Specifiche di Architettura Tecnica MyPay4

Versione 1.0

SOMMARIO

1 GLOSSARIO

Termine	Descrizione
Kubernates	Piattaforma di orchestrazione dei container Open Source che consente l'utilizzo di un framework elastico del server web per le applicazioni cloud
Docker	Progetto open-source che automatizza il deployment di applicazioni all'interno di contenitori software, fornendo un'astrazione aggiuntiva grazie alla virtualizzazione a livello di sistema operativo di Linux
POD	I pod sono gruppi di container che condividono le stesse risorse di calcolo e la stessa rete. Rappresentano anche l'unità di scalabilità di Kubernetes e, quindi, se un container in un pod riceve più traffico di quello che può gestire, Kubernetes replica il pod su altri nodi del cluster
Container	I container sono simili alle macchine virtuali, ma presentano un modello di isolamento più leggero, condividendo il sistema operativo (OS) tra le applicazioni
YAML	File di definizione per Kubenetes
Ingress	Ingress è un servizio Kubernetes che bilancia i carichi di lavoro del traffico di rete nel cluster inoltrando le richieste pubbliche o private alle applicazioni
Dockerfile	File contenente un insieme di istruzioni per la definizione di immagini Docker

2 ACRONIMI

Termine	Descrizione
K8s	Kubernetes

3 CONTESTO

Lo scopo del presente documento è di illustrare la composizione architetturale dell'applicazione MyPay4.

4 ARCHITETTURA

Si descrive l'architettura della soluzione distinguendola sui tre livelli: funzionale, tecnologico e fisico.

4.1 FUNZIONALE

Il sistema è dedicato alla gestione del mondo dei pagamenti da parte di un intermediario tecnologico verso il nodo centrale nazionale dei pagamenti per le Pubbliche Amministrazioni PagoPA.
Il sistema sarà gestito dall'intermediario tecnologico e sarà messo a disposizione delle pubbliche amministrazioni aderenti al sistema e che avranno a disponibili:

- funzionalità dedicate ai cittadini per pagare posizioni debitorie come da normativa tramite PagoPA, visualizzare le posizioni aperte e chiuse e relativi avvisi di pagamento e ricevute di pagamento.

- funzionalità dedicate agli operatori per la gestione, riconciliazione e regolarizzazione delle posizioni debitorie dei cittadini.

- Funzionalità dedicate agli operatori dell'ente e amministratori del sistema per operazioni di configurazione degli enti e dei dovuti e relativi pagamenti previsti.

4.2 TECNOLOGICO

L'architettura sulla quale si baserà l'applicativo potrà essere costruita a container, questo permetterà una scalabilità orizzontale delle varie componenti: FE, BE e batch per poter reggere il carico fortemente variabile nel tempo costituito dai pagamenti e mantenere prestazioni e livello di servizio costanti.

Di fatto effettuando il deploy dei container sul cluster Kubernetes sarà possibile aumentare il numero delle istanze di un singolo o più componente a seconda della necessità, in modo da gestire picchi di carico di lavoro senza dover creare in anticipo macchine (virtuali o fisiche).
Ad esempio, in un momento di forte pressione sulla componente di back-end per le scadenze di fine mese si potranno aumentare le istanze di questo layer, oppure nel caso in cui si stiano accodando molti flussi di import, si potrà aumentare il numero delle istanze del batch dedicato alla gestione e processamento di tale flusso.

Si avrà così la possibilità di avere un cluster dimensionato correttamente bilanciando il carico su un layer specifico a seconda della necessità.

L'applicazione MyPay è composta da:

- Un insieme di servizi implementati in linguaggio Java su framework spring-boot
- Moduli di front end sviluppati con "**Angular 10 + Angular Material UI + FlexLayoute**" serviti dall'applicazione spring-boot

L'applicazione Spring Boot dovrà leggere e scrivere in uno schema DB PostgreSql dedicato con versione minima di riferimento 11.

L' Applicazione ha le seguenti dipendenze verso i seguenti servizi applicativi offerti da MyPlace dell'eco-sistema MyPortal, in particolare:

- **MyProfile:** Per la gestione dei Profili
- **MyDictionary:** per la definizione di form

A livello di servizi infrastrutturali si evidenziano le seguenti dipendenze:
- **Elastic Search:** log e monitoraggio attivo delle transazioni **OpenDistro v1.11.0**
- **PostgreSQL:** il database utilizzato da parte dell'applicazione
- **Ceph:** dovrà essere utilizzato per la gestione dei file dall'applicazione
- **Redis:** Per la gestione della cache

Al fine di incrementare l'esercibilità del sistema è disponibile una gestione a code per l'attivazione dei processi batch e per il monitoraggio degli stessi.

