UNIONE EUROPEA

Ministero dello Sviluppo Economico

Regione Puglia
Dipartimento Sviluppo Economico, Innovazione, Istruzione, Formazione e Lavoro

PUGLIA FESR·FSE 2014/2020
Il futuro alla portata di tutti

BANDO INNOLABS

"Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale"

Y95B457Progetto

serviziLocali S.p.A.

SBSoft

UNIVERSITÀ DEL SALENTO

easyPAL

Deliverable D.C1.2 – Definizione delle interfacce applicative programmabili (API)

Data 28/06/2019

Versione V2.0

NOME DEL DOCUMENTO

Deliverable D.C1.2 – Definizione delle interfacce applicative programmabili (API)

INFORMAZIONI GENERALI

Nome progetto	EasyPAL “Ecosistemi e Servizi Digitali in Cloud per i Cittadini e la PA Locale”
Ambito	BANDO INNOLABS “Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale”
Riservatezza	Riservatezza ai sensi dell'art. 14 dell’atto di costituzione dell’ATS denominata “Easy PAL” registrata a Lecce in data 01/06/2018 al n. 5694/1T da Notaio Pellegrino.

RESPONSABILITA'

Funzione	Nome	Data
Redatto da	Unisalento	28/06/2019
Contributi di	Unisalento, Servizi Locali, SbSoft	
Controllato e approvato da	Servizi Locali	

INDICE

1. Introduzione

Il presente rapporto tecnico ha l'obiettivo di descrivere le nuove interfacce applicative programmabili (API) utente per smartphone e Web-cloud riusando i servizi di base di Ente Digitale, esposti alla piattaforma come API.

Il requisito di rendere la piattaforma multi-canale (multi-tenant) e, di conseguenza, semplificarne l'adozione da parte delle pubbliche amministrazioni locali (PAL), ha rivolto l'attenzione verso l'impiego di un'architettura Cloud e basata su micro-servizi. Questi ultimi sono porzioni autonome di software, che possono interagire tra loro e scambiare dati, nell'ottica di rilasciare all'utente singole funzionalità autoconsistenti.

Si descriverà pertanto il modello e l'architettura di riferimento, focalizzando l'attenzione su Cloud e micro-servizi. Saranno delineate le linee guida teoriche per l'identificazione e documentazione dei servizi. Infine si descriveranno nel dettaglio e secondo le linee guida le nuove interfacce applicative da esporre sopra lo strato di Ente Digitale.

2. Modello e architettura di riferimento

Come da scheda di progetto, il modello concettuale di EasyPAL è un'istanza del "Modello Strategico della PA" su cui sta operando AgID (http://www.agid.gov.it/notizie/2016/07/28/online-linee-guida-il-design-servizi-digitali-pubblica-amministrazione). In accordo con le linee guida enunciate da AgID stessa, ci sono cinque principi fondamentali che devono essere alla base della progettazione e sviluppo dei servizi della PA, ovvero: "partire dal cittadino e dal soddisfacimento delle sue esigenze, agevolando l'effettiva partecipazione civica; garantire un dialogo costante finalizzato al miglioramento della performance, prestare attenzione al design come tratto distintivo per la progettazione, assicurare affidabilità, semplicità e chiarezza, oltre a portare i tratti caratteristici dello stile italiano (progettualità, creatività, estetica) nella pubblica amministrazione". A questi principi si aggiunge l'importanza che riveste l'interazione tra un utente e un'organizzazione, nello specifico tra il cittadino e la Pubblica Amministrazione. Si parla di touchpoint, ovvero di canali che definiscono la relazione tra le due parti consentendo da un lato, la fruizione, dall'altro, la pubblicazione dei servizi della PA. Sempre più frequentemente il canale digitale è cruciale per l'erogazione e acquisizione dei servizi, e da questo è necessario prevedere, progettare e realizzare altri punti di contatto tra PA e cittadino in ottica multi-canale.

In generale, un sistema multi-canale, deve essere in grado di gestire il cambiamento dinamico e continuo del suo ambiente di esecuzione oppure delle caratteristiche dell'utente.

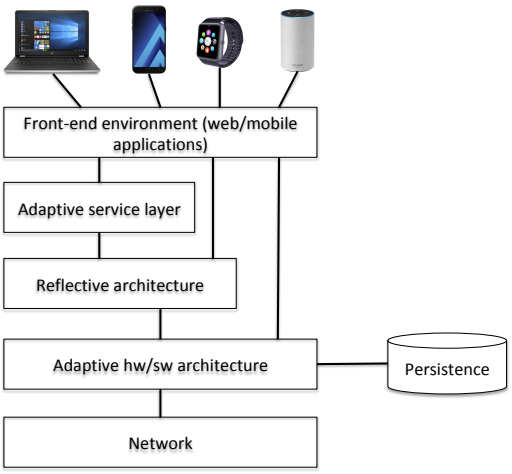


Figura 1: Architettura generale di un sistema multi-canale

Si tratta di un requisito funzionale e non funzionale che si traduce, a livello tecnico, nell'architettura generale riportata in Figura 1, in cui i componenti possono essere orchestrati per fornire funzioni di adattamento e multi-canalità a più livelli

- I componenti di front-end, hanno la responsabilità di adattare dinamicamente l'interfaccia utente delle applicazioni Web (fisse e mobili);
- I componenti di back-end, si occupano di fornire i servizi Web in modo adattativo al front-end, in base al contesto d'uso e al canale di comunicazione;
- Un'architettura riflessiva, consente sia al front-end sia al back-end di avere il controllo dinamico dell'ambiente hardware di esecuzione e delle relative "capability", per garantire la qualità del servizio;
- Componenti infrastrutturali, gestiscono l'adattamento dinamico del sistema al canale di rete.

Entrando in maggiore dettaglio, i componenti di front-end possono gestire due aspetti differenti di adattamento dinamico a supporto della multi-canalità: (1) adattamento alle caratteristiche fisiche (ad esempio, di banda) della rete di trasmissione disponibile per i dispositivi mobili, oppure ad architetture specifiche che possono essere adottate per minimizzare il consumo di potenza; (2) strumenti e metodi che possono assistere l'ingegnere durante la progettazione di applicazioni Web adattative (ad esempio, mash-up, profiling, content adaptation/summarization, location-awareness, off-line management, ecc.).

I componenti di back-end gestiscono l'adattamento multi-canale, mettendo a disposizione dell'intero sistema un meccanismo di "provisioning" dinamico dei servizi: differenti servizi descritti all'interno di un "service repository" possono essere selezionati in risposta alle richieste, in base alle caratteristiche del contesto di invocazione. Inoltre, servizi a più alto livello possono essere generati per orchestrazione dinamica di servizi elementari.

2.1 Modello di provisioning

Per ogni servizio (atomico o composto) che offre il sistema, deve essere predisposta la specifica funzionale del servizio dal punto di vista del provider (o dei provider, se i servizi sono in più esemplari alternativi), oltre che la specifica di invocazione del servizio medesimo dal punto di vista di chi lo consuma. In Figura 2 è riportato un modello di provisioning di servizi molto flessibile, riscontrato in fase di analisi dello stato dell'arte in [Baresi 2003].

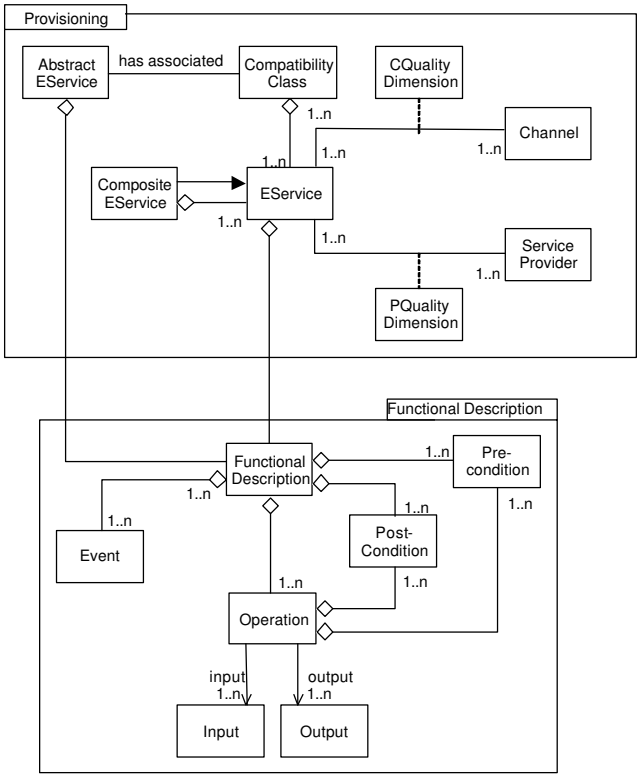


Figura 2: Modello di provisionig dei servizi [Baresi 2003]

Il modello si ripartisce in attributi generali di provisioning del servizio (tra questi, la Compatibility Class per la scelta dinamica di esemplari tra più alternative, Quality of Service sia dal punto di vista del canale di comunicazione sia del provider) e descrizione funzionale (la semantica del servizio) specificata con la tecnica dei contratti (pre-condizioni, post-condizioni, eventi). La Functional Description del servizio ne

descrive gli aspetti operativi. E' un insieme di Operation aventi un nome e una descrizione testuale di quello che fa l'operazione stessa.

Analogamente, in Figura 3 è rappresentato un modello di invocazione dei servizi [Baresi 2003] altrettanto flessibile.

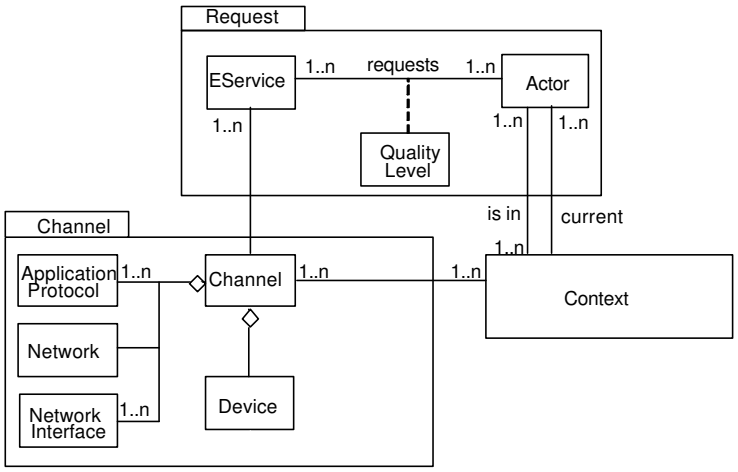


Figura 3: Modello di invocazione dei servizi [Baresi 2003]

L'invocazione del servizio avviene specificando i parametri di qualità desiderati. Il servizio è selezionato tenendo conto del contesto di utilizzo corrente, delle caratteristiche del canale di comunicazione e del dispositivo su cui i servizi sono fruiti.

2.2 Il modello del contesto

Più volte nel documento è stato menzionato il contesto di utilizzo, al fine di caratterizzarlo meglio, si riporta di seguito (Figura 4) una rappresentazione degli attributi che lo compongono (come in [Pernici 2006]).

