



AlphaSix

AlphaSix

Manuale Utente

Informazioni sul documento

Nome Documento	ManualeUtente v2.0.0.pdf
Versione	2.0.0
Data di Creazione	24 marzo 2019
Data ultima modifica	14 maggio 2019
Stato	Approvato
Redazione	Timoty Granziero Ciprian Voinea Laura Cameran
Verifica	Samuele Gardin Matteo Marchiori
Approvazione	Timoty Granziero
Uso	Esterno
Distribuzione	AlphaSix
Destinato a	Prof. Tullio Vardanega, Prof. Riccardo Cardin, Imola Informatica
Email di riferimento	alpha.six.unipd@gmail.com

Descrizione

Questo documento è il manuale utente inerente al prodotto Butterfly del gruppo AlphaSix, utile agli utenti che vogliono interagire e usufruire del sistema.

Registro delle modifiche

Versione	Descrizione	Ruolo	Nominativo	Data
2.0.0	Approvazione per il rilascio	Responsabile	Timoty Granziero	2019-05-14
1.1.0	Verifica finale	Verificatore	Matteo Marchiori	2019-05-12
1.0.5	Ampliamento di alcuni contenuti in §3.1.2.4 e §3.1.1.3 perchè mancanti	Verificatore	Laura Cameran	2019-05-08
1.0.4	Inseriti esempi di file .json per i casi di errore in §3.2	Amministratore	Nicola Carlesso	2019-05-08
1.0.3	Inseriti esempi di file .json in §3.2	Verificatore	Matteo Marchiori	2019-05-07
1.0.2	Aggiornata §4 per la gestione dei messaggi persi	Verificatore	Nicola Carlesso	2019-05-05
1.0.1	Aggiornati §3.1.2.2 e §3.1.2.3	Verificatore	Samuele Gardin	2019-05-02
1.0.0	Approvazione per il rilascio	Responsabile	Nicola Carlesso	2019-04-11
0.3.0	Verifica finale	Verificatore	Matteo Marchiori	2019-04-09
0.2.6	Aggiornate parole in glossario §A	Programmatore	Timoty Granziero	2019-04-07
0.2.5	Aggiunto paragrafo §5	Programmatore	Laura Cameran	2019-04-03
0.2.4	Aggiunto paragrafo §4	Programmatore	Laura Cameran	2019-04-03
0.2.3	Modificato §3.2, che elenca come accedere alle risorse utenti e preferenze	Programmatore	Matteo Marchiori	2019-04-01
0.2.2	Aggiunto §3.2, che elenca come accedere alle risorse utenti e progetti	Programmatore	Timoty Granziero	2019-04-01
0.2.1	Aggiunto paragrafo §3	Programmatore	Ciprian Voinea	2019-03-31
0.2.0	Verifica	Verificatore	Samuele Gardin	2019-03-29
0.1.2	Aggiunto paragrafo §2	Programmatore	Ciprian Voinea	2019-03-28
0.1.1	Aggiunto scheletro per §2 e §3 per arricchire il contenuto	Programmatore	Timoty Granziero	2019-03-26
0.1.0	Verifica	Verificatore	Samuele Gardin	2019-03-25
0.0.2	Aggiunta §1 per l'introduzione al documento	Programmatore	Timoty Granziero	2019-03-24
0.0.1	Creazione template documento	Programmatore	Timoty Granziero	2019-03-24



Indice

1	Introduzione	1
1.1	Glossario	1
1.2	Scopo del documento	1
1.3	Scopo del prodotto	1
2	Configurazione	2
2.1	Requisiti di sistema	2
2.1.1	Software	2
2.1.2	Hardware	2
2.2	Mappatura delle porte	3
2.3	Suddivisione dei container ed immagini utilizzate	3
2.4	Configurazione webhook	4
2.4.1	Redmine	5
2.4.1.1	Configurazione del plugin	5
2.4.1.2	Configurazione destinazione	5
2.4.1.3	Eventi supportati	5
2.4.2	GitLab	5
2.4.2.1	Configurazione webhook nella rete locale	5
2.4.2.2	Configurazione destinazione	6
2.4.2.3	Eventi supportati	6
2.5	Configurazione servizi Butterfly	6
2.5.1	Producer	6
2.5.2	Consumer	7
2.5.3	Gestore Personale	7
2.5.4	Kafka	7
3	Utilizzo di Butterfly	8
3.1	Gestore Personale	8
3.1.1	Interfaccia utente	8
3.1.1.1	Accesso	8
3.1.1.2	Pannello di controllo	9
3.1.1.3	Modifica preferenze	9
3.1.1.4	Modifica dei propri dati	11
3.1.1.5	Uscita dal sistema	11
3.1.2	Interfaccia amministratore	11
3.1.2.1	Pannello di controllo	11
3.1.2.2	Iscrizione a Butterfly	13
3.1.2.3	Rimozione di un utente	13
3.1.2.4	Visualizzazione utenti	14
3.1.2.5	Rimozione di un progetto	14
3.1.2.6	Visualizzazione dei progetti	14
3.2	API Rest	15
3.2.1	User	15
3.2.1.1	Visualizzazione	15
3.2.1.1.1	Esempio di input	15
3.2.1.1.2	Esempio di output	15
3.2.1.2	Inserimento	16
3.2.1.2.1	Esempio di input	17
3.2.1.2.2	Esempio di output	17

3.2.1.3	Modifica	17
3.2.1.3.1	Esempio di input	17
3.2.1.3.2	Esempio di output	18
3.2.1.4	Rimozione	18
3.2.1.4.1	Esempio di input	18
3.2.1.4.2	Esempio di output	18
3.2.1.5	Riepilogo	19
3.2.2	Project	19
3.2.2.1	Visualizzazione	19
3.2.2.1.1	Esempio di input	19
3.2.2.1.2	Esempio di output	19
3.2.2.2	Rimozione	20
3.2.2.2.1	Esempio di input	20
3.2.2.2.2	Esempio output	20
3.2.2.3	Riepilogo	20
3.2.3	Preference	20
3.2.3.1	Inserimento	20
3.2.3.1.1	Esempio di input	21
3.2.3.1.2	Esempio di output	21
3.2.3.2	Modifica	21
3.2.3.2.1	Esempio di input	22
3.2.3.2.2	Esempio di output	22
3.2.3.3	Rimozione	23
3.2.3.3.1	Esempio di input	23
3.2.3.3.2	Esempio output	23
3.2.3.4	Riepilogo	23
3.3	Piattaforma di messaggistica	24
3.3.1	Email	24
3.3.2	Telegram	24
4	Note sulla gestione delle notifiche	26
4.1	Caso di persona non disponibile	26
5	Segnalazione problematiche	27
A	Glossario	28
B	28
Broker	28
C	28
Cluster	28
Consumer	28
Container	28
D	28
Docker	28
Docker Compose	28
I	28
ID	28
J	29
JSON	29
K	29
Kubernetes	29
M	29



	Metadato	29
N		29
	Namespace	29
P		29
	Payload	29
	Plug and play	30
	Plugin	30
	Pod	30
	Prodotto	30
	Producer	30
	Push	30
R		30
	Rancher	30
	Root path	30
T		30
	Topic	30
U		30
	Update	30
W		31
	Webhook	31



Elenco delle tabelle

1	Suddivisione del range delle porte a disposizione su Rancher	3
2	Configurazione delle porte in fase di sviluppo e consegna	3
3	Riepilogo delle Rest API per la risorsa User	19
4	Riepilogo delle Rest API per la risorsa Project	20
5	Riepilogo delle Rest API per la risorsa Preference	23

Elenco delle figure

1	Lista plugin configurati	5
2	Menu plugin <code>redmine_webhook</code>	5
3	Configurazione webhook con destinazione in rete locale	6
4	Form di accesso al sistema	8
5	Messaggio di errore durante accesso al sistema	9
6	Pannello di controllo per utente generico	9
7	Interfaccia modifica preferenze	10
8	Modifica dei dati di un utente	11
9	Pannello di controllo per utente con privilegi di amministratore	12
10	Interfaccia inserimento nuovo utente	13
11	Interfaccia rimozione di un utente	13
12	Interfaccia visualizzazione degli utenti	14
13	Interfaccia rimozione di un progetto	14
14	Interfaccia visualizzazione dei progetti	15
15	Formato dell' Email ricevuta da un utente finale	24
16	Formato del messaggio Telegram ricevuto da un utente finale	25



1 Introduzione

1.1 Glossario

Al fine di rendere il documento più chiaro possibile, i termini che possono assumere un significato ambiguo o che richiedono una spiegazione avranno una G a pedice (e.g. `WEBHOOKG`), e saranno riportati nell'appendice §A.

