome del corso, la data di inizio, la data di fine, la percentuale di svolgimento. Nella Figura 6.14 possiamo vedere una parte del progresso della formazione dell'utente scelto nella request.

Nel caso in cui venga inserito il nome del servizio errato (Figura 6.15) verrà viasualizzato un messaggio di errore visibile nell'applicativo Postman che notifica l'incorrettezza della request (Figura 6.16).

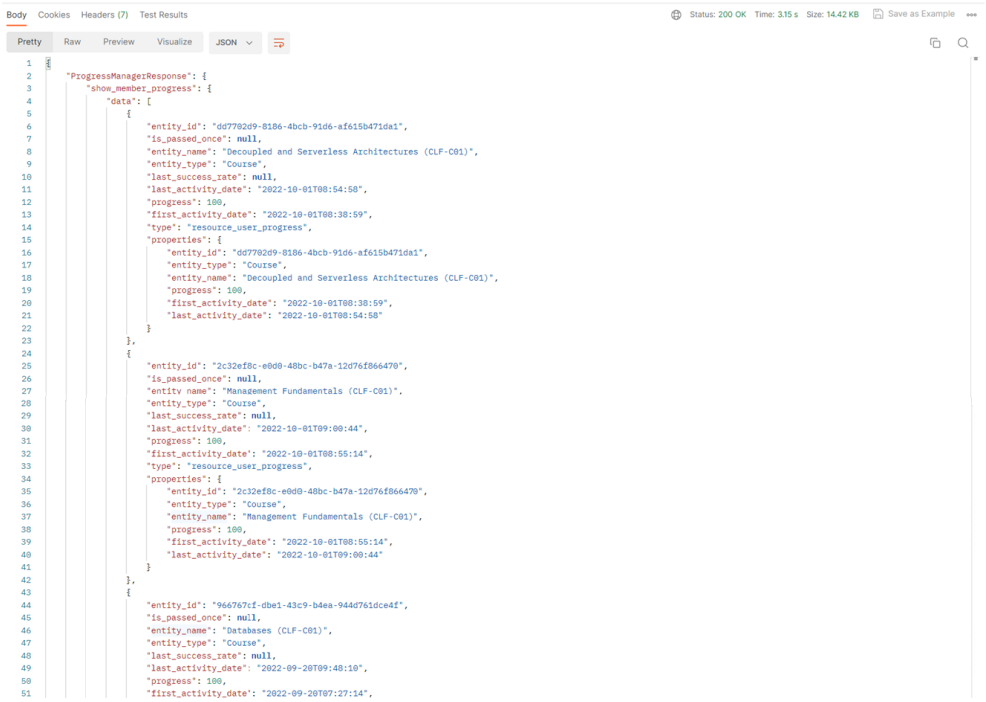


**Figura 6.14:** Response del servizio per la visualizzazione dell'andamento di uno specifico membro dell'azienda

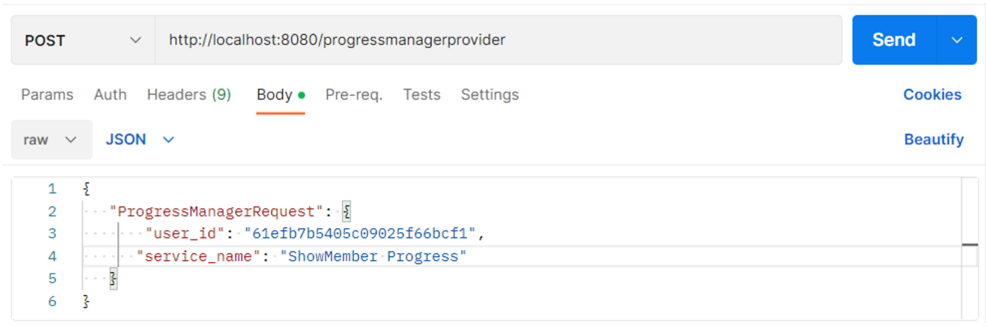


**Figura 6.15:** Request con service_name errato

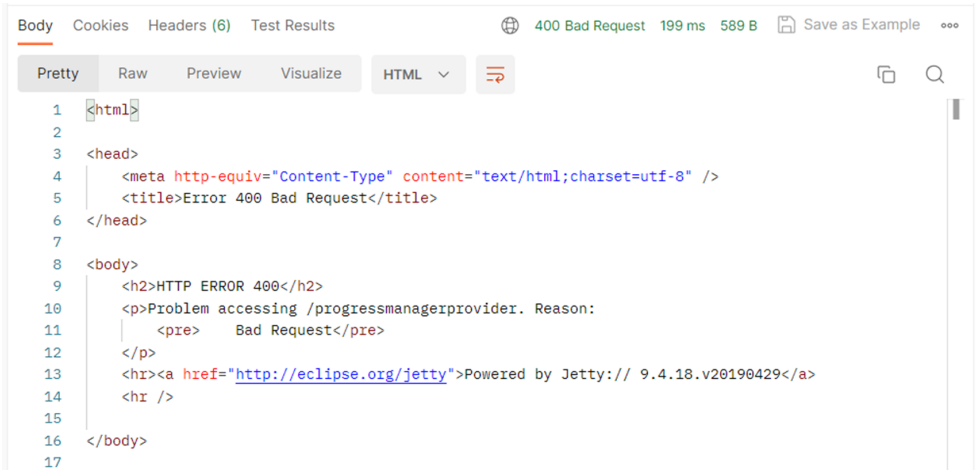


**Figura 6.16:** Response con notifica di errore

CAPITOLO 7

# Discussioni in merito al lavoro svolto

*In questo capitolo verrà analizzato, in maniera critica, il progetto della presente tesi, esponendo i punti di forza e di debolezza del lavoro svolto e soffermandosi su quelle aree che presentano alcune carenze, o che sono state curate di meno, e che, quindi, potrebbero essere migliorate in futuro.*

## 7.1 Punti di forza del sistema realizzato

Al termine di un progetto è opportuno fare un sunto di quanto realizzato con un atteggiamento "critico". In questo modo è possibile capire dove, ed in che modo, è possibile migliorare e perfezionare il lavoro svolto. Riportiamo di seguito quelli che, a parer nostro, rappresentano i punti di forza del progetto:

- L'applicazione realizzata è caratterizzata da un utilizzo facile ed intuitivo e non richiede nessuna specifica competenza per il suo utilizzo. È strutturata in modo che l'utente possa facilmente utilizzarla avendo a disposizione le informazioni necessarie per le attività che voglia svolgere.
- L'applicazione realizzata permette una manutenzione agevole, in quanto è stata sviluppata con una struttura a strati, che permette di disaccoppiare le interfacce e la logica operativa rendendo semplice e rapida l'individuazione di possibili problematiche. A tal proposito è stata prevista una gestione dettagliata dei log, che aiutano a comprendere nello specifico le problematiche di sistema.
- Altro punto di forza, che in qualche modo si collega a quello precedente, riguarda il fatto che l'approccio ad una struttura a strati favorisce anche la modularità e la riusabilità del codice, permettendo un'ottimizzazione e agevolazione delle fasi di sviluppo.
- Il risultato è un sistema pienamente funzionante che integra le API messe a disposizione dalla piattaforma Cloud Academy, rendendo, così, più agile ed intuitivo il lavoro dell'*HR Manager*.