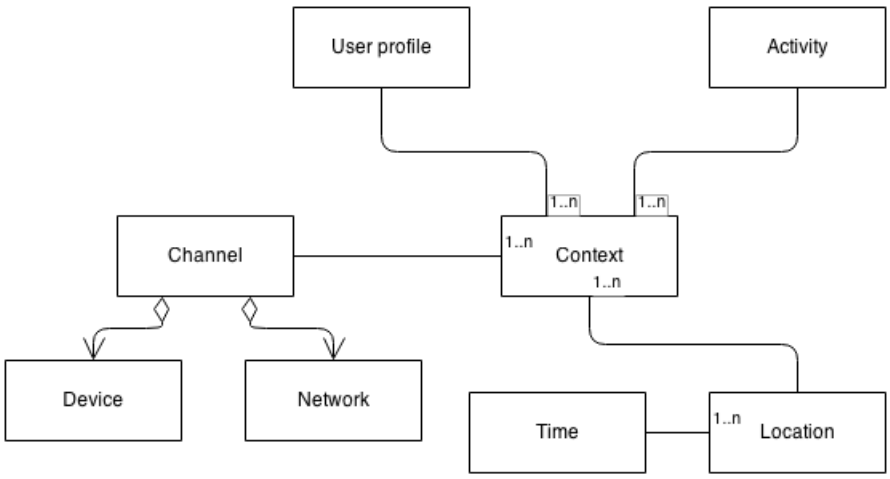


Figura 4: Attributi del contesto di utilizzo

Non esistendo una definizione universalmente adottata di contesto, si considerano appartenenti ad esso tutti quegli attributi che influenzano i meccanismi di accesso ai servizi, il profilo e le preferenze dell'utente. Ad esempio, gli attributi che caratterizzano il dispositivo di accesso, la QoS della rete di accesso, il profilo

dell'utente, il luogo fisico (location) e il momento (time) di accesso. Più precisamente, il contesto può essere descritto come l'aggregazione di 4 gruppi principali di proprietà (Figura 4):

- *Channel (Canale)*, che corrisponde al canale di interazione tra l'utente e il sistema, inteso come il dispositivo che l'utente utilizza e la connessione di rete che questo instaura con l'applicazione;
- *Location and Time*, che identifica la posizione spazio/temporale dell'utente durante l'interazione con il sistema;
- *User Profile*, che a suo volta aggrega un insieme di attributi che caratterizzano secondo varie dimensioni l'utente che interagisce con il sistema;
- *Activity*, che raggruppano un insieme di attività che l'utente sta effettuando durante l'interazione con il sistema.

Channel (Canale). Comprende gli attributi che caratterizzano il concetto di "canale di distribuzione delle informazioni" da parte dei servizi. Tale caratterizzazione può essere data a due livelli: (i) a livello concettuale (canale concettuale di distribuzione), la descrizione degli elementi che intervengono è molto aggregata, ad esempio un canale concettuale è il Web (quindi terminale fisso su protocollo http) oppure il Mobile (quindi dispositivo mobile); (ii) a livello logico (canale logico di distribuzione) è necessario entrare in un maggiore dettaglio delle caratteristiche del dispositivo fisico che l'utente utilizza (anche più di uno), dell'interfaccia di rete che il dispositivo impiega per la connessione all'infrastruttura, della rete impiegata per il trasferimento dei dati e dei protocolli applicativi utilizzati dai servizi. Descrivere a livello logico in modo dettagliato gli attributi che specificano le caratteristiche del dispositivo utente, della rete e dei protocolli applicativi, consente di introdurre politiche di selezione e orchestrazione dinamica di servizi nel caso il catalogo ne fornisca di versioni e istanze differenti. La Figura 5 illustriamo un possibile modello logico di canale.

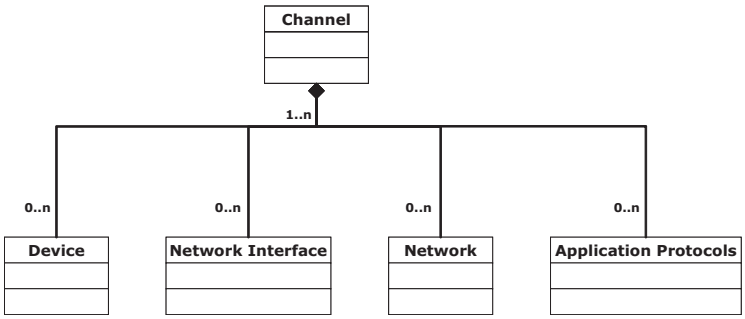


Figura 5: Modello logico di canale

Location and Time. Mentre la localizzazione temporale è di per sé atomica, la localizzazione spaziale può essere scomposta in localizzazione geografica (eventualmente distinguendo anche tra geografia politica e geografia fisica) e localizzazione architettonica (posizionamento all'interno di edifici). La localizzazione negli edifici (e comunque nei luoghi chiusi) è particolarmente rilevante per applicazioni di domotica, di assistenza domiciliare e, più in generale, per servizi di "ambient assisted living".

User profile. E' possibile distinguere tre sottoinsiemi di attributi: (i) *user information*, impiegato per descrivere tutto ciò che il sistema conosce dell'utente (informazioni personali, expertise, ruolo, ecc.); (ii) *user preferences*, utilizzate per descrivere preferenze d'uso del sistema o di sui componenti anche osservate o specificate in sessioni d'interazione passate tra utente e servizio (anche caratterizzate in modo quantitativo oppure qualitativo); (iii) *user abilities*, che descrivono – ad esempio impiegando lo standard ICF (International Classification of Functioning, Disability and Health) – le abilità dell'utente nell'utilizzo di

sistemi ICT. In Figura 6 si illustra un possibile modello logico di profilo dell'utente che tiene in conto di quanto descritto.

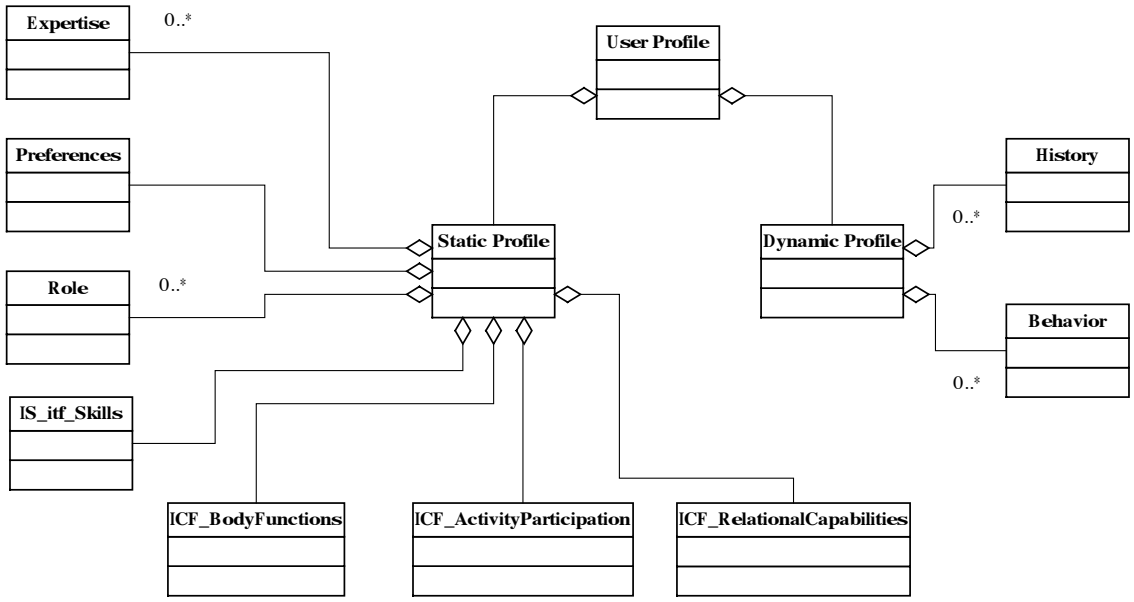


Figura 6: Modello di profilo utente

2.3 Macro-architettura

In questa sezione verrà ripresa la macroarchitettura di EasyPAL specificando per ciascun sottocomponente, le funzionalità erogate, le principali interfacce, sia intercomponente che verso l'esterno definendo le corrispondenti mutue relazioni.

2.3.1 Diagramma delle componenti logiche

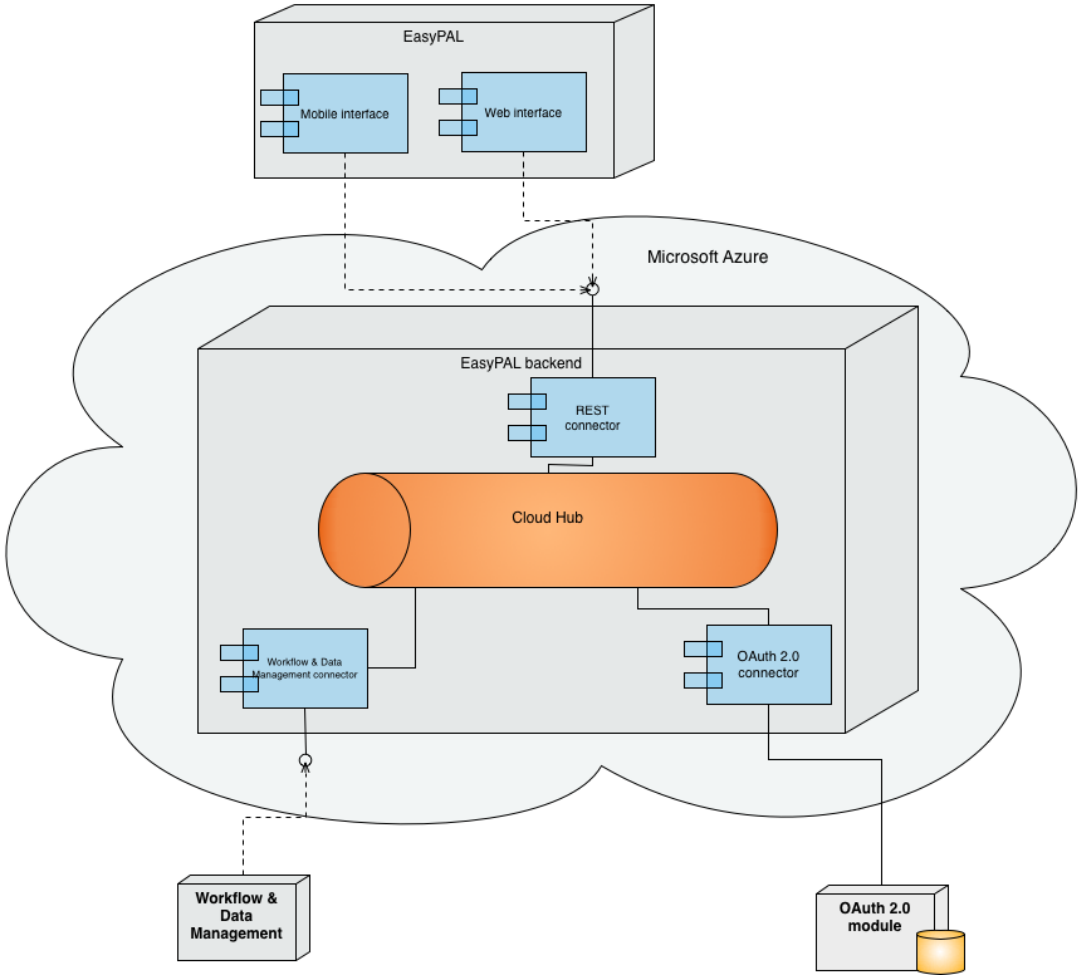


Figura 7: Diagramma delle componenti logiche

L'architettura del componente (Figura 7) si basa sulla predisposizione di una soluzione multi-tenant di Cloud Hub. Tale architettura dovrà garantire la definizione e la gestione del delivery di un insieme di servizi provenienti da diverse componenti orchestrate in maniera opportuna.

A tale scopo si dovranno definire delle sotto-componenti, che avranno il ruolo di connettori adattativi, fondamentali per le integrazioni automatiche o pseudo automatiche tra la piattaforma e i vari sistemi informatici "terze parti" delle imprese (partner) aderenti all'HUB. Tali connettori dovranno svolgere le funzioni di traduttori e trasportatori delle informazioni.

Nel caso in cui un generico partner volesse personalizzare i servizi offerti, la piattaforma EasyPAL dovrà permettere l'accesso alle specifiche funzionalità tramite un accesso OAuth2.0. Al momento non si prevede che la piattaforma abbia specifiche interfacce utente Web per la fruizione di funzionalità dirette da e verso gli utenti finali. Tutti i servizi che si vogliono sottoporre ad autenticazione e autorizzazione mediante questo canale dovranno essere configurati a monte per dialogare con lo standard OAuth.