1.2 Scopo del documento

Il documento corrente ha lo scopo di illustrare la procedura per configurare correttamente il software Butterfly, indirizzato all'amministratore che avrà il compito di inizializzarlo e agli utenti nel caso di registrazione e modifica delle proprie informazioni.

1.3 Scopo del prodotto

Lo scopo del `PRODOTTOG` è creare un `APPLICATIVOG` per poter gestire i messaggi o le segnalazioni provenienti da diversi prodotti per la realizzazione di software. Queste segnalazioni passano attraverso un `BROKERG` che gestisce i canali a loro dedicate per poi distribuirle ad applicazioni di messaggistica.

Il software dovrà inoltre essere in grado di riconoscere il `TOPICG` dei messaggi in input per poterli inviare a determinati canali a cui i destinatari dovranno iscriversi.

È anche richiesto di creare un canale specifico per gestire le particolari esigenze dell'azienda. Questo dovrà essere in grado, attraverso la lettura di particolari `METADATIG`, di reindirizzare i messaggi ricevuti al destinatario più appropriato.

2 Configurazione

Questa sezione è dedicata alla persona delegata alla configurazione di Butterfly. Su richiesta di Imola Informatica, tutti i servizi sono stati configurati sotto forma di `CONTAINERG`. È quindi possibile avviarli su macchine fisiche differenti, ma collegate in rete fra loro, con `DOCKERG`. Sempre sotto richiesta di Imola Informatica viene utilizzato `RANCHERG` come software per la gestione dei container, quindi questa guida verterà principalmente sulla configurazione di Butterfly utilizzando questa tecnologia.

2.1 Requisiti di sistema

Non sono necessari particolari requisiti in modo da poter configurare e utilizzare il nostro prodotto, tuttavia valgono i requisiti minimi dei sistemi di terze parti che vengono utilizzati da Butterfly.

2.1.1 Software

- **Docker**¹: è necessario avere installata e configurata correttamente almeno la versione v18.09.
- **Docker Compose**²: nel caso si decidesse di utilizzare `DOCKER COMPOSEG` per l'avvio dei container è consigliata l'installazione e configurazione corretta della versione v3.7.
- **Kubernetes**³: nel caso si decidesse di utilizzare `KUBERNETESG` per la gestione dei container è consigliata l'installazione e configurazione corretta della versione v1.13.
- **Rancher**⁴: nel caso si decidesse di utilizzare Rancher per la gestione grafica di oggetti Kubernetes contenenti i container Docker, è consigliata l'installazione e configurazione della versione v2.1.4.
- **Kafka**⁵: è necessario avere installata e configurata correttamente almeno la versione v2.12.
- **GitLab**⁶: durante lo sviluppo di Butterfly abbiamo fatto riferimento alla versione v11.7.
- **Redmine**⁷: durante lo sviluppo di Butterfly abbiamo fatto riferimento alla versione v3.3.9.

GitLab e Redmine sono componenti esterne a Butterfly, tuttavia vengono citate in quanto è possibile configurare tutto il progetto in un ambiente unico.

Le versioni specificate possono essere trovate anche tra i requisiti di vincolo nel documento *AnalisiDeiRequisiti v3.0.0_D*.

2.1.2 Hardware

Per Butterfly non sono necessari ulteriori requisiti a livello hardware particolari se non quelli di cui hanno bisogno i software precedentemente elencati. Unicamente per Butterfly, senza tenere conto di servizi di terze parti, i requisiti sono:

- OS: Ubuntu 16.04 (o successive versioni)

¹<https://docs.docker.com/v17.09/datacenter/ucp/2.1/guides/admin/install/system-requirements>

²<https://docs.docker.com/compose/install/>

³<https://kubernetes.io/docs/setup/independent/install-kubeadm>

⁴<https://rancher.com/docs/rancher/v2.x/en/installation/requirements/>

⁵<https://docs.confluent.io/current/installation/system-requirements.html>

⁶<https://git.ucd.ie/help/install/requirements.md>

⁷<https://www.easyredmine.com/faq/technical-info/176-hardware-and-software-requirements-for-server-solution>

- RAM: 3GB
- CPU: IntelCore i3 con 2.1 GHz (o processori con potenza analoga o superiore)
- Spazio su disco: 2GB
- Completa possibilità di comunicazione in rete dei container

2.2 Mappatura delle porte

Per la configurazione di ciascun tipo servizio abbiamo deciso di dare un range di porte da poter esporre in modo tale da effettuare una separazione logica.

Questa regola non influisce col corretto funzionamento di Butterfly, ma è solamente per non assegnare le porte in modalità casuale e facilitare le analisi di eventuali errori, la manutenzione e l'aggiunta futura di nuovi servizi. La suddivisione delle porte è la seguente:

Servizio	Porta inizio	Porta fine
Software di terze parti	30000	30029
Kafka e servizi correlati	30030	30059
Producer	30060	30089
Consumer	30090	30119
Gestore Personale	30120	30149

Tabella 1: Suddivisione del range delle porte a disposizione su Rancher

La suddivisione che abbiamo utilizzato, durante lo sviluppo e la consegna del progetto, prevede la seguente esposizione delle porte:

Servizio	Porta interna	Porta esposta
Redmine	3000	30000
GitLab	80	30001
Kafka	9092	30030
Producer GitLab	5000	30060
Producer Redmine	5000	30061
Gestore Personale Client	5000	30120

Tabella 2: Configurazione delle porte in fase di sviluppo e consegna

Le specifiche relative alla configurazione delle porte in Rancher possono essere trovate nella documentazione presente sul sito di Rancher nella pagina dedicata⁸.

2.3 Suddivisione dei container ed immagini utilizzate

Come detto in precedenza, su consiglio di Imola Informatica, Butterfly è stato sviluppato con l'utilizzo di Rancher per la gestione dei container.

I container sono stati suddivisi in modo tale da utilizzare un unico $CLUSTER_G$ (nel nostro caso chiamato Butterfly), vari $NAMESPACE_G$ per separare i servizi fra loro e i POD_G , che rappresentano ciascuno un container. Le immagini utilizzate sono tutte presenti in DockerHub, incluse quelle

⁸<https://rancher.com/docs/rancher/v2.x/en/installation/references/>

dei `PRODUCERG` e dei `CONSUMERG` che sono state create utilizzando build automatiche innescate ad ogni `PUSHG` nella repository del codice.