- Il modello di architettura utilizzato per le applicazioni implementate nel presente lavoro di tesi potrebbe essere preso come spunto per l'implementazione di chiamate ad altri servizi.
- Altro punto di forza dell'applicazione realizzata è la versatilità dei pacchetti sviluppati con la suite TIBCO BusinessWorks che possono essere distribuiti in maniera semplice su

diversi target platform. La soluzione offre, perciò, un'estrema versatilità e compatibilità con infrastrutture onPremises, a container e cloud.

## 7.2 Punti di debolezza del sistema realizzato

L'applicazione oggetto della tesi, oltre ai punti di forza citati precedentemente, presenta degli aspetti migliorabili, che, per necessità implementative, di durata del progetto di tesi, non si sono potuti approfondire. Tali aspetti sono i seguenti:

- L'applicazione necessita dell'implementazione di una logica per la gestione della response nel caso di rimozione di un membro da un team. Per il momento non è stato previsto un messaggio di response contenente un body specifico. Come conferma dell'avvenuta rimozione del membro del team, viene interpretato lo status code HTTP restituito dalla chiamata.
- L'applicazione necessita, per la chiamata al servizio di visualizzazione del progresso di un membro specifico, la conoscenza da parte dell'utente dell'id specifico del membro. Questo implica una ricerca preventiva da parte dell'utilizzatore nel caso in cui non fosse a conoscenza dello stesso.
- È necessario utilizzare il protocollo di comunicazione HTTPS (*HyperText Transfer Protocol over Secure Socket Layer*), che implementa la sicurezza attraverso il protocollo di livello di trasporto SSL/TLS. In questo modo si aggiungerebbe un livello di sicurezza ulteriore dettato dall'encrypting dei dati che transitano nel canale applicativo.

# Conclusioni

In questo progetto di tesi sono state trattate le varie fasi relative alla progettazione di un'applicazione che permettesse l'integrazione tra i sistemi Key Partner e API REST messa a disposizione dal portale Cloud Academy, utilizzato dall'azienda Key Partner per la fruizione della formazione al personale.

L'applicazione costituisce un supporto al processo di HR Strategy, agevolando e rendendo più efficiente il lavoro dell'HR Manager. Attraverso il sistema realizzato, l'HR Manager avrà la possibilità di manipolare i team all'interno della piattaforma di Cloud Academy e di analizzare e gestire l'attività di formazione dei diversi dipendenti.