La componente Cloud dovrà avere un ruolo primario all'interno della infrastruttura operativa della piattaforma EasyPAL in quanto dovrà asservire a tutte le comunicazioni che si dovranno attuare ed inter scambiare all'interno della piattaforma EasyPAL da parte dei diversi componenti previsti dal progetto. A tale scopo dovranno essere implementati specifici connettori per la interoperabilità con tali componenti. Uno specifico, ad esempio, rappresenta quello relativo al sistema di Workflow & Data management, oppure

quello della definizione ed esecuzione dei processi di Business (POD) ecc. Si precisa comunque, affinché non ci sia una proliferazione di connettori con tecnologie variegate e differenti, e laddove fosse possibile, che le componenti di piattaforma si integreranno attraverso uso di tecnologie Web Service basate sui principi REST che rappresentano una alternativa ai Web Service di tipo SOAP e WSDL.

L'integrazione con i sistemi informatici di "terze parti", mediante connettori, deve essere, nei casi ad esempio di partner di una certa rilevanza (ad esempio grandi operatori della PA), definita in maniera adattativa e non invasiva. Questo significa che i connettori dovranno essere implementati in modo da adattarsi ai protocolli di interoperabilità informatici tipici del sistema informatico di riferimento del partner.

Non si esclude che per i restanti casi, laddove una analisi del sistema informativo del partner lo consenta e in particolari condizioni favorevoli, sviluppare dei connettori ad hoc specifici da essere integrati all'interno del sistema informatico del partner.

2.4 Modulo Cloud Hub

Esistono diverse implementazioni di Cloud Hub sia open source che commerciali, con diversi gradi di integrazione in piattaforme e standard preesistenti. Per la realizzazione del componente Cloud Hub del modulo EasyPAL si è preferito realizzare una soluzione custom da zero.

Lo sviluppo del Cloud Hub è stato dettato dalla necessità di eliminare tutte le connessioni punto-punto esistenti tra sistemi, servizi, API e dispositivi, e gestire il routing dei messaggi, la sicurezza, l'affidabilità, e scalabilità tra i nodi. Quindi, fondamentalmente, un Cloud Hub consente di passare da uno scenario del tipo (Figura 8):

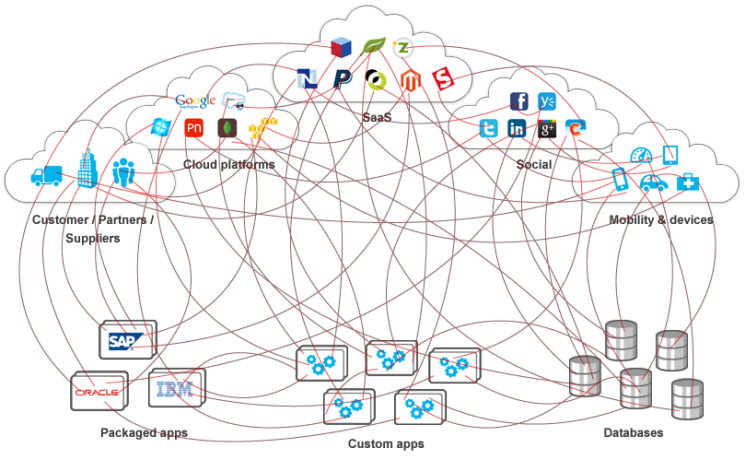


Figura 8: Scenario senza Cloud Hub: "spaghetti-like" point-to-point integration

al seguente (Figura 9):

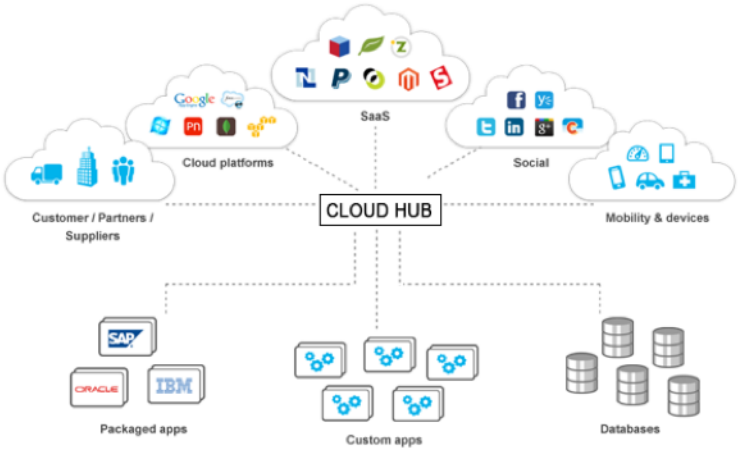


Figura 9: Scenario con Cloud Hub

Con l'utilizzo di un Cloud Hub è possibile:

- integrare applicazioni o sistemi on premise o in cloud;
- usare connettori in uscita per integrare applicazioni come servizi (Software as a Service (SaaS));
- realizzare e rendere disponibili API;
- creare servizi Web che gestiscono le chiamate ad altri servizi;
- creare interfacce per l'utilizzo di applicazioni su dispositivi mobili;
- integrare la B2B con soluzioni che siano sicure, efficienti e veloci da realizzare e rilasciare;
- spostare attività su cloud;
- collegare attività di B2B e-commerce.

2.4.1 Service Oriented Architecture (SOA)

Il Cloud Hub è basato sul concetto di Service Oriented Architecture (SOA). L'approccio SOA allo sviluppo permette alle aziende IT di comporre applicazioni costruendo interfacce di servizi attorno ai componenti applicativi ed esponendo tali interfacce ai fruitori dei servizi. Per servizi si intende un insieme discreto di funzionalità che sono completamente separate tra loro, ma che possono lavorare insieme su un insieme canonico di oggetti. Per esempio, dovendo processare le fatture dei clienti, si potrebbe avere un servizio che inserisca nella fattura i dati relativi al cliente presenti su un database, un altro servizio che verifichi la presenza in magazzino delle merci richieste consultando il database di inventario e un terzo servizio che soddisfi l'ordine.

Dal momento che ogni servizio è indipendente dagli altri, essi possono essere utilizzati come building block per vari processi, evitando di doverli ricreare per ogni nuovo processo o tipo di messaggio. Ad esempio, il servizio che unisce i dati del cliente dal database alla fattura citato prima potrebbe essere utilizzato per inserire tali dati in qualsiasi altro documento. La logica che determina come i dati sono inseriti nella fattura è disaccoppiata dalla logica che cerca e ritorna i dati necessari. Questo approccio modulare permette la creazione di funzionalità riusabili, snellendo lo sviluppo.

Tale disaccoppiamento è l'elemento chiave della service mediation, un pattern per promuovere il basso accoppiamento. Nell'ambito SOA, la mediation descrive anche le facoltà del Cloud Hub di effettuare conversioni tra protocolli di trasporto differenti (anche tra protocolli sincroni e asincroni), cambiando la rappresentazione dei messaggi trasformando i dati, e imponendo la conformità alle politiche attraverso auditing, logging, monitoraggio di sicurezza etc.

L'operazione di comporre i building block in un processo logico è detta orchestrazione. L'orchestrazione dei servizi usando un Cloud Hub permette l'automatizzazione dei processi di back end più comuni a livello applicativo. Questi processi possono essere schedulati o essere esposti come servizi che vengono scatenati da eventi esterni. L'orchestrazione dei servizi differisce dall'orchestrazione dei processi di business, dal momento che quest'ultima può prevedere processi di business stateful più lunghi, interazioni e approvazioni umane complesse o l'esecuzione in parallelo di questo tipo di processi a livello di business process.

Usando l'approccio SOA, le aziende possono realizzare risparmi notevoli sui costi di sviluppo e possono adattarsi rapidamente alle condizioni di business riusando e/o riconfigurando i servizi esistenti per lo sviluppo di nuove applicazioni. Tale metodologia permette inoltre una migliore integrazione delle risorse IT dell'impresa, comprese applicazioni esistenti e sistemi legacy.

2.4.2 Architettura logica Cloud Hub del modulo EasyPAL

Il Cloud Hub oggetto della progettazione (Figura 10), espone una serie di endpoint inbound che servono per la ricezione dei messaggi sui vari canali di trasporto e una serie di endpoint outbound per l'invio dei messaggi al destinatario. Inoltre sono presenti degli endpoint specifici:

- per l'eventuale archiviazione sostitutiva (database o File System);
- per il collegamento con il portale (eventualmente possono essercene più di uno a seconda di come viene strutturata l'interazione tra Il Cloud Hub e il portale);

- per il collegamento all'eventuale modulo di apposizione della firma digitale/marca temporale.

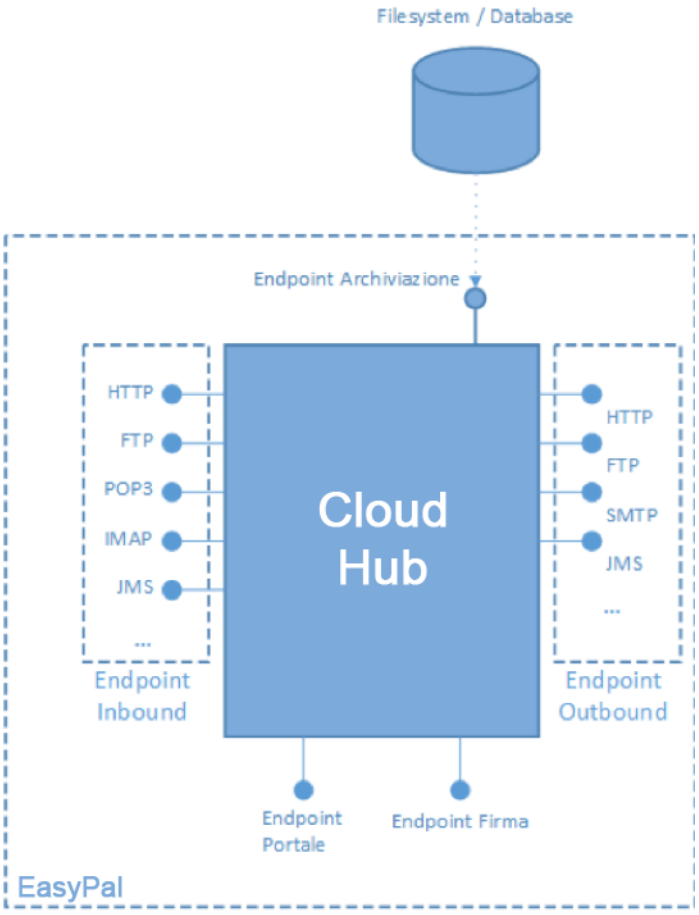


Figura 10: Endpoint esposti dal Cloud Hub

Entrando maggiormente in dettaglio, la figura successiva riporta l'elenco di servizi che saranno mascherati dal Cloud Hub.