La configurazione che abbiamo utilizzato all'interno del Cluster Butterfly è:

- Namespace: `gitlab`
 - Pod: `gitlab` (immagine: `gitlab/gitlab-ce:latest`⁹)
- Namespace: `redmine`
 - Pod: `redmine` (immagine: `redmine:3.3.9`¹⁰)
 - Pod: `postgres-redmine` (immagine: `postgres:11-alpine`¹¹)
- Namespace: `producer`
 - Pod: `producer-gitlab` (immagine: `alphasix/producer-gitlab:master`¹²)
 - Pod: `producer-redmine` (immagine: `alphasix/producer-redmine:master`¹³)
- Namespace: `kafka`
 - Pod: `kafka-kafka` (immagine: `confluentinc/cp-kafka:4.0.1-1`¹⁴)
 - Pod: `kafka-zookeeper` (immagine: `confluentinc/cp-zookeeper:4.1.1`¹⁵)
- Namespace: `gestore-personale`
 - Pod: `gestore-personale` (immagine: `alphasix/gestore-personale:master`¹⁶)
 - Pod: `gestore-personale-client`
(immagine: `alphasix/gestore-personale-client:master`¹⁷)
 - Pod: `mongo` (immagine: `mongo/mongo:4.0.9`¹⁸)
- Namespace: `consumer`
 - Pod: `consumer-telegram` (immagine: `alphasix/consumer-telegram:master`¹⁹)
 - Pod: `consumer-email` (immagine: `alphasix/consumer-email:master`²⁰)

È importante quando si utilizza un'istanza non in locale di MongoDB modificare il file `etc/mongod.conf.orig` in modo tale che si possano effettuare richieste in remoto, impostando quindi la variabile `bindIp` sia uguale a `0.0.0.0`.

2.4 Configurazione webhook

Per il corretto invio dei `WEBHOOKG` (in formato `JSONG`) da parte dei software che comunicano con i nostri Producer, è necessaria una breve configurazione delle applicazioni che dialogano con i Producer (GitLab e Redmine) spiegata nel dettaglio nei prossimi paragrafi.

⁹<https://hub.docker.com/r/gitlab/gitlab-ce/>

¹⁰https://hub.docker.com/_/redmine

¹¹https://hub.docker.com/_/postgres

¹²<https://hub.docker.com/r/alphasix/producer-gitlab>

¹³<https://hub.docker.com/r/alphasix/producer-redmine>

¹⁴<https://hub.docker.com/r/confluentinc/cp-kafka/>

¹⁵<https://hub.docker.com/r/confluentinc/cp-zookeeper/>

¹⁶<https://hub.docker.com/r/alphasix/gestore-personale>

¹⁷<https://hub.docker.com/r/alphasix/gestore-personale-client>

¹⁸https://hub.docker.com/_/mongo

¹⁹<https://hub.docker.com/r/alphasix/consumer-telegram>

²⁰<https://hub.docker.com/r/alphasix/consumer-email>

2.4.1 Redmine

2.4.1.1 Configurazione del plugin

Per poter inviare un webhook con Redmine è necessario installare un `PLUGING` esterno chiamato `redmine_webhook`²¹. Per installarlo è necessario scaricarlo (tramite il comando `git clone` ad esempio) nella cartella `/plugins/redmine`. Non sono necessarie ulteriori configurazioni in quanto questo software è di tipo `PLUG AND PLAYG`. Per verificare la corretta integrazione con Redmine, controllare da un profilo con privilegi amministratore che nella sezione **Settings** > **Plugins** sia presente il plugin appena scaricato.

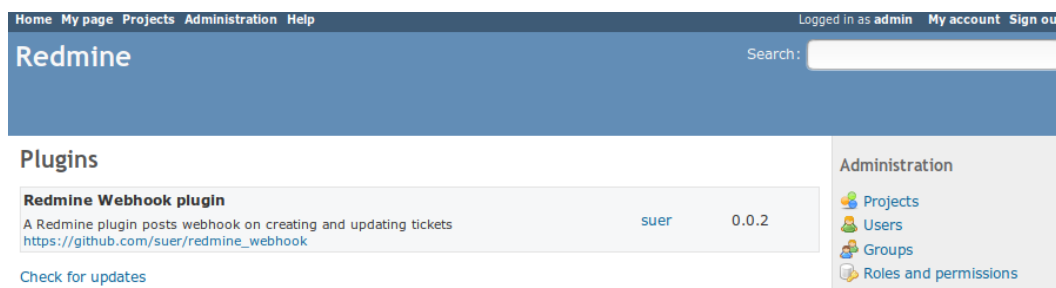


Figura 1: Lista plugin configurati

2.4.1.2 Configurazione destinazione

Per aggiungere una destinazione del webhook per un progetto, andare nella sezione relativa a quest'ultimo all'interno del menu del progetto selezionato (**Settings** > **Webhook**). Nell'area di input aggiungere l'indirizzo di destinazione e premere "Add". Nel caso si volesse rimuovere un indirizzo precedentemente inserito allora premere "Remove".

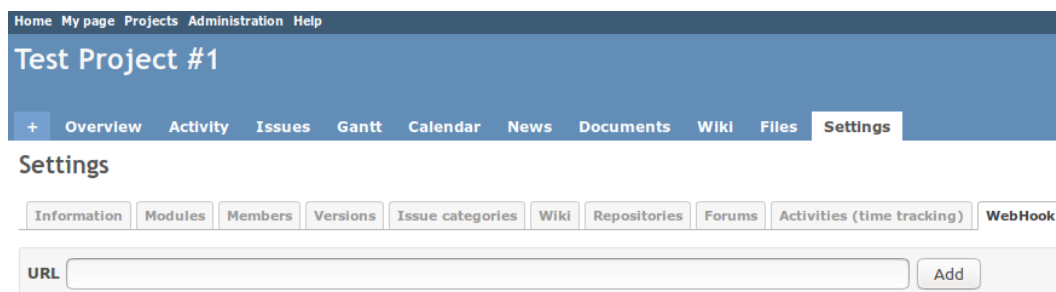


Figura 2: Menu plugin `redmine_webhook`

2.4.1.3 Eventi supportati

Per Redmine gli eventi supportati sono:

- Issue²²

2.4.2 GitLab

2.4.2.1 Configurazione webhook nella rete locale

È necessario abilitare l'invio dei webhook a dispositivi presenti nella stessa rete dalla parte amministrativa. Questo può essere effettuato accedendo con un account con privilegi amministratore all'indirizzo:

²¹https://github.com/suer/redmine_webhook.git

²²Il plugin `redmine_webhook` offre l'invio dei webhook solamente da parte degli eventi relativi alle Issue

/admin/application_settings/network

Per abilitare questa funzionalità, è necessario cliccare sul riquadro visibile nell'immagine 3 e che si trova all'indirizzo sopra indicato.

Outbound requests

Allow requests to the local network from hooks and services.

☐ Allow requests to the local network from hooks and services

Figura 3: Configurazione webhook con destinazione in rete locale

Ulteriori informazioni a riguardo possono essere trovate nella pagina²³ relativa a tale argomento nella sezione relativa alla documentazione di GitLab.

2.4.2.2 Configurazione destinazione

Successivamente si può aggiungere un indirizzo di destinazione di un webhook al progetto in **Settings > Integrations**. Da qui si possono selezionare gli eventi di interesse per i quali i webhook verranno attivati.

Nel campo relativo all'URL di destinazione inserire l'indirizzo con la relativa porta (nel nostro caso quella specificata nella Tabella 2) su cui il Producer GitLab è in ascolto.

2.4.2.3 Eventi supportati

Per GitLab gli eventi supportati sono:

- Push events
- Comments
- Issues events

2.5 Configurazione servizi Butterfly

2.5.1 Producer

Per i Producer è possibile impostare nell'apposito file di configurazione le variabili relative. Per il Producer di Redmine i campi sono:

- "kafka":{"bootstrap_servers"}: indirizzo e porta dell'istanza di Kafka disponibile.
- "redmine":{"ip"}: indirizzo dell'istanza di Redmine disponibile.
- "redmine":{"port"}: porta dell'istanza di Redmine disponibile.

Per il Producer di GitLab i campi sono:

- "kafka":{"bootstrap_servers"}: indirizzo e porta dell'istanza di Kafka disponibile.
- "gitlab":{"ip"}: indirizzo dell'istanza di GitLab disponibile.
- "gitlab":{"port"}: porta dell'istanza di GitLab disponibile.