A tale scopo si utilizza un broker di messaggi **Artemis v2.17.0** che prevede:
- l'implementazione del protocollo client AMQP (Active Message Queueing Protocol);
- l'installazione in modalità cluster per garantire HA;

Infine l'applicativo dovrà interfacciarsi inoltre ad un **Servizio Posta** per invio email a utenti internet (almeno una per pagamento) e ad operatori (per operazioni di servizio).

Riportiamo nella figura sottostante uno schema riassuntivo delle componenti dell'applicazione e dell'interazione con infrastruttura regionale.

4.3 FISICO

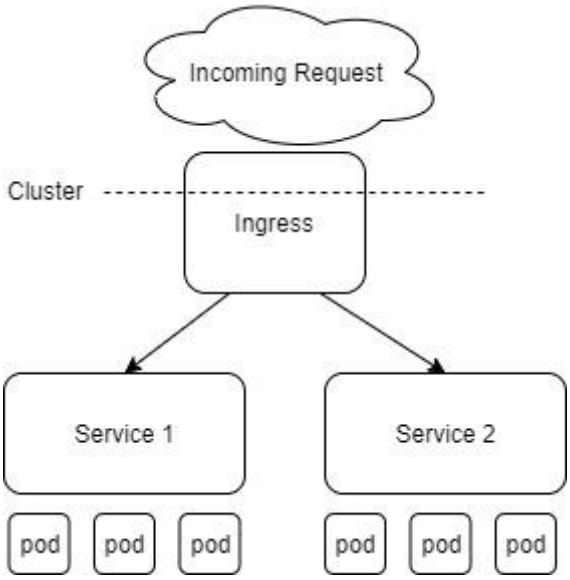
4.3.1 Distribuzione e deployment

L'architettura di Mypay sarà basata su container Docker, utilizzando come orchestratore Kubernetes al fine di implementare e garantire la scalabilità del sistema come da richiesta.

Aumentando il numero delle istanze di uno specifico pod (il container eseguito in kubernetes), manualmente o dinamicamente, a seconda del tipo di deploy effettuato, attraverso i file di configurazione specifici di fatto si attua la scalabilità del sistema come richiesto e precedentemente riportato.

Più precisamente Mypay sarà composto da una serie di container rilasciati su cluster Kubernetes strutturati in:
- container di front-end + container fluentd in modalità sidecar
- container di back-end + container fluentd in modalità sidecar
- container di processi batch + container fluentd in modalità sidecar

La raggiungibilità da rete esterna delle applicazioni passerà attraverso l'Ingress Nginx secondo il seguente schema esplicativo.



Come descritto sopra le tipologie di container saranno logicamente 3, una per la componente di front-end, una per il back-end ed una per i processi batch, mentre il database sarà un servizio erogato dall'infrastruttura.

Si prevede di utilizzare :

- Cluster Kubernetes v1.19.4
- Container runtime: Docker v19.03.13
- Ingress controller: Nginx

4.3.2 MyPay4_FrontEnd

Le componenti di MyPay_FrontEnd saranno applicazioni single page application sviluppate con il framework Angular v10 (insieme alle librerie Angular Material + Angular Flex Layout).
I moduli di FrontEnd saranno:

- **modulo MyPay-fe-PA-cittadino** su cui accederanno i cittadini;
- **modulo MyPay-fe-PA-operatore** su cui accederanno gli operatori per accedere alle funzionalità di back-office;

Il container di FrontEnd sarà basato sulla distribuzione Red Hat Universal Base Image 8 ubi8/openjdk-11.
Su questo container sarà installato il web server che servirà i contenuti statici di cui sono composti i moduli di FrontEnd.

4.3.3 MyPay4_BackEnd

Le componenti di MyPay4_BackEnd saranno applicazioni web sviluppate con il framwork Spring Boot v2.3.