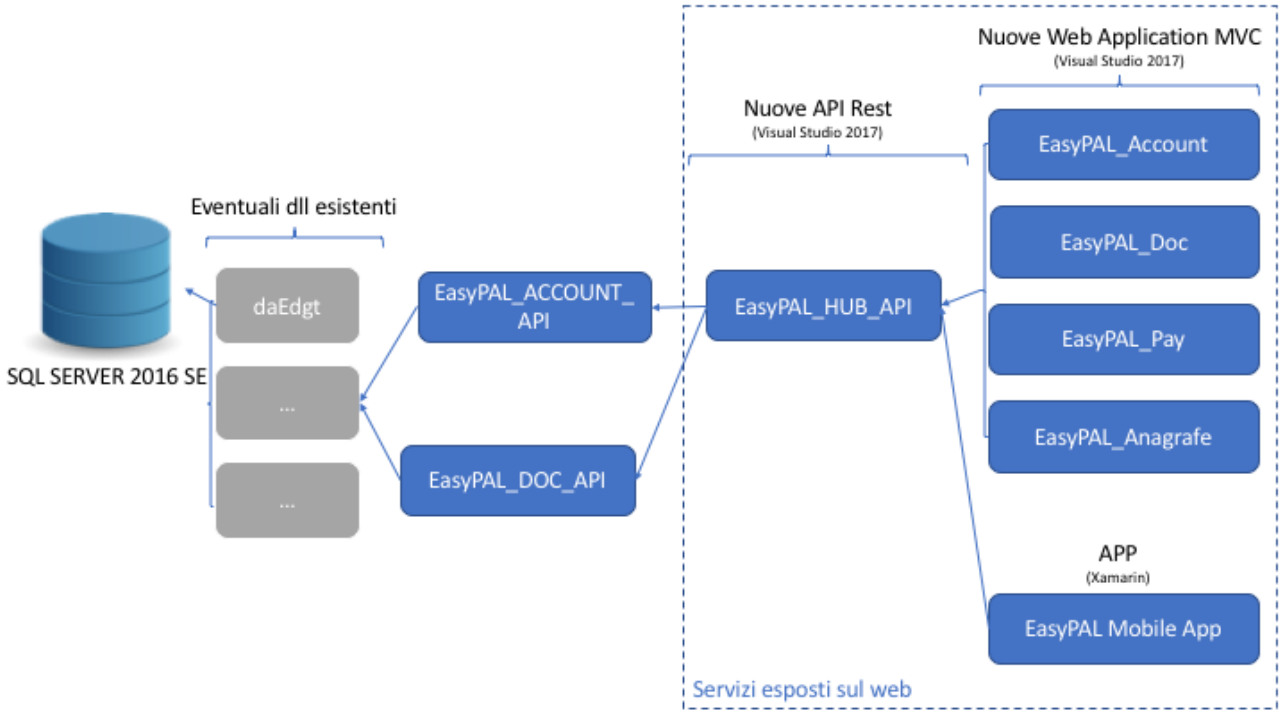


Figura 11: Macroarchitettura fisica

Nel diagramma, il flusso dell'informazione è da intendersi orientato da destra verso sinistra. Le nuove Web Application create nel progetto chiamano tutte il medesimo endpoint REST, rappresentato appunto dal Cloud Hub, pubblicato in una DMZ. Il Cloud Hub conosce l'API specifica da contattare per soddisfare la specifica richiesta in arrivo, la quale a sua volta può utilizzare librerie a basso livello (dll) per accedere a specifiche risorse, fra cui ad esempio una base di dati.

2.4.3 Il protocollo OAuth

OAuth è un protocollo aperto, sviluppato da Blaine Cook e Chris Messina a partire da novembre 2006 e definito all'interno della *RFC 5849* (*Request For Comments*) pubblicata nell'aprile 2010. Tale protocollo permette l'autorizzazione di *API* (*Application Programming Interface*) di sicurezza con un metodo standard e semplice sia per applicazioni mobile, sia per applicazioni stand-alone che per le applicazioni web.

OAuth sta emergendo come standard web per l'autorizzazione di accesso ad app e dati; è progettato in modo che gli utenti, relativamente alle risorse (e-mail, foto profilo, numero di cellulare) presenti su un sito come ad esempio Facebook - detto *Service Provider* - di cui sono titolari, possano concedere accesso limitato a un client di terze parti - chiamato *Consumer* - quale un app o un sito web.

Il mondo dei social è il principale sostenitore di tale protocollo e questi ultimi devono gran parte del loro successo al fatto di non essere dei siti web autonomi, ma piattaforme che promuovono l'integrazione con altre app.

Per l'utente finale si tratta di un processo semplice e intuitivo e questo rappresenta gran parte del fascino di OAuth. Questa semplicità nasconde però un'interazione in background molto più complessa.

In questo scenario il server di autorizzazione è nettamente separato dal server delle risorse, che sarà accessibile dall'utente solo dopo la fase di autenticazione.

Il *client* richiede l'accesso in nome dell'utente che lo autorizza al *Authorization Server* (AS) tramite l'utilizzo di apposite API (Application *Programming* Interface).

Per utilizzare questi metodi è necessario che gli utenti si autentichino al sito in questione con le loro credenziali, per poi autorizzare l'accesso all'applicazione. Quindi, l'AS restituirà al *client* un *token* che verrà utilizzato successivamente per accedere al *Resource Server* (RS), cosi come mostrato in Figura 12.

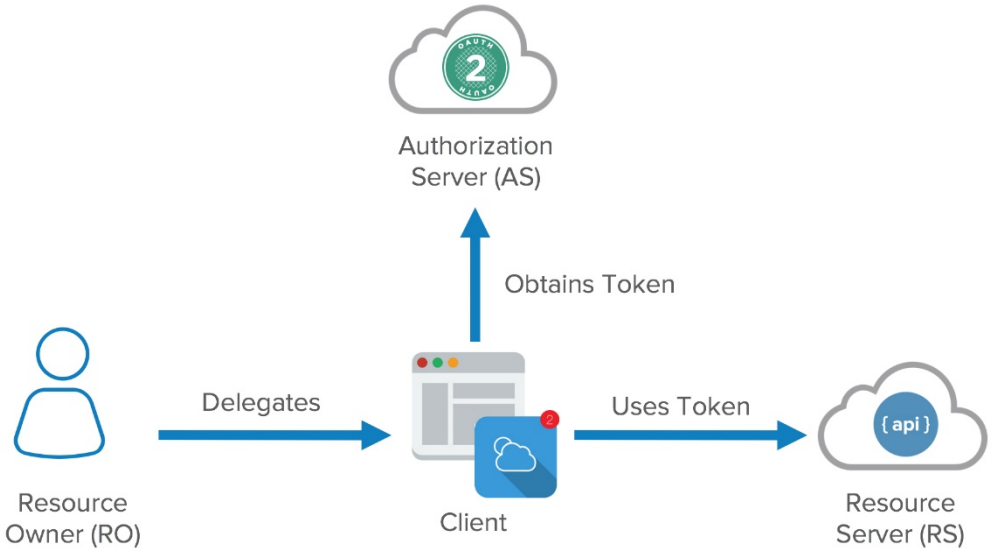


Figura 12: Protocollo OAuth

Come *Authorization Server* (AS) verrà utilizzato l'end point SPID di accesso ai servizi del Comune di **Altamura** (per le utenze di tipo **Cittadino**). Il token verrà restituito mediante un meccanismo custom. Per le utenze di tipo operatore sarà invece implementato un accesso classico mediante username/password.

Esistono molti tipi di token; questi hanno una data di scadenza e possono essere aggiornati. Di solito, una volta effettuato l'accesso correttamente, l'AS restituisce ulteriori informazioni come la durata del token e un token secondario valido da utilizzare dopo la scadenza del primo. Infatti, all'interno dell'ecosistema di applicazioni, una volta che il cliente è già loggato non dovrà, nei successivi accessi, reinserire le proprie credenziali per accedere alla propria area riservata piuttosto che al catalogo se il token d'accesso è ancora valido.

Tale protocollo si integra completamente con i sistemi di controllo aziendale, consentendo ai fornitori di risorse di sapere chi accede a cosa e quando, e quindi di studiare il comportamento degli utenti per sviluppare un CRM sempre più solido, e aumentare di conseguenza la percentuale di fidelizzazione.

3. Architettura Cloud a micro-servizi

Il concetto di micro-servizio è stato introdotto quasi contemporaneamente da Lewis in [Lewis 2012] e George in [George 2012]. Ha avuto rapidamente successo, ma anche varie evoluzioni [Lewis 2014, Hassan 2016, Balalaie 2016, Garriga 2017] fino ad arrivare a quello che oggi la comunità degli ingegneri del software condivide [Jamshidi 2018].

Non esiste, tuttavia, una definizione univoca di micro-servizio. Il concetto ruota attorno a uno stile architetturale ideato per lo sviluppo di applicazioni come insieme di servizi di "piccole dimensioni", ognuno dei quali esegue un processo dedicato e comunica con gli altri livelli dell'architettura (in primis, il livello di presentazione dei dati) ricorrendo a meccanismi e protocolli agili, molto spesso con API RESTful [Lewis 2014]. Un micro-servizio è, quindi, un'unità di business logic, autonoma, indipendente, isolata, che rappresenta un valore riconoscibile per l'utente (consente all'utente di svolgere un'attività identificata nei requisiti dell'applicazione), ma anche per gli atri stakeholder, poiché consente di raggiungere un valore di business per cui l'applicazione è stata ideata. Un micro-servizio non coincide con una singola operazione, ma con un gruppo coeso di operazioni. Ogni micro-servizio deve avere la sua porzione di dati dedicata, sulla quale può agire indipendentemente da altri micro-servizi. Ogni micro-servizio può essere orchestrato con altri micro-servizi per eseguire operazioni di più alto livello, processi e transazioni.

3.1 Identificazione dei micro-servizi dall'analisi dei processi

Obiettivo di queste linee guida è definire una strategia per supportare il progettista nella (re)ingegnerizzazione di un'applicazione nell'ottica di renderla fruibile sotto forma di micro-servizi. Le linee guida, in estrema sintesi, mirano a individuare all'interno di un'applicazione i servizi potenzialmente estraibili, che potrebbero essere pubblicati in modalità esplicita.

Le linee guida fanno riferimento alla definizione di servizio che, secondo i dettami dell'ingegneria, è legata ai concetti di Service Oriented Architecture (SOA). Tra le molteplici definizioni in tale ambito quella fornita da "The Open Group":

" A service:

- Is a logical representation of a repeatable business activity that has a specified outcome (e.g., check customer credit, provide weather data, consolidate drilling reports);
- Is self-contained;
- May be composed of other services;
- Is a "black box" to consumers of the service".

Tale definizione fa esplicitamente riferimento al concetto di "business activity" proprio per sottolineare come il servizio debba essere considerato più dal punto di vista del compito di business che dal punto di vista tecnologico. Sebbene tale definizione non comprenda esplicitamente aspetti tecnologici, è necessario considerare che i micro-servizi devono essere completamente allineati alle proprietà previste dal manifesto delle SOA. Tale manifesto definisce un insieme di proprietà e prevede che ogni servizio debba essere progettato con specifici canoni:

1. *Standardized service contract*: i servizi devono definire un contratto formale di invocazione in modo da descrivere in modo univoco e predicibile le proprie modalità d'interazione con il client;
2. *Service loose coupling*: i servizi devono essere progettati per interagire in modo debolmente accoppiato;

3. *Service abstraction*: i servizi realizzano un'astrazione della logica sottostante. Il servizio, infatti, espone una sua astrazione e la sola parte del servizio che è visibile all'esterno è ciò che è esposto tramite la sua descrizione (Standardized service contract). La logica sottostante (l'implementazione del servizio) è invisibile e irrilevante per chi vuole richiedere il servizio;
4. *Service reusability*: i servizi sono progettati per sostenere un potenziale riuso, indipendentemente dal fatto che esistano, o meno, delle opportunità immediate di riuso;
5. *Service autonomy*: l'autonomia di un servizio riguarda la sua implementazione e il suo ambiente di esecuzione. L'implementazione di ogni servizio deve essere autonoma. Inoltre, ogni servizio deve avere capacità di auto-governo entro il suo ambiente di esecuzione, inclusi i dati.
6. *Service statelessness*: al fine di aumentare il livello di disaccoppiamento tra i servizi, gli stessi devono essere il più possibile "stateless" e devono essere progettati per massimizzare questa assenza di stato, eventualmente delegando la sua gestione a oggetti specifici dell'architettura di esecuzione;
7. *Service discoverability*: il descrittore dei servizi (interfaccia pubblica) deve essere reperibile e comprensibile da chi vuole utilizzare il servizio stesso;
8. *Service composability*: i servizi possono essere utilizzati per comporre altri servizi. Quindi, la logica applicativa può essere rappresentata a diversi livelli di granularità. Tale scelta consente: (i) di aumentare la riusabilità e la creazione di livelli di astrazione; (ii) di aumentare la componibilità dei servizi; (iii) di far evolvere rapidamente applicazioni e processi di business.

Il micro-servizio oltre a dover essere aderente al SOA Manifest, riportato in precedenza, deve essere reso fruibile come componente di business. In altre parole, l'atomicità del codice da racchiudere in un micro-servizio non dipende solo dai dettami dell'ingegneria del software quali disaccoppiamento, atomicità ecc., ma grande importanza ricopre anche il valore intrinseco che tale micro-servizio avrà per l'utilizzatore della piattaforma che dovrà comporlo con altri servizi o utilizzarlo al fine di creare un'applicazione su più canali. Alla luce di quanto scritto, è evidente che la definizione di micro-servizio più aderente alle specificità di progetto è quella di componente di business. Già nel 2006, IBM aveva definito il modello CBM (Component Business Modeling) che permetteva di rappresentare un'organizzazione in termini di Business Component, cioè moduli auto-consistenti che ricoprono un preciso ruolo nell'enterprise system. Sulla base di tale approccio, IBM ha definito la sua offerta Component Business Modeling Services (SM) che, ad oggi, comprende le strategie e le tecnologie IBM orientate a supportare le aziende nel proprio business attraverso la corretta gestione dei propri applicativi e infrastrutture IT.