²³<https://docs.gitlab.com/ee/security/webhooks.html>

Sono inoltre disponibili variabili di ambiente configurate tramite Dockerfile oppure definite dall'interno di Rancher.

- KAFKA_IP : indirizzo dell'istanza di Kafka disponibile.
- KAFKA_PORT : porta dell'istanza di Kafka disponibile.

2.5.2 Consumer

Per i Consumer, come per i Producer, è possibile impostare le variabili tramite un file di configurazione. Queste sono, per il Consumer di Telegram:

- "kafka":{"bootstrap_servers"}: indirizzo e porta dell'istanza di Kafka disponibile.
- "telegram":{"token_bot"}: ID_G del Bot di Telegram che inoltra la notifica.

Per il Consumer Email, invece:

- "kafka":{"bootstrap_servers"}: indirizzo e porta dell'istanza di Kafka disponibile.
- "email":{"sender"}: indirizzo email che inoltra l'email.
- "email":{"psw"}: password della casella email.

È inoltre possibile impostare altre configurazioni tramite le seguenti variabili di ambiente:

- KAFKA_IP : indirizzo dell'istanza di Kafka disponibile.
- KAFKA_PORT : porta dell'istanza di Kafka disponibile.
- BUTTERFLY_CONSUMER_EMAIL_PSW: password della casella email.

2.5.3 Gestore Personale

Per il Gestore Personale, come per i Producer e per i Consumer, è possibile impostare le variabili di file di configurazione.

Per la componente Producer del Gestore Personale queste sono:

- "kafka":{"bootstrap_servers"}: indirizzo e porta dell'istanza di Kafka disponibile.

Per la componente Consumer del Gestore Personale queste sono:

- "kafka":{"bootstrap_servers"}: indirizzo e porta dell'istanza di Kafka disponibile.

È inoltre possibile impostare altre configurazioni tramite le seguenti variabili di ambiente:

- KAFKA_IP : indirizzo dell'istanza di Kafka disponibile.
- KAFKA_PORT : porta dell'istanza di Kafka disponibile.
- MONGO_IP: password della casella email.

2.5.4 Kafka

Per l'utilizzo richiestoci da parte dell'azienda di Kafka non sono state necessarie ulteriori configurazioni oltre a quelle di default successive all'installazione.

3 Utilizzo di Butterfly

3.1 Gestore Personale

Il Gestore Personale è la componente principale di Butterfly e si può suddividere in due sotto-componenti principali:

- Interfaccia utente a sua volta divisa in
 - Interfaccia per amministratore
 - Interfaccia per utente normale
- Message processor, che contiene la logica di business di Butterfly

Il secondo punto non è di pertinenza di questo manuale (ma del *ManualeSviluppatore v2.0.0_D*), in quanto l'utente non lo usa direttamente, per cui verrà solamente discusso l'utilizzo del sistema tramite l'interfaccia utente.

L'inserimento della figura dell'amministratore, non presente nel *Manuale Sviluppatore V1.0.0_D*, è giustificata in *VE_2019-04-26_D*.

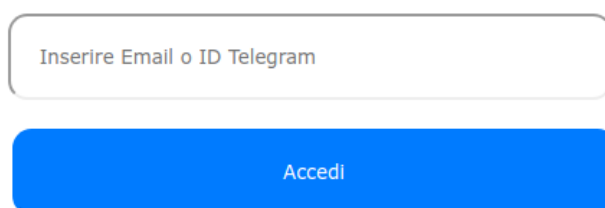
3.1.1 Interfaccia utente

Le seguenti opzioni del Gestore Personale sono accessibili da ogni tipo utente, perciò anche da quelli coi privilegi di amministratore.

3.1.1.1 Accesso

È possibile effettuare l'accesso all'interno del sistema tramite il link del container sul quale è in esecuzione, specificato nella configurazione effettuata precedentemente. Per effettuare correttamente l'accesso è necessario l'inserimento del proprio ID Telegram o Email con il quale si è iscritti nel sistema.

Accesso Butterfly

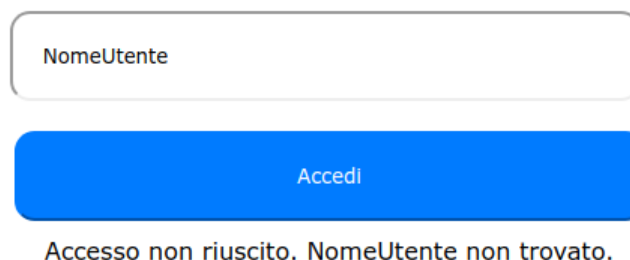


Il form di accesso al sistema Butterfly è composto da due elementi principali: un campo di input rettangolare con bordi arrotondati e un pulsante di azione rettangolare con bordi arrotondati. Il campo di input ha un placeholder testuale "Inserire Email o ID Telegram" in grigio. Il pulsante è di colore blu e contiene il testo "Accedi" in bianco.

Figura 4: Form di accesso al sistema

In caso l'identificativo inserito non fosse valido, e non corrispondesse quindi a un match nel database, verrà mostrato all'utente un messaggio di errore.

Accesso Butterfly



NomeUtente

Accedi

Accesso non riuscito. NomeUtente non trovato.

Figura 5: Messaggio di errore durante accesso al sistema

Nel caso in cui un utente cerchi di accedere ad una sezione, ma senza aver effettuato l'accesso, questo verrà rimandato alla pagina di accesso.


Se un utente iscritto correttamente nel sistema ha registrato precedentemente sia indirizzo Email che ID Telegram, allora può accedere utilizzando indipendentemente il primo o il secondo. In cima alla pagina, per identificare l'utente che ha acceduto, viene mostrato l'ID Telegram o la mail con cui si ha acceduto.

3.1.1.2 Pannello di controllo

Dopo aver effettuato l'accesso al sistema, si viene rimandati alla pagina relativa al pannello di controllo che contiene i comandi principali per la navigazione del sito e le operazioni che un utente può eseguire. Le sezioni a cui si può navigare dal pannello di controllo sono:

- Modifica dei propri dati
- Modifica delle proprie preferenze
- Uscita dal sistema (logout)

Pannello di controllo



Modifica i tuoi dati

Modifica le tue preferenze

Logout

Figura 6: Pannello di controllo per utente generico

3.1.1.3 Modifica preferenze

La modifica delle preferenze è possibile solamente per utenti già iscritti ed autenticati nel sistema. È raggiungibile tramite il pannello di controllo sotto la voce “Modifica le tue preferenze”. In questa

sezione si possono trovare le principali impostazioni del sistema dal punto di vista dell'utente finale:

- Lista dei progetti a cui si è iscritti
- Modifica della priorità assegnata ad un progetto che può essere:
 - **Alta:** 1
 - **Media:** 2
 - **Bassa:** 3
- Lista dei Topic (formati dall'insieme di label e keyword) disponibili e iscrizione o disiscrizione da questi, che vengono mostrati dinamicamente in base ai progetti a cui si è iscritti. L'inserimento per keyword d'interesse deve essere fatta in modo testuale e la loro divisione viene riconosciuta attraverso una “,”
- Modifica dei progetti d'interesse (aggiunta o rimozione dei progetti dalle proprie preferenze)
- Inserimento dei giorni di indisponibilità da calendario
- Impostazione delle due piattaforme di messaggistica (Email o Telegram) su cui ricevere le notifiche. È possibile scegliere solo una delle due piattaforme di messaggistica

Le “label” non sono modificabili in quanto vengono aggiornate solamente da una componente del Gestore Personale che le memorizza quando avvengono UPDATE_G alle issue relative ai progetti. Le “keyword” invece, sono formate da una lista di parole che possono essere contenute nei messaggi di push di GitLab e di cui si è interessati a ricevere notifiche.