Il lavoro descritto nel presente elaborato non può essere considerato un punto di arrivo per l'azienda Key Partner e per il progetto di integrazione; infatti, il progetto ha tempi che vanno ben oltre la fine di questo elaborato per conseguire l'obiettivo che punta a riprogettare il processo di HR e le componenti tecnologiche ad esso collegato.

Alcuni futuri sviluppi di questo progetto riguarderanno, in primo luogo, delle migliorie sul sistema sviluppato, per poter far fronte ai punti di debolezza descritti nel capitolo precedente.

In un secondo momento verrà effettuata l'integrazione del sistema realizzato all'interno del tool gestionale utilizzato dall'azienda, in questo caso specifico ECOS Agile. Fatto ciò, vi sarà una fase di valutazione e validazione dei risultati ottenuti.

Rimanendo sempre sugli sviluppi futuri e sulle conseguenze di questo progetto, è necessario notare come il modello di architettura utilizzato per le applicazioni implementate nel presente lavoro di tesi. Potrebbe essere preso come spunto per l'implementazione di chiamate ad altri servizi offerti dalla stessa piattaforma Cloud Academy o da altre piattaforme.

Come ulteriore esempio di evoluzione, possiamo mensionare la possibilità di riutilizzare le funzionalità incluse nelle applicazioni oggetto della tesi, inserendole all'interno di processi di elaborazione schedulata (ad esempio a cadenza settimanale) per l'estrazione delle informazioni sui progressi di tutti i membri dei team. Le informazioni estratte potranno essere raccolte e salvate in opportuni database. Questi dati potranno essere analizzati e rappresentati in cruscotti o dashboard di monitoraggio e controllo, a supporto delle mansioni dell'HR Manager.

# Bibliografia


- Key Partner – https://www.keypartner.com/
- Cloud Academy – https://cloudacademy.com/
- Cloud Academy API – https://cloudacademy.com/restapi/
- C. Ghezzi and D. Mandrioli M. Jazayeri. Ingegneria del software Fondamenti e principi. Pearson, 2nd edition, 2004.
- F. A. Cummins. Enterprise Integration: An Architecture for Enterprise Application and System Integration. February 2002.
- RESTful Web Services. – https://www.html.it/guide/restful-web-services-la-guida/
- TIBCO – https://www.tibco.com/
- TIBCO ActiveMatrix BusinessWorks™. Concepts. November 2020. Version 6.7 – https://docs.tibco.com/pub/activematrix_businessworks/6.7.0/doc/pdf/TIB_BW_6.7_concepts.pdf?id=3
- TIBCO ActiveMatrix BusinessWorks™. Getting Started. December 2020. Version 6.7 – https://docs.tibco.com/pub/activematrix_businessworks/6.7.0/doc/pdf/TIB_BW_6.7_getting_started.pdf?id=5
- TIBCO ActiveMatrix BusinessWorks™. Bindings and Palettes Reference. December 2020. Version 6.7 – https://docs.tibco.com/pub/activematrix_businessworks/6.7.0/doc/pdf/TIB_BW_6.7_bindings_palettes_reference.pdf?id=7
- TIBCO ActiveMatrix BusinessWorks™. Administration. December 2020. Version 6.7 – https://docs.tibco.com/pub/activematrix_businessworks/6.7.0/doc/pdf/TIB_BW_6.7_administration.pdf?id=9
- TIBCO ActiveMatrix BusinessWorks™. Application Development. December 2020. Version 6.7 – https://docs.tibco.com/pub/activematrix_businessworks/6.7.0/doc/pdf/TIB_BW_6.7_application_development.pdf?id=10

# Ringraziamenti

Il primo ringraziamento va al mio relatore, il professor Domenico Ursino, che mi ha accompagnato nella stesura della mia tesi seguendomi costantemente con estrema disponibilità.

Ringrazio infinitamente l'azienda Key Partner, che mi ha permesso di intraprendere questo percorso formativo portato a termine con il presente lavoro di tesi.

Ringrazio, altresì, i miei tutor aziendali, Cesare e Fabrizio, senza il cui contributo non sarei riuscita a terminare il mio lavoro di tesi. Li ringrazio per la loro grandissima disponibilità e gentilezza.

Ringrazio infinitamente i miei colleghi, che mi hanno accolta dal primo giorno facendomi sentire a casa.

Ringrazio la mia famiglia, per il costante supporto e la costante vicinanza. Senza di loro non sarei diventata quello che sono e non avrei potuto coronare i miei molteplici sogni.

Ringrazio i compagni di università, perchè tutti hanno contribuito ad aiutarmi a giungere fino qui. Li ringrazio per aver condiviso con me i momenti belli e meno belli di questo percorso.

Infine, voglio ringraziare me stessa per non aver mai perso la voglia e la grinta di arrivare alla fine di questo percorso, nonostante le tante difficoltà.