Nella stessa prospettiva, SAP© nella sua release R/3 definisce esplicitamente il concetto di business component che è utilizzato come sinonimo di business application. Un business component (business application) è composto da business object ognuno dei quali incapsula una specifica business function. Le business function sono definite da Porter in [Porter 1985] come "the technologically and economically distinct activities a company performs to do business." Anche la Comunità Europea, tramite Eurostat, definisce le Business Function: "they are the activities carried out by an enterprise; they can be divided into core functions and support functions".

E' evidente, quindi, il parallelismo con le definizioni di processi di business, siano essi primari o di supporto. In tale ottica, un'organizzazione può essere vista come un insieme di Business Component (BC) composti da Business Function (BF) che a loro volta (sfruttando il parallelismo con i business process) sono composte da attività elementari (o task). Una Business Function è, quindi, un insieme orchestrato di servizi elementari (micro-servizi) intesi come Business Activity (BA) cosi come definito nell'architettura SOA.

Dal punto di vista applicativo, si deve notare che, in genere, un business process richiede differenti applicazioni (cioè Business Component) per raggiungere il proprio obiettivo. È possibile, quindi, affermare

che i Business Component possono essere condivisi tra i più business process e che i business process eseguono/attraversano trasversalmente differenti Business Component.

Per comodità, la Tabella 1 riporta una sintesi della comune semantica dei vari termini.

Tabella 1: Glossario dei termini per sistemi a servizi

Glossario BPM	Nomenclatura SOA	Definizione
Business Activity	Servizio Elementare / Task / Micro-servizio	Componente software isolato e autonomo.
Business Function	Servizio Composito	Orchestrazione di due o più servizi (micro-servizi o compositi).
Business Component	Applicazione	Sistema software realizzato tramite servizi.
Business Process	Processo	Descritti in BPMN e utilizzati per rappresentare un'orchestrazione di più servizi.

Sulla base di tali considerazioni, è evidente come l'obiettivo di individuare i servizi dai requisiti di un'applicazione multi-canale possa essere tradotto nell'individuare le Business Function (Servizi Compositi) all'interno dei Business Component (Applicazioni). Allo stesso modo, è possibile individuare i micro-servizi o Business Activity in accordo con gli obiettivi dell'analisi AWARE.

Da questo punto in poi, si utilizzeranno come sinonimi i termini "service" e "business function", intendendo semanticamente "elementi del sistema software che implementano specifiche funzioni di business, che hanno un valore intrinseco per l'utilizzatore finale, che possono essere composti al fine di formare applicazioni, che rispettano i dettami del manifesto SOA".

In tale ottica, l'individuazione di servizi può essere principalmente basata sulla documentazione di analisi dei requisiti (modello AWARE, scenari d'uso, casi d'uso, glossario di dominio, modello di dominio, modello dei dati). Particolarmente utile è l'analisi e l'annotazione dei modello AWARE contenente i goal e i sotto-goal, con l'obiettivo di indicare anche il valore e le funzioni che implementano gli elementi di business. Sulla scorta di tali considerazioni, si propone la strategia riportata in Figura 13.

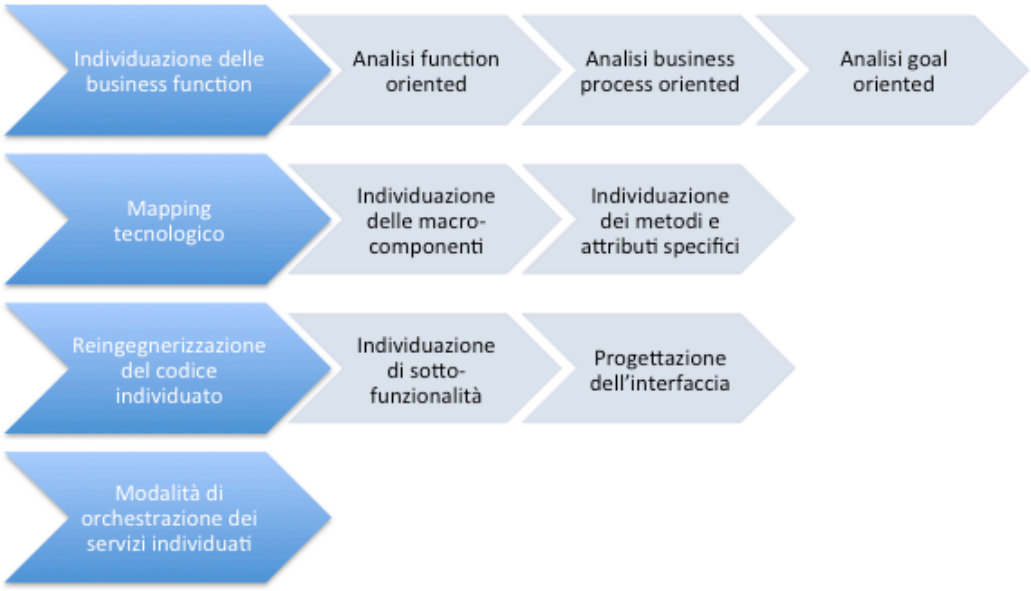


Figura 13: Strategia per l'individuazione dei servizi dall'analisi dei processi

Più in dettaglio, si propongono i seguenti passi:

1. Individuazione delle business function. Questa fase prevede che siano individuate quelle funzionalità che potrebbero essere estratte come servizi dalla documentazione di analisi dei requisiti. Tale analisi deve essere fatta rispettando le definizioni date in precedenza.
2. Mapping tecnologico delle business function. In questa fase, si esegue un mapping tra i servizi individuati e l'architettura software dell'applicazione multi-canale. Il punto di partenza di tale mapping è la documentazione tecnica disponibile, come diagrammi delle classi e, in particolare, il class model estratto dal domain model della base di dati.
3. Reingegnerizzazione del codice individuato in ottica di servizio dell'architettura finale. Questa fase prevede, che sulla base del codice individuato al punto precedente, se ne compia un'analisi e una reingegnerizzazione al fine di:
    a. Individuare elementi/sotto-funzionalità del servizio identificato che potrebbero essere ulteriormente decomposti, isolati ed esposti come servizi a sé stanti;
    b. Progettare l'interfaccia che dichiara i metodi che possono essere invocati dall'utilizzatore per chiamare/comporre il servizio pubblicato.
4. Definizione delle modalità di orchestrazione dei servizi individuati.

3.1.1 Individuazione delle business function e delle business activity

Obiettivo di questa fase è individuare dai requisiti quali sono i componenti/funzionalità dell'applicazione che potrebbero essere pubblicati all'interno della piattaforma come servizi compositi e come micro-servizi. Poiché spesso l'estrazione di micro-servizi deriva da un percorso di modernizzazione di sistemi legacy, è opportuno definire un processo che:

- Permetta la scomposizione dell'applicazione in business function/business activity, ognuna delle quali abbia le caratteristiche descritte in precedenza;
- Assicuri la possibilità di orchestrare tali business function/business activity per realizzare, all'interno della piattaforma di produzione, le funzionalità disponibili nell'applicazione di partenza.

Questa fase, quindi, deve consentire al progettista di individuare i servizi, ma anche definirne le mutue relazioni. In merito all'individuazione dei servizi, si propongono tre strategie:

- Analisi function oriented: permette di individuare i servizi/micro-servizi in un'ottica di decomposizione funzionale dell'applicazione esistente;
- Analisi business process oriented: permette di individuare i servizi/micro-servizi sulla base dei processi di business presenti nell'applicazione esistente;
- Analisi goal oriented: permette di individuare i servizi/micro-servizi sulla base dei requisiti dell'applicazione.

Le tre strategie non sono mutuamente esclusive, ma la scelta di quale strategia primaria utilizzare, eventualmente integrata con le altre, dipende essenzialmente dalla disponibilità di documentazione di analisi e dalla qualità della stessa.

3.1.1.1 Analisi function oriented

Questo tipo di analisi prevede che si studino i requisiti della nuova applicazione oppure l'applicazione esistente esplorando/ispezionando il suo modello IDM, oppure i suoi mockup di interfaccia oppure la sua interfaccia utente legacy al fine di individuare le funzionalità che possano essere estratte come servizi. Tale analisi può essere svolta attraverso sessioni dimostrative eseguite da personale esperto o attraverso l'analisi di documenti come il manuale utente dell'applicazione e la descrizione dei casi d'uso.

L'analisi function oriented tende a enfatizzare nella definizione dei servizi/micro-servizi gli aspetti funzionali dell'applicazione e conduce a individuare, in prima istanza, servizi a grana grossa, cioè servizi compositi. Per tal motivo è indispensabile, nella fase di reingegnerizzazione, valutare con un'analisi più approfondita la possibilità di decomposizione funzionale dei servizi individuati in micro-servizi, ponendo l'attenzione su quei servizi che:

- Possono essere ulteriormente isolati e pubblicati come micro-servizi autonomi, eseguibili da più canali;
- Possono essere isolati poiché fanno riferimento a fonti di dati esterne al servizio individuato.

Si deve notare che, con questo tipo di tecnica, le modalità di orchestrazione dei servizi non sono direttamente ricavabili, ma devono essere ricostruite dopo aver individuato tutti i servizi/micro-servizi e le mutue relazioni. Questa ricostruzione può essere ricavata esaminando i casi d'uso dell'applicazione, i mockup funzionali, gli scenari dai quali è stata ottenuta la documentazione AWARE.

3.1.1.2 Analisi process oriented

L'analisi process oriented parte dal presupposto che sia disponibile l'elicitazione dei processi dell'applicazione multi-canale. Tuttavia, qualora l'elicitazione non sia presente, è possibile immaginare di modellare i processi dalla descrizione degli scenari d'uso e dall'analisi AWARE. I business process derivabili rappresentano una porzione dei processi primari o di supporto, a un livello di dettaglio che rende facilmente orchestrabili i processi stessi.

Dal modello BPMN dei processi, in cui sono evidenti le attività e i sub-process, è possibile individuare in modo agevole i servizi/micro-servizi che possono essere estratti. Questo tipo di tecnica si adatta molto bene alle applicazioni che implementano workflow o transazioni, ma può essere applicata in modo trasversale a tutte le tipologie di dialogo utente/sistema. I vantaggi dell'approccio process oriented sono notevoli:

- Poiché i processi, per definizione, sono strettamente collegati al business, è chiaro che l'individuazione dei processi primari comporta l'identificazione diretta delle Business Function (servizi) che sono riconducibili ai sub-process individuati o a gruppi di task;
- La modellazione dei processi e, quindi, dei servizi da essi derivati, comporta la definizione delle attività e del flusso che le connette. È evidente, quindi, che tale flusso definisce e/o dà forti indicazioni sulle modalità di orchestrazione dei servizi stessi;
- La definizione dei singoli task di cui il processo è composto, fornisce indicazioni dettagliate su quali saranno i micro-servizi e le operazioni in essi contenute, e che dovranno essere esposte. Infatti, poiché i task sono elementi di un processo che eseguono un determinato compito elaborando informazioni in input e generando output, con molta probabilità ogni task dovrà essere invocato dal client che utilizza il servizio per eseguire le elaborazioni previste dallo specifico elemento. Riguardo al "valore" comunicato all'utente, al suo livello di atomicità e isolamento concettuale, il task può sottintendere un micro-servizio oppure un'operazione contenuta in un micro-servizio;
- La definizione dei "message element" BPMN dei processi permette immediatamente di individuare le relazioni che i processi (e, quindi, i servizi) potrebbe avere con altri processi (servizi).

L'approccio process oriented presenta, di converso, l'inconveniente che la documentazione di processo necessaria per eseguire l'analisi richiesta, spesso non è disponibile in forma esaustiva. Quindi, l'analisi process oriented richiede uno sforzo iniziale per modellare i processi sottostanti all'applicazione da cui ricavare i servizi/micro-servizi.