User id: alpha.six.unipd@gmail.com

Modifica preferenze utente

Modifica preferenze dei topics

URL	Nome progetto	Applicazione	Priorità	Labels	Keywords
http://192.168.150.216:30001/root/butterfly-test-project-1	Butterfly Test Project 1	gitlab	3	bug fix report	

Modifica preferenze di progetti e topic

Progetto aggiunto correttamente.

Aggiungi e rimuovi progetti

Butterfly Test Project 1 - gitlab

Aggiungi il progetto

Rimuovi il progetto

Giorni di indisponibilità

May 2019

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28

29 30 31

Mese precedente

Mese successivo

Modifica irreperibilità

Piattaforma preferita

Email ☒ Telegram ☐

Modifica piattaforma preferita

Torna al pannello

Figura 7: Interfaccia modifica preferenze

3.1.1.4 Modifica dei propri dati

La modifica dei propri dati è possibile attraverso la sezione “Modifica i tuoi dati” accessibile dal pannello di controllo. Qui un utente può modificare i dati inseriti dall’amministratore in fase di registrazione.

Modifica dati utente

Nome:	<input type="text" value="Simone"/>
Cognome:	<input type="text" value="Granziero"/>
Email:	<input type="text" value="simone.granziero@email.it"/>
Telegram:	<input type="text" value="123432"/>

Figura 8: Modifica dei dati di un utente

Nel caso questi fossero già associati ad una persona iscritta a Butterfly, verrà visualizzato un messaggio di errore.

Per quanto riguarda l’inserimento dell’ID Telegram, questo non deve essere nella forma @nome_utente, ma bensì numerico. Per poterlo ottenere consigliamo l’utilizzo del bot Telegram “MyIDBot²⁴” dandogli il comando /getid.

3.1.1.5 Uscita dal sistema

Per effettuare il logout dal sistema, cliccare sul bottone “Logout” presente nel pannello di controllo.

3.1.2 Interfaccia amministratore

All’interno dell’applicazione Butterfly, gli utenti con i permessi di amministratore potranno eseguire le stesse azioni di un utente normale e in più gestire l’insieme degli utenti iscritti al sistema. Il sistema di accesso per l’utente amministratore non differisce da quello per gli altri utenti.

3.1.2.1 Pannello di controllo

Dopo aver effettuato l’accesso al sistema, si viene rimandati alla pagina relativa al pannello di controllo che contiene i comandi principali per la navigazione del sito e le operazioni che un utente può eseguire. Le sezioni a cui si può navigare dal pannello di controllo sono:

- Inserimento di un nuovo utente
- Rimozione di un utente
- Visualizzazione degli utenti iscritti

²⁴<https://telegram.me/storebot?start=myidbot>

- Rimozione di un progetto
- Visualizzazione dei progetti
- Modifica dei propri dati
- Modifica delle proprie preferenze
- Uscita dal sistema (logout)

Pannello di controllo



Figura 9: Pannello di controllo per utente con privilegi di amministratore

3.1.2.2 Iscrizione a Butterfly

L'iscrizione di un nuovo utente al sistema Butterfly è permessa solamente all'utente amministratore. La sezione di inserimento di un nuovo utente è raggiungibile dal pannello di controllo e prevede l'inserimento dei seguenti dati:

- Nome
- Cognome
- Email
- Telegram

Inserimento nuovo utente



The form consists of four stacked input fields, each with a label to its left: 'Nome:', 'Cognome:', 'Email:', and 'Telegram:'. Each input field contains the placeholder text 'Inserire nome', 'Inserire cognome', 'Inserire email', and 'Inserire Telegram ID' respectively. Below the input fields are two blue buttons: 'Inserisci nuovo utente' on the left and 'Torna al pannello' on the right.

Figura 10: Interfaccia inserimento nuovo utente

È necessario inserire almeno un campo a scelta tra Email e Telegram. Nel caso questi fossero già associati ad un utente iscritto a Butterfly verrà visualizzato un messaggio di errore.

3.1.2.3 Rimozione di un utente

Viene data la possibilità di rimuovere un utente dal sistema solamente all'amministratore. La rimozione è possibile attraverso la sezione "Rimuovi un utente" accessibile dal pannello di controllo.

Rimozione utente



The form features a single input field labeled 'Email o ID Telegram:' containing the text '12312'. Below this field are two blue buttons: 'Rimuovi utente' on the left and 'Torna al pannello' on the right.

Figura 11: Interfaccia rimozione di un utente

3.1.2.4 Visualizzazione utenti

Un utente amministratore ha la possibilità di vedere la lista di tutti gli utenti iscritti al sistema e i relativi progetti a cui sono interessati. Per ogni progetto viene inoltre riportata la priorità che ha per l'utente, le labels e la keywords d'interesse. La visualizzazione avviene attraverso la sezione "Visualizza gli utenti" accessibile dal pannello di controllo.

Visualizzazione dettagli utente

Email o ID Telegram:

Url	Priorità	Labels	Keywords
http://localhost/redmine/project-1	1	bug,CI	kw1,kw2,kw3
http://localhost/gitlab/gitlab-2	3	java,coding	kw1,kw2,kw3

Figura 12: Interfaccia visualizzazione degli utenti

3.1.2.5 Rimozione di un progetto

I progetti provenienti da GitLab o Redmine vengono aggiunti in automatico al sistema e la loro gestione (intesa come creazione e modifica) non è competenza degli utenti. Perciò i progetti possono solo essere aggiunti o rimossi dalle preferenze di un utente, oppure rimossi completamente dal sistema nel momento in cui questi, ad esempio, vengono chiusi nelle relative applicazioni di provenienza.

Per rimuovere un progetto dal sistema, l'interfaccia dell'amministratore possiede un'apposita pagina chiamata "Rimuovi un progetto" dove è possibile selezionare i progetti da rimuovere.

Rimozione progetto

Url:

Figura 13: Interfaccia rimozione di un progetto

3.1.2.6 Visualizzazione dei progetti

Un utente amministratore ha la possibilità di vedere la lista di tutti i progetti presenti nel sistema. Di ognuno viene mostrata la url identificativa, il nome, l'applicazione di provenienza e i topics rilevati relativi al progetto. La visualizzazione avviene attraverso la sezione "Visualizza i progetti" accessibile dal pannello di controllo.

Visualizzazione dettagli progetto

Url:

[Mostra dettagli progetto](#) [Torna al pannello](#)

Url	Name	App	Topics
http://localhost/redmine/project-2	Project-2	redmine	golang,bug,wontfix

Figura 14: Interfaccia visualizzazione dei progetti

3.2 API Rest

Per la gestione delle risorse di Butterfly abbiamo utilizzato lo standard architetturale delle API Rest. Nelle sezioni successive viene descritto come interagire con le API fornite dal sistema. Il `ROOT_PATH_G` sottinteso sarà sempre `home_url/api/v1/`. Ad esempio, per effettuare la GET dell'user `@user1`, l'indirizzo sarà:

```
GET home_url/api/v1/user/@user1
```

3.2.1 User

User è la risorsa utente. È possibile visualizzare, aggiungere, modificare o rimuovere gli utenti tramite una semplice richiesta HTTP.

3.2.1.1 Visualizzazione

È possibile visualizzare i dati di un utente tramite la richiesta

```
GET /user/<id>
```

3.2.1.1.1 Esempio di input

Un esempio di richiesta è

```
GET /user/abcd@bc.it
```

senza alcun dato nel `PAYLOAD_G` della richiesta.

