ONION

SOFTWARE

softwareonion@gmail.com

Verbale esterno
2019-05-07

Informazioni sul documento

Responsabili	Federico Omodei
Redattori	Matteo Lotto
Verificatori	Nicola Zorzo
	Linpeng Zhang
Uso	Esterno
Lista distribuzione	*Onion Software*
	prof. Tullio Vardanega
	prof. Riccardo Cardin

L'attuale verbale riporta le decisioni prese dal gruppo Onion Software finalizzate allo sviluppo del progetto Butterfly in seguito al confronto col proponente.

Indice

1 Informazioni generali

Luogo	Videochiamata tramite Hangouts
Data	2019-05-07
Ora	19.00-19.35
Partecipanti	Alessio Lazzaron, Nicola Pastore, Matteo Lotto, Federico Brian
Esterni	Davide Zanetti
Redattore	Matteo Lotto

Per i verbali esterni la sezione §3 è scritta sotto forma di domanda-risposta.

Ogni domanda è rappresentata dalla lettera **D** seguita dal numero progressivo che la identifica; allo stesso modo la relativa risposta è rappresentata dalla lettera **R** seguita dal numero progressivo. Oltre alla domanda e alla risposta possono comparire delle *precisazioni*, indicate con la lettera **P** e seguite dal loro numero progressivo, che aiutano a identificare meglio il contesto o la spiegazione data.

Ogni risposta riportata tenta di rispecchiare in modo più verosimile possibile quanto riferito dal componente esterno che rappresenta il proponente *Imola Informatica S.P.A.*

2 Ordine del giorno

Gli argomenti trattati sono stati i seguenti:

- versione degli strumenti utilizzati dall'azienda;
- gestione di possibili problemi per i webhook di Redmine;
- confronto tra la nostra visione dell'architettura software e quella del proponente;
- valutazione di una possibile architettura *serverless*;
- disposizione dei server per il testing;
- scelta di MongoDB.

3 Resoconto

- versione degli strumenti utilizzati dall'azienda:

  D1 Che versioni dei vari strumenti usati dal programma preferite o utilizzate all'interno dell'azienda?

  R1 Per Kafka usate pure l'ultima versione. La versione di Redmine che usiamo è la 3.4.6; vi consiglio di utilizzare il container *passenger*, altrimenti Redmine è monothread.

- gestione di possibili problemi per i webhook di Redmine:

  D2 Abbiamo riscontrato delle difficoltà nell'interfacciarci con Redmine, dovute principalmente all'assenza di un metodo di gestione diretta dei webhook. Per poter risolvere il problema pensavamo di usare un plugin, che potrebbe però avere dei problemi nel riconoscere tutti i webhook mandati da Redmine che ci interessano. Avete qualche suggerimento a riguardo?

  R2 Esiste un plugin chiamato *Custom Workflow* che permette di definire per Redmine degli script in Ruby da fargli effettuare quando svolge delle funzioni. Con questo potete creare uno script che, quando viene aggiornata una issue, viene fatta un'attività. In questo caso viene mandata una chiamata *http* a un servizio, e va quindi in push al vostro programma. Se volete farlo bene questo è il modo.

  D3 Se riusciamo a trovare e risolvere il problema del plugin di Redmine andrebbe ugualmente bene? Abbiamo già iniziato a vedere Ruby ed eravamo consapevoli del fatto che se la versione di Redmine utilizzata dalla vostra azienda fosse stata la numero 4, allora il plugin sarebbe stato incompatibile.

  R3 Sì, se poi lo committate e lo mettete state facendo *open source* e io lo metterei nella documentazione quando mostrate il progetto.

- confronto tra la nostra visione dell'architettura software e quella del proponente:

  D4 Va bene il discorso di avere nel gestore personale un consumer e un producer?

  R4 Voi state usando un'architettura a microservizi, con un pattern *publisher-subscriber*, dove avete degli *adapter* che sono i vostri servizi che si interfacciano con i vostri strumenti, con una serie di microservizi che fanno gestire queste interazioni. L'unico servizio è il gestore personale: voi avete diviso questo microservizio in tre servizi: uno che si interfaccia al database, uno che fa la parte di *dispatcher-consumer*, e uno che fa la parte di *dispatcher-producer*. *Dispatcher-consumer* e *dispatcher-producer* li immagino molto piccoli, quindi se hanno il solo scopo di interfacciare Kafka con il gestore personale non ha senso che siano servizi esterni. Un *dispatcher-consumer* può avere senso come servizio esterno se riutilizzabile per altre cose; ma se è cablato appositamente per funzionare da interfaccia tra Kafka e il gestore personale, allora è una parte del gestore personale. Piuttosto potete dividere il gestore personale in due parti: una front-end, che faccia da interfaccia grafica e dietro chiama con *API Rest* il servizio del gestore personale.

- valutazione di una possibile architettura *serverless*:

  D5 Stavamo valutando di usare un'architettura serverless, dato che è la più usata nel mondo dell'informatica. Andrebbe bene?

  R5 Imparate a diversificare il trend dallo "stato dell'arte". È vero che il *serverless* viene sfruttato meglio nel cloud, portando a costi minori, ma si sta creando questo "trend" dove un'architettura monolite o SOA (*Service-Oriented Architecture*) viene fatta distribuita nel cloud. La scelta viene comunque lasciata a voi.

- disposizione dei server per il testing:

D6 Avreste dei server che potreste mettere a nostra disposizione per il programma?

R6 Come preferite: c'è l'infrastruttura su Docker o un cluster Rancher.

- scelta di MongoDB:

  D7 Come valuta la nostra scelta di usare MongoDB come database?

  R7 MongoDB è un database documentale, quando sarebbe stato meglio un database relazionale. Vi consiglio di ragionare sul tipo di dati che avete e sulla maniera più intelligente di versionarli e metterli nel database. Una volta che avete chiaro questo, scegliete che database usare.

4 Tracciamento delle decisioni

Codice	Decisione
VER-2019-05-07.1	Utilizzo di Redmine versione 3.4.6 e ultima versione di Kafka
VER-2019-05-07.2	Disposizione di un container Docker per il testing

Tabella 1: Tracciamento decisioni del verbale