Si deve osservare che questo tipo di analisi (in cui sono modellati i singoli task) potrebbe portare a definire una serie di servizi a grana troppo fine. Per ovviare a tale problematica, nella fase di reingegnerizzazione è indispensabile eseguire successive riflessioni per valutare se sia opportuno accorpare i servizi inizialmente individuati.

3.1.1.3 Analisi goal oriented

L'analisi goal oriented ha come input i requisiti dell'applicazione e parte dal presupposto che un goal di alto livello sia direttamente connesso con una o più attività di business. È chiaro, quindi, che un goal possa essere associato al servizio che è necessario per soddisfarlo. Questo tipo di strategia è la più vicina alla progettazione dell'architettura informativa dell'applicazione multi-canale. Infatti, a differenza dell'analisi dei processi e dell'analisi delle funzioni, che hanno come punto di partenza elementi concreti del software (processi o funzionalità), l'analisi dei goal prende in considerazione gli obiettivi di business dell'applicazione, ancora prima che siano declinati in specifiche strutture software. Proprio per la sua generalità, l'analisi goal oriented è sicuramente la meno costosa, poiché permette di individuare i servizi senza esaminare documentazione tecnica, che spesso è insufficiente o richiede gravosi sforzi per ricostruirla. La contropartita è la povertà di dettagli nella descrizione dei business process; infatti, la definizione dei dati in input e delle elaborazioni di un servizio rischia spesso di rimanere a un livello molto astratto e approssimativo.

3.1.1.4 Descrizione delle business function e delle business activity

Indipendentemente dal tipo di analisi che s'intende eseguire (function, process o goal oriented), l'output di questa prima fase è la definizione di una tabella dei servizi estratti, strutturata come in Tabella 2.

Tabella 2: Tabella dei servizi estratti

Nome del servizio	Descrizione del servizio	Informazioni elaborate	Metodi pubblicati

Dove:

- Nome del servizio: è il nome del servizio (composito, elementare) che dovrà essere esposto;
- Descrizione del servizio: è la descrizione della funzionalità del servizio;
- Informazioni elaborate: descrive le informazioni di input e di output del servizio;
- Metodi pubblicati: è un'ipotesi iniziale delle operazioni esposte dal micro-servizio (solo per i servizi elementari).

Come abbiamo già detto, le differenti strategie di analisi permettono di elicitare i servizi a diversi livelli di accuratezza. La Tabella 3 mostra, su una scala da uno a cinque (1-scarso, 2-sufficiente, 3-discreto, 4-buono, 5-ottimo), i livelli di dettaglio che si ipotizza di poter raggiungere con le differenti strategie.

Tabella 3: Livelli di accuratezza di elicitazione dei servizi

Strategia	Descrizione	Informazioni elaborate	Metodi pubblicati	Facilità di orchestrazione	Agilità di analisi
Function oriented	4	4	5	3	3
Process oriented	5	5	4	5	1
Goal oriented	3	2	3	1	5

Dove:

- Descrizione: esprime la qualità della descrizione dei servizi che è possibile raggiungere con la specifica tecnica;
- Informazioni elaborate: esprime la quantità di informazioni (tecniche e non tecniche) che si devono considerare per individuare i servizi;
- Metodi pubblicati: esprime l'efficacia del metodo nell'individuare l'interfaccia del servizio elicitato;
- Facilità di orchestrazione: esprime l'efficacia del metodo nel supportare il progettista nella definizione delle modalità di orchestrazione;
- Agilità di analisi: esprime una valutazione sulla rapidità e la semplicità con cui è possibile eseguire l'analisi complessiva con tale strategia.

Le tre strategie qui illustrate possono essere utilizzate in modo integrato allo scopo di offrire all'analista una metodologia completa. Infatti, data un'applicazione la cui unica documentazione sia composta dagli scenari d'uso e dall'analisi AWARE, è possibile:

- Fare una prima analisi nella prospettiva illustrata dalla strategia function oriented;
- Disegnare il flusso dei processi a partire dai casi d'uso individuando i sub-process e i gruppi di task;

Eseguire un incrocio evidenziando in una tabella a due dimensioni (

- Tabella 4) la corrispondenza tra le business function identificate con il primo metodo e le business function individuate con il secondo metodo.

Tabella 4: Corrispondenza tra sub-process e business function

BF/Sub process	Sub-process 1	Sub-process 2	Sub-process 3
BF1	X		X
BF2	X	X	

Tabella 4 fornisce un'idea abbastanza precisa della copertura dei servizi individuati rispetto all'applicazione multi-canale e fornisce, inoltre, un'indicazione sulla riusabilità dei servizi.

3.1.2 Mapping delle business function e delle business activity sul codice di sistemi legacy

Qualora l'oggetto di lavoro sia la modernizzazione di un sistema legacy in termini di applicazione multi-canale, dopo aver individuato i servizi che devono essere pubblicati in produzione, è necessario determinare le componenti del codice dell'applicazione originale che implementano tali servizi. Si tratta di un esercizio di mapping dei servizi sulle funzionalità erogate dall'applicazione legacy.

Poiché un servizio include business logic, dati, parametri in input e output, questo mapping coinvolge differenti livelli architetturali. L'operazione può richiede diversi passaggi per ogni servizio. In particolare, definito il servizio su cui s'intende fare il mapping, è necessario:

1. Individuare le macro-componenti nel codice sorgente legacy. Questa fase prevede che si determinino le classi e le entità indispensabili per l'esecuzione del servizio. In merito all'individuazione delle classi, è necessario fare una distinzione tra le classi che codificano la business logic del servizio e quelle di utilità (helper). In particolare, le classi di utilità sono quelle che implementano i servizi di logging, funzioni di debug, reperimenti di parametri, ecc. In genere, se la progettazione del diagramma delle classi è stata eseguita in modo corretto, le classi di utilità sono impiegate dalle altre classi di business e non richiedono di avere metodi esposti sulla piattaforma. Per tal motivo, le classi di utilità non sono oggetto della successiva fase di reingegnerizzazione.
2. Individuare i metodi e attributi specifici del micro-servizio. Si deve evidenziare, infatti, che le classi individuate al passo precedente potrebbero contenere più metodi/attributi di quelli effettivamente utilizzati dal servizio. Quindi, al fine di isolare il codice specifico del micro-servizio stesso, è necessario estrarre le sole informazioni indispensabili.

L'output del mapping della business logic è un insieme di tabelle (una per servizio) con le informazioni riportate in Tabella 5.

Tabella 5: Dettaglio dei servizi

Nome del servizio			
Classe	Attributi	Metodi	Metodi Pubblici

Dove:

- Classe: la classe di business coinvolta nell'esecuzione del servizio;
- Attributi: sono gli attributi della classe di business che caratterizzano il servizio;
- Metodi: sono i metodi della classe di business che caratterizzano il servizio;
- Metodi pubblici: indica i metodi della classe che devono essere esposti all'interno della piattaforma migrata.

4. Identificazione dei micro-servizi nell'ambito del progetto EasyPAL

Nell'ambito del presente progetto sono state utilizzate le strategie function oriented e goal oriented, a partire dall'analisi e specifica dei requisiti (per cui si rimanda al Deliverable D.A3), al fine di identificare servizi compositi (business function) più legati alle funzionalità e servizi elementari (business activity).

In accordo con le linee guida precedentemente descritte, nella tabella sottostante è riportata la definizione dei servizi astratti identificati.

Nome del servizio	Descrizione del servizio	Informazioni elaborate	Metodi pubblicati
Account	Servizio di autenticazione e accesso alla piattaforma EasyPAL, per cittadini, operatori e amministratori. L'accesso può essere effettuato, mediante SPID (per i cittadini) e mediante username e password (per operatori e amministratori).	Credenziali inserite dall'utente	GET api/Account/LoginSPID?client_id={client_id}&IDPName={IDPName}&session_id={session_id} POST api/Account/ControllaToken POST api/Account/EsitoLoginSPID POST api/Account/LoginOperatore
Documentation	Servizio per la consultazione e il download di tutti i documenti del cittadino, inviati o ricevuti, da uno dei comuni aderenti al progetto EasyPAL.	Token di autenticazione e codice fiscale dell'utente e dell'operatore, oltre alle informazioni sul documento in caso di richiesta di dwnload.	POST api/Doc/getAllDocumenti POST api/Doc/downloadFile
Anagrafe	Servizio per la richiesta ed il conseguente rilascio (in tempo reale) di certificati anagrafici (es. nascita, matrimoni, residenza, etc.)		POST api/Anpr/getSchedaSoggetto POST api/Anpr/getComponentiFamigliaConvivenza POST api/Anpr/getCertificato

Di seguito si dettagliano ulteriormente i metodi dei servizi di autenticazione e documentazione.

Nome del servizio	Metodi pubblicati	**Descrizione**
Account	GET api/Account/LoginSPID?client_id={client_id}&IDPName={IDPName}&session_id={session_id}	Servizio di reindirizzamento verso l'Authentication Server (AS) SPID configurato. In caso di login positivo, l'AS SPID richiamerà il metodo EsitoLoginSPID per verificarne la coerenza e generare il token.
	POST api/Account/ControllaToken	Servizio per il controllo della validità del token di autenticazione. Sono passati in post, in formato json, i parametri cf e token. Il servizio ritorna true se il controllo è andato a buon fine ed il token è valido, in caso contrario il servizio ritorna false.
	POST api/Account/EsitoLoginSPID	Servizio per la verifica dell'esito di login mediante SPID. Sono passati in post, in formato json, i parametri cf, client_id, session_id e scope. Il servizio ritorna un url su cui si viene reindirizzati a seconda se il login tramite SPID abbia avuto esito positivo o negativo.
	POST api/Account/LoginOperatore	Servizio per l'autenticazione del'operatore mediante username e password. Sono passati in post, in formato json, i parametri username, password, client_id, scope e session_id. Il servizio ritorna l'oggetto EasyPAL_PresentationDLL.LoginParameter avvalorato con le seguenti informazioni dell'operatore: idAccount, access_token, cognome, nome, username, codicefiscale, messaggioErrore, session_id.
Documentation	POST api/Doc/getAllDocumenti	Servizio per la consultazione di tutti i documenti di un utente. Sono passati in post, in formato json, i parametri cf, token e cfInVisualizzazione. Il servizio ritorna l'elenco dei documenti associati al codice fiscale/partita iva passato come parametro con le informazioni necessarie all'utente per identificare univocamente ogni documento.
	POST api/Doc/downloadFile	Servizio per il download dei documenti. Sono passati in post, in formato json, i parametri tipoDocumento, pkDocumento, pkConfigurazione, nomefile, token e cf. Il servizio ritorna il file selezionato.

Le informazioni ottenute a seguito di autenticazione SPID con esito positivo, sono restituite in forma cifrata, per essere decifrati lato applicazione Web o mobile.

In Appendice A – Esempi di chiamate ai micro-servizi si riportano alcuni esempi di chiamate ai servizi descritti.

5. Validazione dell'architettura e del modulo Cloud Hub

In questa sezione verrà descritto nel dettaglio il caso d'uso specifico di "Accesso cittadino tramite SPID", particolarmente interessante per validare l'intera architettura perché riguarda sicuramente tutte le app lato cittadino ed utilizza i diversi moduli di identificazione OAuth e Cloud Hub.

Nella figura seguente si riporta il flusso informativo del caso d'uso, con evidenza dei diversi moduli interessati e dei parametri scambiati.