3.2.1.1.2 Esempio di output

In caso la richiesta vada a buon fine, un esempio di output è

```
1 {
2   "_id": {
3     "$oid": "5cd07bb9c331755af7f5ea00"
4   },
5   "name": "Mattia",
6   "surname": "Cruciani",
7   "email": "abcd@bc.it",
8   "telegram": "1234",
9   "admin": false,
10  "preference": "email",
11  "irreperibilita": [
12    {
```

```
13         "$date": "2018-12-05"
14     },
15     {
16         "$date": "2019-04-08"
17     },
18     {
19         "$date": "2019-04-15"
20     },
21     {
22         "$date": "2019-04-07"
23     },
24     {
25         "$date": "2019-06-07"
26     },
27     {
28         "$date": "2019-06-08"
29     },
30     {
31         "$date": "2019-06-09"
32     }
33 ],
34 "projects": [
35     {
36         "url": "http://localhost/gitlab/gitlab-2",
37         "priority": 2,
38         "topics": [
39             "java",
40             "coding",
41             "criptovaluta"
42         ],
43         "keywords": [
44             "kw1",
45             "kw2",
46             "kw3"
47         ]
48     }
49 ]
50 }
```

Nel caso l'utente richiesto non venga trovato, verrà restituito il seguente messaggio

```
1 {
2     "error": "Utente inesistente."
3 }
```

3.2.1.2 Inserimento

È possibile inserire un nuovo utente tramite la richiesta:

POST /user

È inoltre possibile dare i seguenti campi di tipo stringa alla richiesta, per aggiungere in fase di creazione i dati:

- name
- surname
- telegram
- email

Almeno uno tra i campi Email e ID Telegram vanno fornite insieme al payload.

3.2.1.2.1 Esempio di input

Un esempio di richiesta è

POST /user

con i seguenti dati nel corpo della richiesta

```
1 {  
2   "name": "Matteo",  
3   "surname": "Marchiori",  
4   "email": "matteo.marchiori97@gmail.com",  
5   "telegram": "123456"  
6 }
```

3.2.1.2.2 Esempio di output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {  
2   "ok": "Utente inserito correttamente"  
3 }
```

Nel caso l'utente da inserire esista già, verrà restituito il seguente messaggio

```
1 {  
2   "error": "L'utente inserito esiste già."  
3 }
```

Nel caso non vengano inseriti almeno email o telegram, verrà restituito il seguente messaggio

```
1 {  
2   "error": "Si prega di inserire almeno email o telegram per inserire  
3     l'utente."  
4 }
```

3.2.1.3 Modifica

È possibile modificare un utente tramite la richiesta

PUT /user/<id>

È possibile dare i seguenti campi di tipo stringa alla richiesta, per aggiungere in fase di modifica i dati:

- name
- surname
- telegram
- email

3.2.1.3.1 Esempio di input

Un esempio di richiesta è

PUT /user/1234

con i seguenti dati nel corpo della richiesta

```
1 {  
2   "name": "Marco",  
3   "surname": "Rossi",  
4   "telegram": "1235"  
5 }
```

3.2.1.3.2 Esempio di output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {  
2   "ok": "Utente modificato correttamente"  
3 }
```

In caso non vengano forniti almeno Email o ID Telegram, verrà restituito il seguente messaggio

```
1 {  
2   "error": "Si prega di inserire almeno email o telegram per  
3     modificare l'utente."  
}
```

In caso almeno uno degli identificativi forniti coincida con quello di un altro utente, verrà restituito il seguente messaggio

```
1 {  
2   "error": "I dati inseriti confliggono con altri già esistenti."  
3 }
```

3.2.1.4 Rimozione

È possibile rimuovere un utente dal sistema Butterfly con la richiesta

DELETE /user/<id>

Se il campo <id> corrisponde a un ID presente nel sistema, esso verrà rimosso.

3.2.1.4.1 Esempio di input

Un esempio di richiesta è

DELETE /user/abcd@bc.it

senza alcun dato nel PAYLOAD_G della richiesta.

3.2.1.4.2 Esempio di output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {  
2   "ok": "Utente rimosso correttamente"  
3 }
```


3.2.1.5 Riepilogo

Metodo HTTP	URI	Action
GET	/user/<id>	Restituisce un payload in JSON dell'utente che corrisponde a <id>
POST	/user	Inserisce un nuovo utente. È necessario fornire uno tra i campi telegram o email
PUT	/user/<id>	Modifica l'utente corrispondente a <id> con i campi passati nella richiesta
DELETE	/user/<id>	Elimina l'utente corrispondente a <id> dal sistema

Tabella 3: Riepilogo delle Rest API per la risorsa User

3.2.2 Project

Project è la risorsa relativa ai progetti. È possibile visualizzare o rimuovere i progetti tramite una semplice richiesta HTTP.

3.2.2.1 Visualizzazione

È possibile visualizzare i progetti tramite la richiesta

GET /project/<id>

Se il campo <id> corrisponde a un progetto, verranno mostrati i dati relativi a tale progetto.

3.2.2.1.1 Esempio di input

Un esempio di richiesta è

GET /project/http://localhost/gitlab/gitlab-2

senza alcun dato nel PAYLOAD_G della richiesta.

3.2.2.1.2 Esempio di output

In caso la richiesta vada a buon fine, un esempio di output è

```
1 {
2   "_id": {
3     "$oid": "5cd1b509c331756598e9c00b"
4   },
5   "url": "http://localhost/gitlab/gitlab-2",
6   "name": "Gitlab-2",
7   "app": "gitlab",
8   "topics": [
9     "java",
10    "coding",
11    "enhancement",
12    "bug"
13  ]
14 }
```

In caso il progetto richiesto non esista, verrà restituito il seguente messaggio

```
1 {  
2   "error": "Progetto inesistente."  
3 }
```

3.2.2.2 Rimozione

È possibile rimuovere un progetto dal sistema Butterfly con la richiesta

DELETE /project/<id>

Se il campo <id> corrisponde a un progetto, esso verrà rimossa dal sistema.

3.2.2.2.1 Esempio di input

Un esempio di richiesta è

DELETE /project/http://localhost/gitlab/gitlab-2

senza alcun dato nel PAYLOAD_G della richiesta.

3.2.2.2.2 Esempio output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {  
2   "ok": "Progetto rimosso correttamente"  
3 }
```

3.2.2.3 Riepilogo

Metodo HTTP	URI	Action
GET	/project/<id>	Restituisce un payload in JSON del progetto corrispondente a <id>
DELETE	/project/<id>	Elimina il progetto corrispondente a <id> dal sistema

Tabella 4: Riepilogo delle Rest API per la risorsa Project

3.2.3 Preference

Preference è la risorsa preferenza. È possibile modificare le preferenze degli utenti tramite questa risorsa.

3.2.3.1 Inserimento

È possibile inserire un nuovo progetto tra le preferenze tramite la richiesta:

POST /preference

È inoltre richiesto dare i seguenti campi di tipo stringa alla richiesta, per aggiungere in fase di creazione i dati:

- user
- project

3.2.3.1.1 Esempio di input

Un esempio di richiesta è

POST /preference

con i seguenti dati nel corpo della richiesta

```
1 {  
2   "user": "1234",  
3   "project": "http://localhost/redmine/project-2"  
4 }
```

3.2.3.1.2 Esempio di output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {  
2   "ok": "Preferenza aggiunta correttamente"  
3 }
```

Nel caso il progetto esista già tra le preferenze, o non esista l'utente specificato, verrà restituito il seguente messaggio

```
1 {  
2   "error": "Progetto già presente o utente inesistente."  
3 }
```

Nel caso non venga specificato il progetto, verrà restituito il seguente messaggio