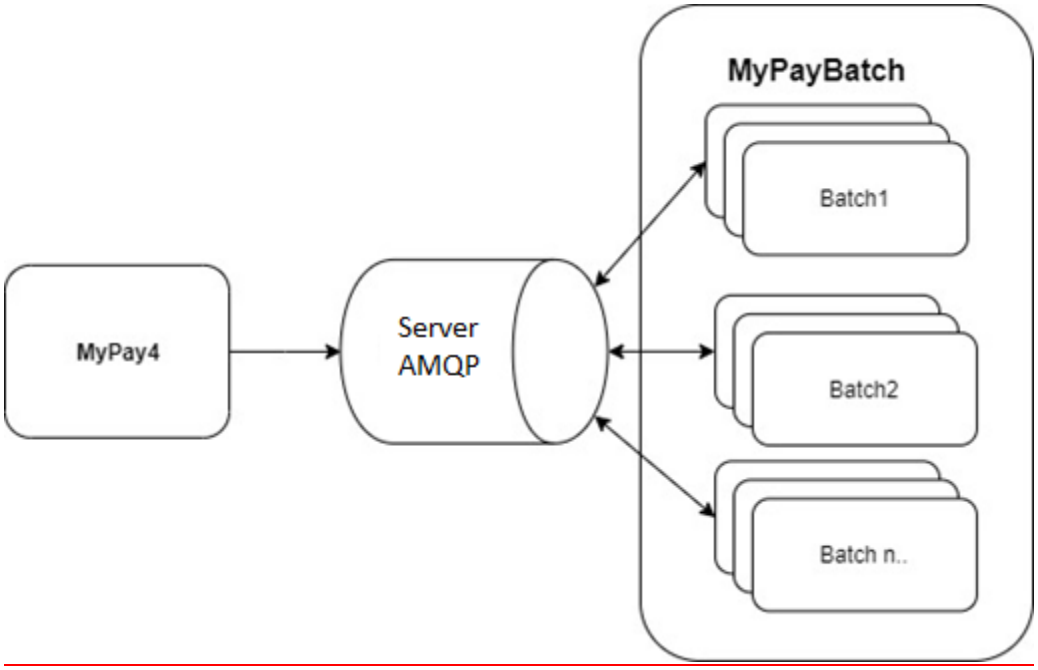
I moduli di BackEnd saranno:

- **modulo MyPay4-be-PA** che fornisce i servizi REST consumati dai moduli di Front-End Angular MyPay4-fe-PA-cittadino e MyPay4-fe-PA-operatore descritti nel precedente paragrafo;
- **modulo MyPay4-be-FESP** che funge da proxy per tutte le interazioni tra la suite MyPay4 e PagoPA;

Il container di BackEnd sarà basato sulla stessa distribuzione base usata per il FrontEnd.

4.3.4 MyPay4_Batch

Alcuni dei processi batch necessari e parte integrante e fondamentale di MyPay4 sono scritti in Talend e utilizzano il sistema a code previsto Artemis v2.17.0.



I processi batch potranno essere suddivisi in gruppi di POD od essere assegnati a POD specifici in modo da poterne rendere dinamica la scalabilità orizzontale, oltre alla facilita di gestione dei rilasci.

Il container sarà basato sulla stessa distribuzione usata per il front-end e back-end.

4.3.5 Volume per dati

Alcune funzionalità previste necessitano di un'area condivisa dove leggere e scrivere file:
Per MyPay4 rilasciato a container, si necessita di un volume condiviso tra i pod da mappare nei file di configurazione; soluzioni possibili possono essere:

1. Un volume NFS mappato opportunamente sui POD
2. Utilizzo di CEPH disponendo di un bucket in cui memorizzare i dati e montando tale bucket condiviso sui POD attraverso CEPHFS.

Tale volume condiviso dovrà essere soggetto a backup giornaliero.

4.3.6 Database

La componente di persistenza dei dati di MyPay4 utilizzerà come repository il database PostgreSQL la cui versione minima prevista è la 11.

Al fine di garantire i livelli di servizio previsti sarebbe preferibile avere istanza dedicata non a container anche alla luce delle seguenti considerazioni:

- Riavvii: poiché i POD sono transitori, la probabilità di eventi di failover è superiore a quella di un database tradizionalmente ospitato o completamente gestito.
- Perdita di dati: legato al punto precedente, le modalità di replica asincrona lasciano spazio alla perdita di dati, poiché il commit delle transazioni potrebbe essere eseguito al database primario ma non ai database secondari in caso di failover; nel caso specifico di postgresql si rischia di perdere la sincronizzazione del wal in caso di crash.
- Modalità di lavoro: i carichi di lavoro containerizzati devono intrinsecamente essere resilienti a riavvii, scalabilità orizzontale, virtualizzazione e altri vincoli. Quindi, la gestione della sincronizzazione dei blocchi dati che dovrà effettuare l'RDBMS è complicata da gestire in questo tipo di ambiente.
- Performances: dal momento che il database necessita di accedere ai dati presenti sul filesystem, con i container si introduce un layer ulteriore tra l'RDBMS e lo strato di persistenza dovendo passare necessariamente attraverso un volume condiviso.

Per il monitoraggio lato database si consiglia di poter disporre di PGWATCH o analogo.
Per il monitoraggio e gestione del cluster Kubernetes si consiglia di poter disporre di uno strumento tipo Rancher: https://rancher.com/.