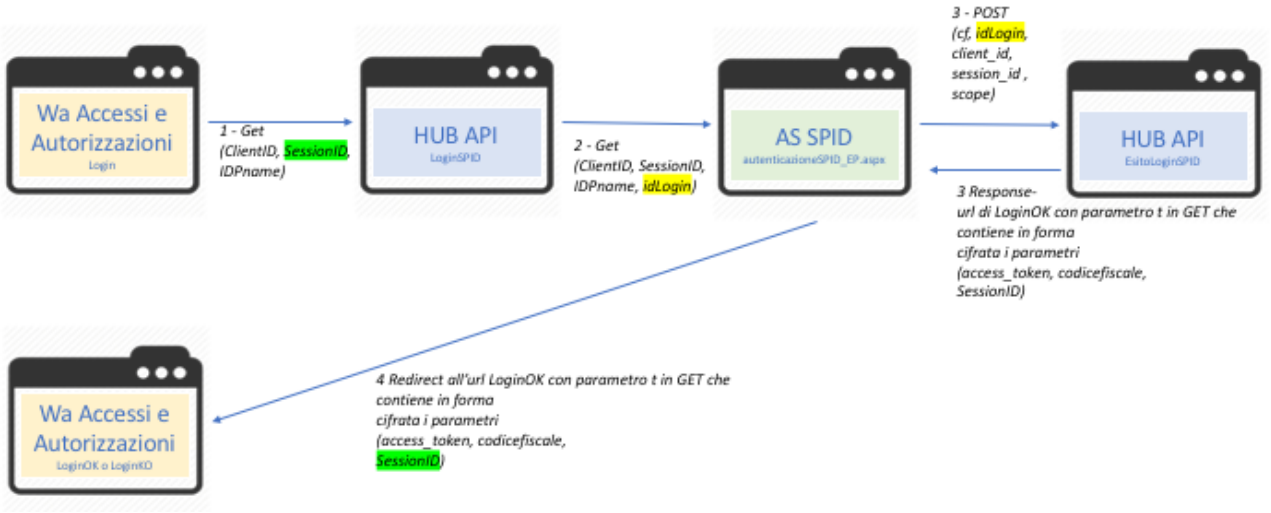


Figura 14: Flusso informativo caso d'uso "Accesso cittadino tramite SPID"

Nella figura sono evidenziati in verde e in giallo i parametri oggetto di verifica di sicurezza.

Questo il dettaglio delle chiamate:

1) Dalla Web Application di «Gestione Accessi e Autorizzazioni» l'utente sceglie l'accesso a mezzo SPID selezionando l'IDP (es. Poste)

2) L'utente viene reindirizzato al metodo REST web di HUB API «**LoginSPID**», a cui vengono passati in GET i parametri clientID e sessionid e idpname. Questo a sua volta effettua il redirect all'Authentication Server (AS) SPID configurato, fornendo in get i parametri idLogin, clientID e sessionid e idpname.

3) In caso di Login positivo l'AS SPID richiama il web method di HubAPI «**EsitoLoginSPID**» *fornendo in post* cf, idLogin, client_id, session_id e scope. Il metodo verifica la coerenza dell'IDLogin (otp che può essere usato una sola volta), e se il test viene superato genera il token e lo scope, restituendo infine l'url di LoginOK con il parametro t in GET che contiene in forma cifrata i parametri access_token, codicefiscale e SessionID. La cifratura del parametro t avviene mediante una chiave segreta legata al parametro client_id condivisa in fase di configurazione iniziale tra HUB API ed il client «Wa Accessi e Autorizzazioni».

4) L'AS SPID effettua il redirect all'url di LoginOK inviando il parametro t in GET, il quale contiene in forma cifrata i parametri access_token, codicefiscale e SessionID.

5) La Web Application di «Gestione Accessi e Autorizzazioni» decodifica il parametro t, verifica la coerenza del parametro SessionID e riconosce le autorizzazioni concesse.

Nel caso in cui al punto 3 l'autenticazione non andasse a buon fine, l'utente verrebbe reindirizzato alla pagina LoginKO. Si sottolinea che le pagine LoginOK, LoginKO ed il client secret sono oggetto di configurazione coerente tra CLIENT – Web Application di «Gestione Accessi e Autorizzazioni», APP mobile, ecc. – e HUB API.

1. Appendice A – Esempi di chiamate ai micro-servizi

In questa sezione si riportano le chiamate di esempio dei micro-servizi identificati mediante la metodologia proposta e l'analisi dei requisiti effettuata.

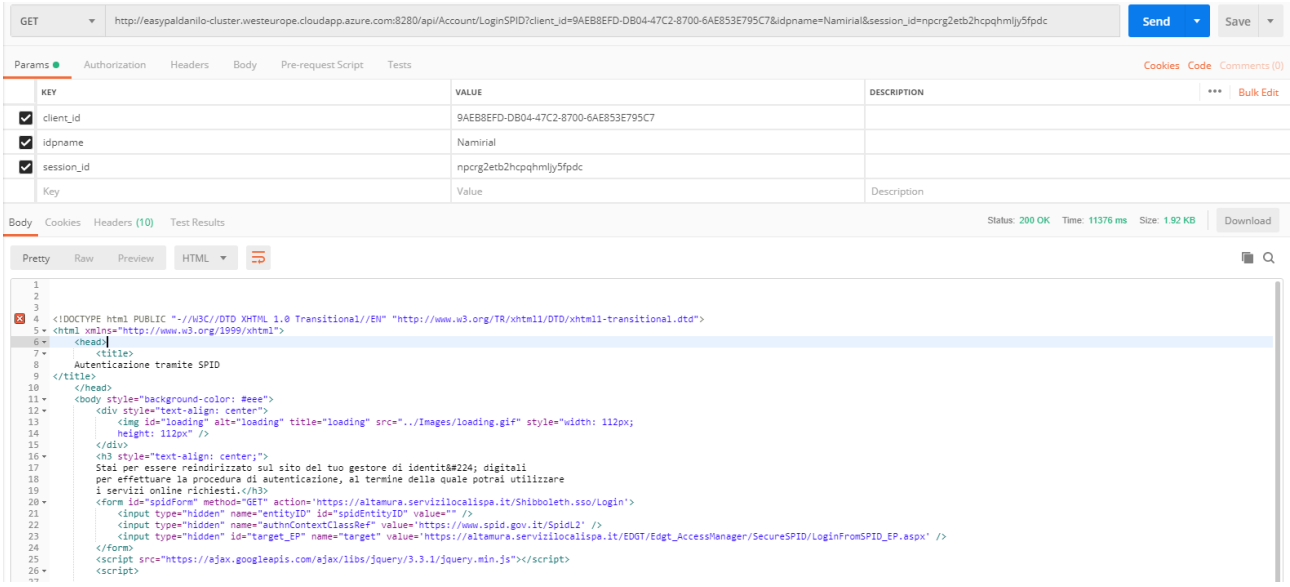
1.1 Account

Servizio di autenticazione e accesso alla piattaforma EasyPAL, per cittadini, operatori e amministratori. L'accesso può essere effettuato, mediante SPID (per i cittadini) e mediante username e password (per operatori e amministratori).

1.1.1 LoginSPID

Servizio di reindirizzamento verso l'Authentication Server (AS) SPID configurato.

In caso di login positivo, l'AS SPID richiamerà il metodo EsitoLoginSPID per verificarne la coerenza e generare il token.

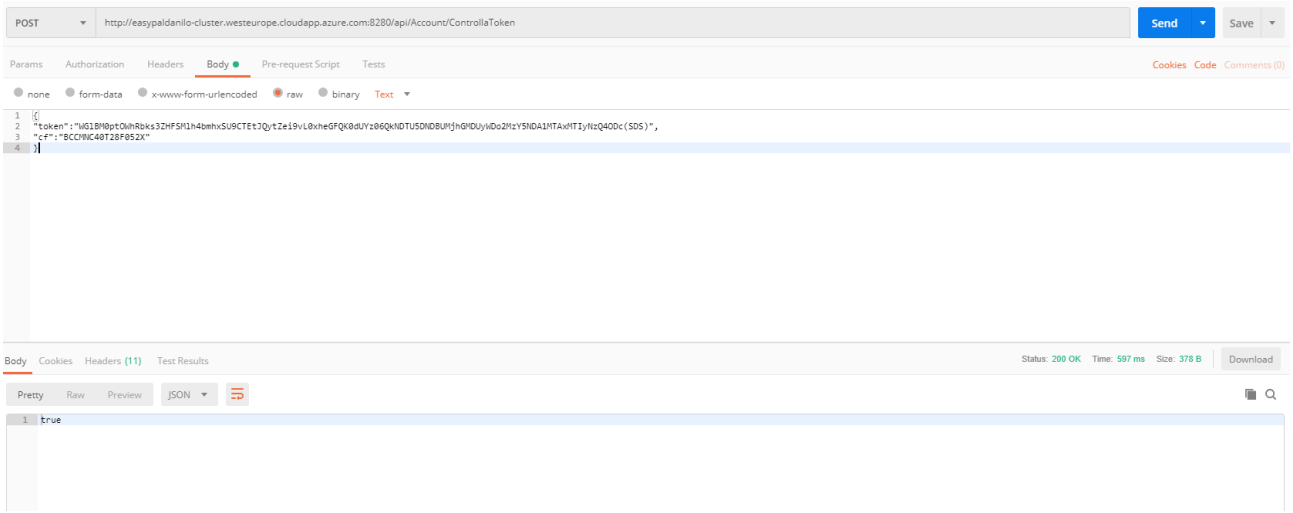


1.1.2 ControllaToken

Servizio per il controllo della validità del token di autenticazione.

Sono passati in post, in formato json, i parametri cf e token.

Il servizio ritorna true se il controllo è andato a buon fine ed il token è valido, in caso contrario il servizio ritorna false.

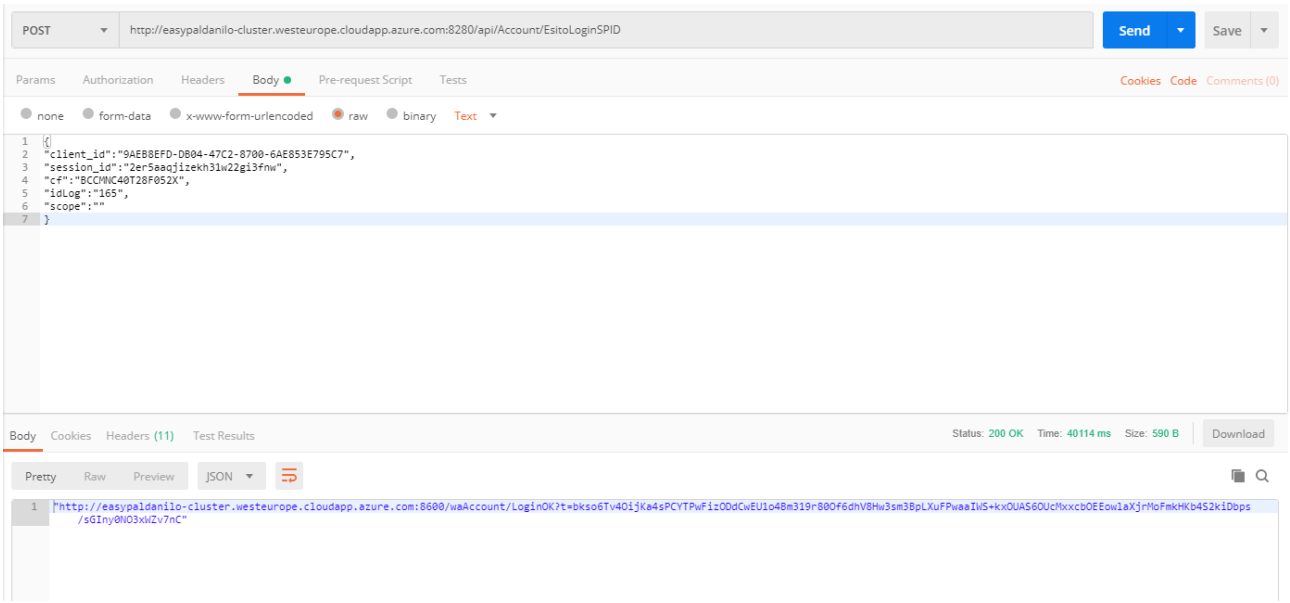


1.1.3 EsitoLoginSPID

Servizio per la verifica dell'esito di login mediante SPID.

Sono passati in post, in formato json, i parametri cf, client_id, session_id e scope.

Il servizio ritorna un url su cui si viene reindirizzati a seconda se il login tramite SPID abbia avuto esito positivo o negativo.