```
1 {  
2   "error": "Nessun progetto selezionato o progetto inesistente."  
3 }
```

3.2.3.2 Modifica

È possibile modificare le preferenze di un utente tramite la richiesta

PUT /preference/<id>

Le preferenze hanno i seguenti tipi:

- topics
- irreperibilita
- piattaforma

Il tipo di preferenza va indicato nel payload usando il campo “tipo”.

In base al tipo di preferenza è possibile dare i seguenti campi di tipo stringa alla richiesta, per aggiungere in fase di modifica i dati:

- topics
 - project
 - priority
 - topics
 - keywords
- irreperibilita
 - giorni
- piattaforma
 - platform

3.2.3.2.1 Esempio di input

Un esempio di richiesta è

PUT /preference/1234

con i seguenti dati nel corpo della richiesta

```
1 {
2   "tipo": "topics",
3   "project": "http://localhost/redmine/project-2",
4   "priority": "2",
5   "topics": "['bug','feature','banana']",
6   "keywords": "['keyword','42']"
7 }
```

Un esempio per l'irreperibilità è

PUT /preference/1234

con i seguenti dati nel corpo della richiesta

```
1 {
2   "tipo": "irreperibilita",
3   "giorni": "['2019-05-25','2019-05-26','2019-06-01']"
4 }
```

Invece un esempio per modificare la piattaforma preferita di ricezione dei messaggi è

PUT /preference/1234

con i seguenti dati nel corpo della richiesta

```
1 {
2   "tipo": "piattaforma",
3   "platform": "email"
4 }
```

3.2.3.2.2 Esempio di output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {
2   "ok": "Preferenza modificata correttamente"
3 }
```

Nel caso si stia cercando di impostare la piattaforma preferita non avendo registrato l'account verrà restituito uno dei seguenti messaggi

```
1 {
2   "error": "Telegram non presente nel sistema."
3 }
```

oppure

```
1 {
2   "error": "Email non presente nel sistema."
3 }
```

3.2.3.3 Rimozione

È possibile rimuovere un progetto dalle preferenze di un utente con la richiesta

```
DELETE /preference/<id>
```

Se il campo <id> corrisponde a un utente, il progetto verrà rimosso dalle sue preferenze.

3.2.3.3.1 Esempio di input

Un esempio di richiesta è

```
DELETE /preference/1234
```

con i seguenti dati nel corpo della richiesta

```
1 {  
2   "project": "http://localhost/redmine/project-2"  
3 }
```

3.2.3.3.2 Esempio output

In caso la richiesta vada a buon fine, verrà restituito il seguente messaggio

```
1 {  
2   "ok": "Preferenza rimossa correttamente."  
3 }
```

In caso non venga fornito un progetto nella richiesta, verrà restituito il messaggio

```
1 {  
2   "error": "Nessun progetto selezionato o progetto inesistente."  
3 }
```

In caso l'utente non abbia il progetto specificato tra le preferenze o non esista l'utente specificato verrà restituito il messaggio

```
1 {  
2   "error": "Progetto non presente nelle preferenze o utente  
           inesistente."  
3 }
```

3.2.3.4 Riepilogo

Metodo HTTP	URI	Action
POST	/preference	Aggiunge la preferenza all'utente specificato nella richiesta con i campi passati nella richiesta
PUT	/preference/<id>	Modifica la preferenza dell'utente corrispondente a <id> con i campi passati nella richiesta
DELETE	/preference/<id>	Rimuove la preferenza dell'utente corrispondente a <id>

Tabella 5: Riepilogo delle Rest API per la risorsa Preference

3.3 Piattaforma di messaggistica

3.3.1 Email

Per ricevere i messaggi di Butterfly tramite Email, è sufficiente fornire tramite l'interfaccia del Gestore Personale l'Email sulla quale si vuole ricevere la notifica.

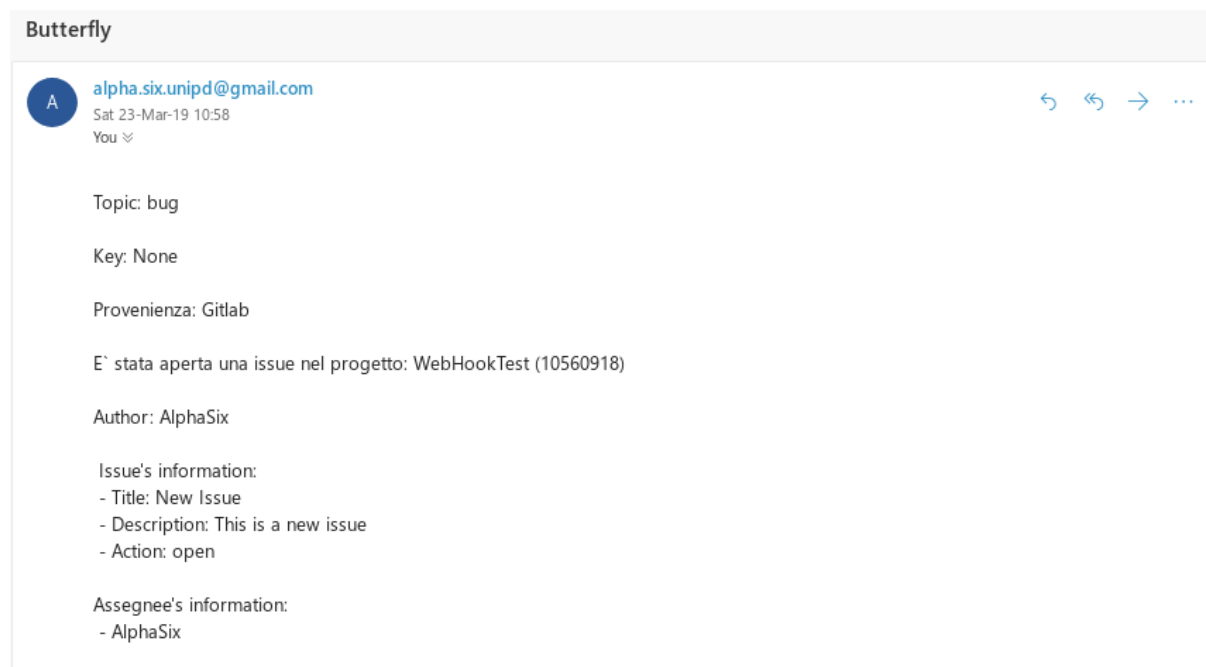


Figura 15: Formato dell' Email ricevuta da un utente finale

3.3.2 Telegram

Per ricevere le notifiche via Telegram, è necessario fare un passaggio aggiuntivo: va fornita l'autorizzazione al bot per poter inviare messaggi agli utenti. Il bot è raggiungibile al seguente link:

<http://t.me/ButterflyBot>

Dare il comando `/start` per dare l'autorizzazione di inoltro dei messaggi al bot. È necessario inoltre aggiungere tramite l'interfaccia del Gestore Personale il proprio account Telegram. In qualsiasi momento sarà possibile bloccare il bot in caso non si vogliano più ricevere messaggi relativi a Butterfly su Telegram, tramite le funzionalità dell'applicazione.



Topic: bug 11:04:57 AM

Key: None

Provenienza: Gitlab

È stata aperta una issue nel progetto:
WebHookTest (10560918)

Author: AlphaSix

Issue's information:

- **Title:** New Issue
- **Description:** This is a new issue
- **Action:** open

Assegnee's information:

- AlphaSix

Figura 16: Formato del messaggio Telegram ricevuto da un utente finale

Nel caso in cui si volesse utilizzare un altro bot, i passaggi da seguire possono essere trovati sulla pagina apposita della documentazione di Telegram²⁵. Per comunicare con questo bisogna modificare la variabile di ambiente `BUTTERFLY_CONSUMER_TELEGRAM_BOT` che rappresenta il `token` univoco del nuovo bot, come descritto in §2.5.2.

²⁵<https://core.telegram.org/bots>

4 Note sulla gestione delle notifiche

Considerando che un messaggio appartiene ad un progetto e possiede un Topic, un utente riceve una notifica se:

- L'utente è iscritto al Topic del messaggio
- L'utente è iscritto al progetto del messaggio e possiede la priorità del progetto più alta tra tutti gli utenti iscritti al progetto indicato
- L'utente è disponibile il giorno di invio del messaggio

In Kafka è presente una coda chiamata “lostmessages” nella quale vengono salvati i seguenti messaggi:

- Nel momento in cui viene inviata la segnalazione di un progetto o di una label non ancora registrato nel sistema, tale segnalazione non viene inviata a nessuno, questo perché il progetto viene salvato nel Gestore Personale solo dopo la prima segnalazione. Bisogna dunque tenere presente che la prima segnalazione di un nuovo progetto o label di una Issue non verrà inviata ad alcun utente
- Nel momento in cui nessun utente è interessato o disponibile a un dato progetto il giorno dell'invio della segnalazione

I messaggi nella coda “lostmessages” non possono essere direttamente letti dall'interfaccia del Gestore Personale, ma solo attraverso un Consumer apposito che l'attuale versione di Butterfly non possiede.

4.1 Caso di persona non disponibile

Seguendo le tre regole ad inizio paragrafo è comprensibile capire la logica per l'inoltro delle notifiche per indisponibilità. Per chiarezza ne esplichiamo comunque i meccanismi.

Nel caso in cui una persona a cui dovrebbe arrivare una notifica non sia disponibile (e quindi lo abbia precedentemente notificato nel calendario come descritto in §3.1.1.3), la procedura del possibile inoltro del messaggio avviene nel seguente modo:

- Se è presente almeno un altro utente iscritto al Topic legato alla segnalazione, disponibile il giorno dell'invio della notifica e con la stessa priorità di progetto, non viene inviata la notifica all'utente in questione perché almeno un altro utente riceverà la segnalazione
- Se non è disponibile nessun altro utente con la stessa priorità iscritto al Topic legato alla segnalazione, la notifica viene inoltrata ad un utente sempre iscritto allo stesso Topic e disponibile il giorno dell'invio della segnalazione, ma con una priorità di progetto inferiore
- Se non è disponibile alcun utente iscritto al Topic legato alla segnalazione e disponibile il giorno dell'invio della notifica, verrà notificata una persona iscritta al progetto indicato con priorità massima tra quelli disponibili

5 Segnalazione problematiche

Nel caso dovessero venire riscontrati bug o problematiche relative a Butterfly, si prega di segnalarlo tramite una delle seguenti procedure:

- Inviare una mail all'indirizzo alpha.six.unipd@gmail.com. Questa deve essere nel seguente formato:
 - Oggetto: [BUTTERFLY PROJECT]: <Nome significativo dell'evento da segnalare>
 - Corpo:
 - * [SUMMARY]: <Riepilogo di come è accaduto il fatto da segnalare>
 - * [TYPE]: <Di tipo: BUG | FIX | ENHANCEMENT >
 - * [PRIORITY]: <Di tipo LOW | MEDIUM | HIGH>
 - * [SEVERITY]: <Di tipo LOW | MEDIUM | HIGH>
 - * [DATE]: <Data in cui è stato riscontrato in formato ANNO-MESE-GIORNO>
 - * [DESCRIPTION]: <Descrizione completa del fatto da segnalare>
 - * [ENVIRONMENT]: <Descrizione del sistema sul quale è successo il fatto>
 - * [OTHER]: <Altre informazioni utili al report e descrizione di eventuali allegati esemplificativi>
- Aprire una issue nel progetto Butterfly su GitHub seguendo, come per le segnalazioni via Email, il seguente formato:
 - Titolo: <Nome significativo dell'evento da segnalare>
 - Corpo:
 - * [SUMMARY]: <Riepilogo di come è accaduto il fatto da segnalare>
 - * [TYPE]: <Di tipo: BUG | FIX | ENHANCEMENT >
 - * [PRIORITY]: <Di tipo LOW | MEDIUM | HIGH>
 - * [SEVERITY]: <Di tipo LOW | MEDIUM | HIGH>
 - * [DATE]: <Data in cui è stato riscontrato in formato ANNO-MESE-GIORNO>
 - * [DESCRIPTION]: <Descrizione completa del fatto da segnalare>
 - * [ENVIRONMENT]: <Descrizione del sistema sul quale è successo il fatto>
 - * [OTHER]: <Altre informazioni utili al report e descrizione di eventuali screenshot esemplificativi presenti>

A Glossario

B

Broker

Componente che gestisce i messaggi inviati da Publisher a Subscriber nel relativo modello architetturale. Mette a disposizione i `TopicG` nei quali i Publisher inviano i messaggi, mentre i Subscriber possono iscriversi a essi per ricevere i messaggi.

C

Cluster

Cluster è un insieme di computer connessi tra loro tramite una rete telematica.

Consumer

Componente applicativa che ha il compito di abbonarsi a determinati Topic su Kafka, recuperandone i messaggi.

Container

Con container si intende la componente con la capacità di eseguire più processi e applicazioni in modo separato per sfruttare al meglio l'infrastruttura esistente pur conservando il livello di sicurezza che sarebbe garantito dalla presenza di sistemi separati.

D

Docker

Piattaforma che consente di automatizzare il deployment di applicazioni all'interno di container software.

Docker Compose

Definisce le configurazioni necessarie per come devono essere eseguite le immagini contenute nei container Docker, i link e le porte esposte verso l'esterno.

I

ID

Termine che indica “identificativo”.

J

JSON

Acronimo di “JavaScript Object Notation”, è un formato adatto all’interscambio di dati fra applicazioni client/server. È basato sul linguaggio JavaScript Standard ma ne è indipendente.

K

Kubernetes

Kubernetes è uno strumento open source di orchestrazione e gestione di container. È stato sviluppato dal team di Google ed è uno dei tool più utilizzati a questo scopo.

M

Metadato

Particolare dato che descrive insiemi di altri dati.

N

Namespace

Ambiente per organizzare i file sorgente secondo una gerarchia specifica o il loro tipo. Un namespace, ad esempio, risulta essere simile alla gerarchia delle cartelle di un qualunque computer.

P

Payload

Il “carico utile” (in inglese payload) è un termine utilizzato (per analogia dal mondo dei trasporti) per indicare la parte di dati trasmessi effettiva che è destinata all’utente, in contrasto con i metadati e con gli header che servono esclusivamente a far funzionare il protocollo di comunicazione.

Plug and play

Termine usato per indicare una tecnologia di cui non è necessario conoscere il funzionamento per eseguirla, basta collegarla al sistema che la utilizza.

Plugin

È un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie.

Pod

All'interno di Kubernetes un Pod è un insieme di più container con una rete e memoria condivisa. All'interno di un Pod sono contenute anche le varie configurazioni per eseguire i container al suo interno.

Prodotto

Risultato finito e funzionante del lavoro fatto durante tutto lo svolgimento di un progetto. Il prodotto è ciò che viene consegnato al cliente alla fine.

Producer

Componente applicativa che ha il compito di raccogliere dei messaggi per pubblicarli su determinati Topic di Kafka.

Push

Comando che aggiorna i riferimenti remoti di una o più repository con i riferimenti locali, mandando gli oggetti necessari per allineare il branch remoto a quello locale.

R

Rancher

Rancher è un software di gestione di oggetti di Kubernetes che fornisce un'interfaccia grafica più ricca rispetto a quella di Dockstation per gestire i container Docker e offre ulteriori funzionalità.

Root path

Con il termine inglese “root path” si intende il percorso di root (dalla radice).

T

Topic

Equivalente di “argomento” in italiano.

U



Update

Equivalente di “aggiornamento” in italiano.

W

Webhook

Metodo per aumentare o modificare il comportamento di una pagina o applicazione Web con chiamate HTTP esterne in modo semplice, standardizzato e intelligente (callback).