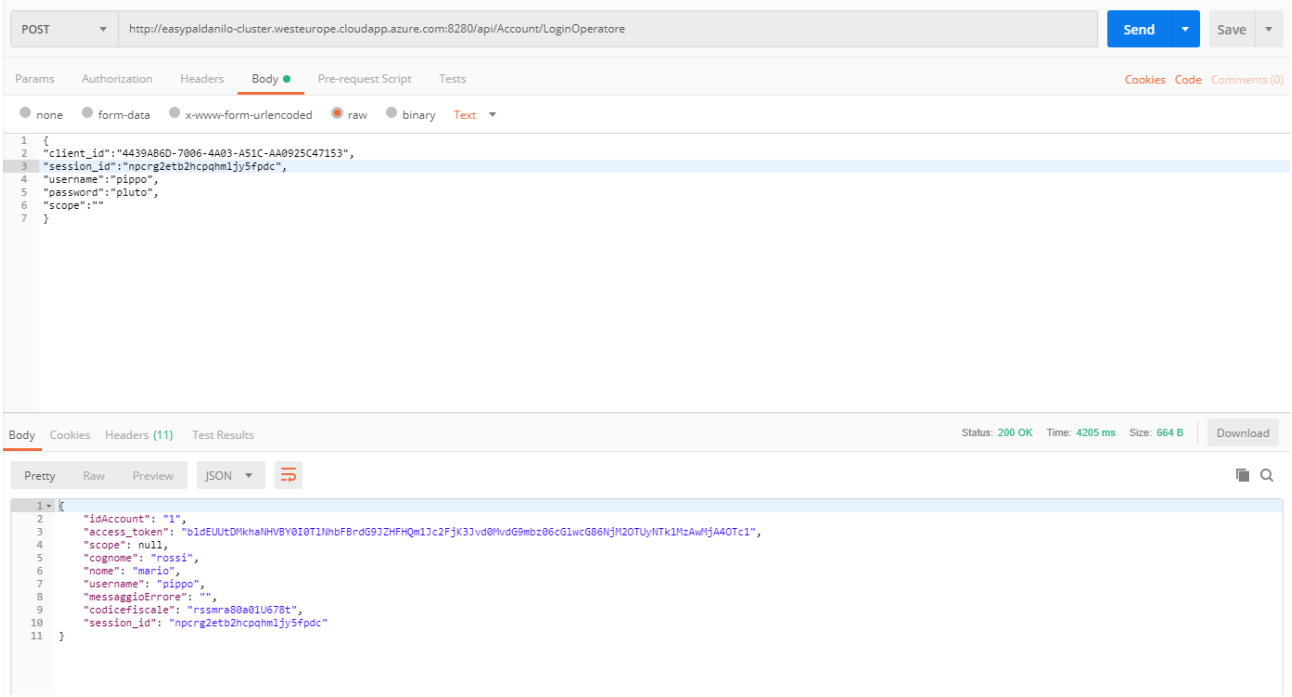


1.1.4 LoginOperatore

Servizio per l'autenticazione dell'operatore mediante username e password.

Sono passati in post, in formato json, i parametri username, password, client_id, scope e session_id.

Il servizio ritorna l'oggetto EasyPAL_PresentationDLL.LoginParameter avvalorato con le seguenti informazioni dell'operatore: idAccount, access_token, cognome, nome, username, codicefiscale, messaggioErrore, session_id.



1.2 Doc

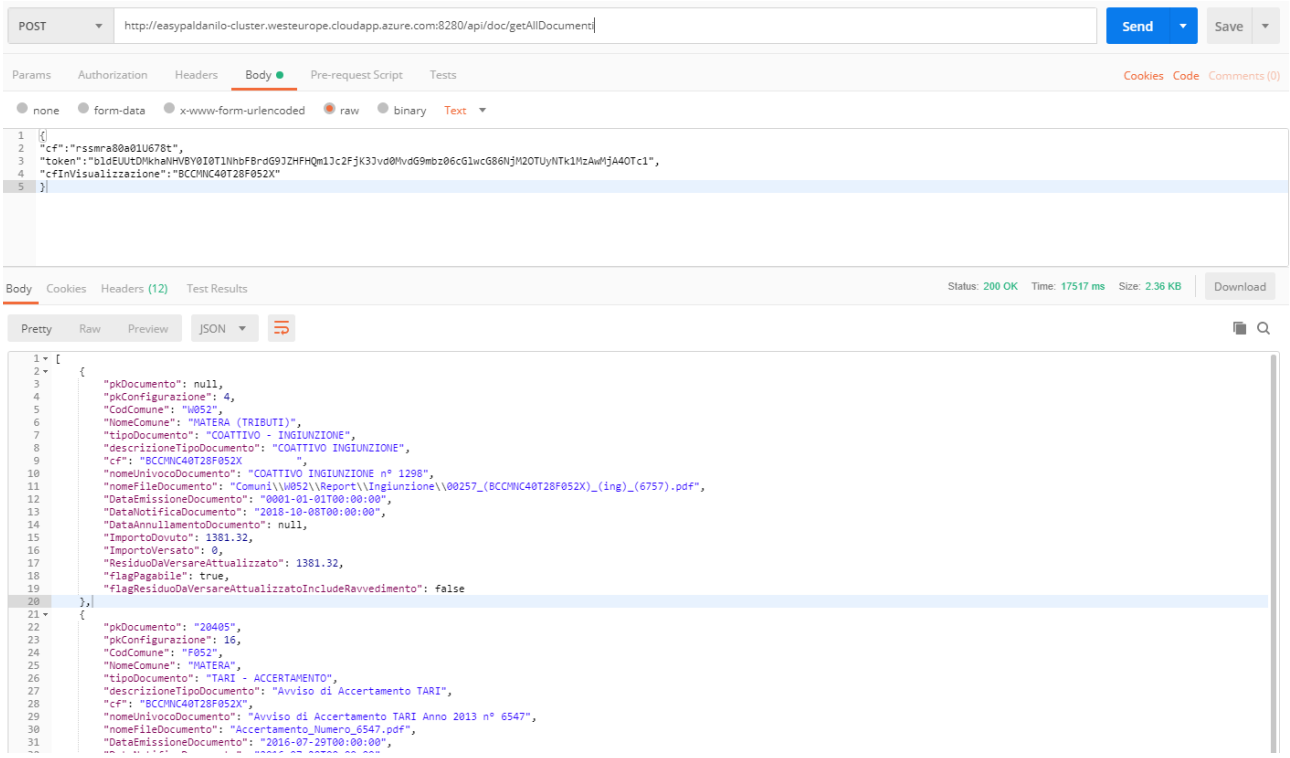
Servizio per la consultazione e il download di tutti i documenti del cittadino, inviati o ricevuti, da uno dei comuni aderenti al progetto EasyPAL.

1.2.1 getAllDocumenti

Servizio per la consultazione di tutti i documenti di un utente.

Sono passati in post, in formato json, i parametri cf, token e cfInVisualizzazione.

Il servizio ritorna l'elenco dei documenti associati al codice fiscale/partita iva passato come parametro con le informazioni necessarie all'utente per identificare univocamente ogni documento.

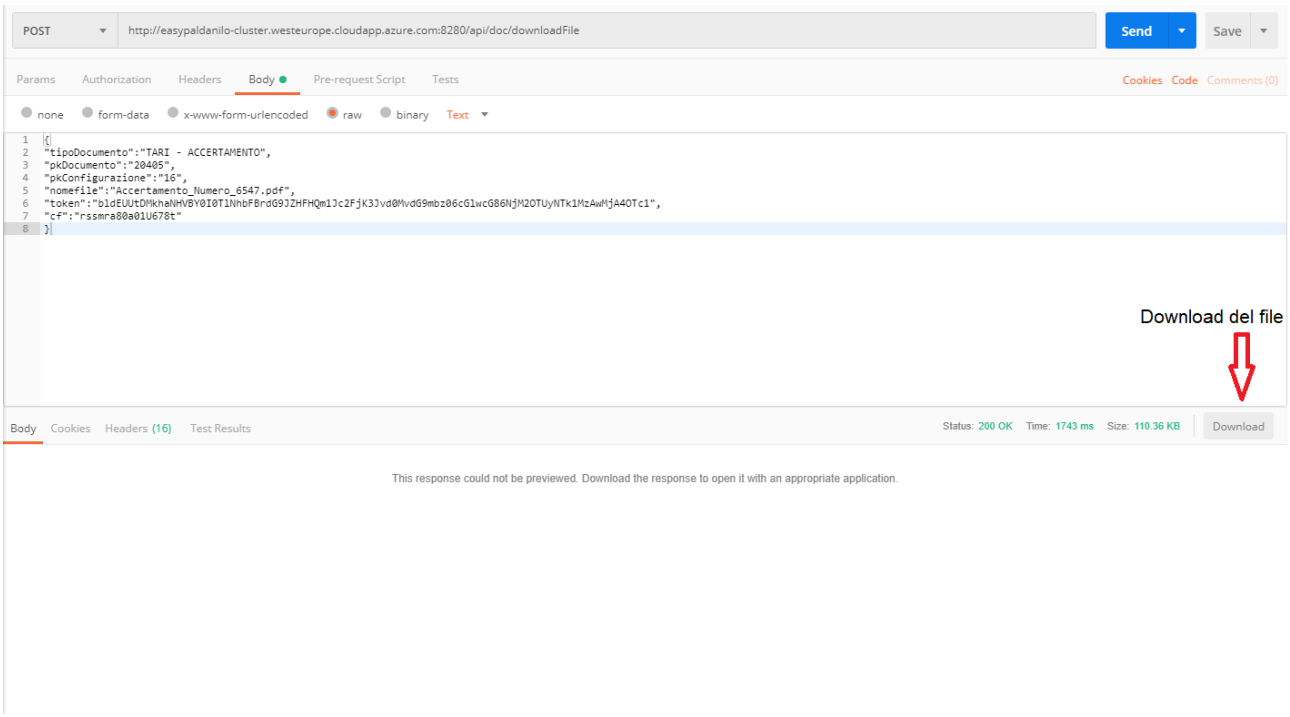


1.2.2 downloadFile

Servizio per il download dei documenti.

Sono passati in post, in formato json, i parametri tipoDocumento, pkDocumento, pkConfigurazione, nomefile, token e cf.

Il servizio ritorna il file selezionato.



2. Bibliografia

[Baresi 2003] L. Baresi, D. Bianchini, V. De Antonellis, M. G. Fugini, B. Pernici, and P. Plebani. Context-aware composition of e-services. In VLDB TES Workshop, pages 28–41, 2003.

[Pernici 2006] B. Pernici, Mobile Information Systems: Infrastructure and Design for Adaptivity and Flexibility, Springer-Verlag 2006, XVI, ISBN 978-3-540-31008-2.

[Lewis 2012] Lewis, J. Micro services - java, the unix way, 33rd Degree Conference for Java Masters. Krakow, Poland, 2012.

[George 2012] George, F. Microservicearchitecture. YOW!2012, Melbourne, Australia, 2012.

[Lewis 2014] Lewis, J., Fowler, M.: Microservices: a definition of this new architectural term (2014). http://martinfowler.com/articles/microservices.html.

[Hassan 2016] Hassan, S., Bahsoon, R.: Microservices and their design trade-offs: a self-adaptive roadmap. In: IEEE International Conference on Services Computing (SCC), pp. 813–818. IEEE, 2016.

[Balalaie 2016] Balalaie, A., Heydarnoori, A., Jamshidi, P.: Microservices architecture enables devops: migration to a cloud-native architecture. IEEE Softw. 33(3), 42–52, 2016.

[Garriga 2017] Garriga, M.: Towards a microservices taxonomy. In: Microservices: Science and Engineering Workshop, Co-located with Software Engineering and Formal Methods (SEFM), Trento, Italy, 2017.

[Jamshidi 2018] Jamshidi, P., Pahl, C., Mendonc, N.C., Lewis, J., and Tilkov, S. Microservices: The Journey So Far and Challenges Ahead, IEEE Software, 35(3):24–35, 2018.

[Porter 1985] Porter, M.E. The Competitive Advantage: Creating and Sustaining Superior Performance. NY: Free Press, 1985.