



*dstackgroup@gmail.com*

# Manuale Utente

## Informazioni sul documento

<b>Nome documento</b>	Manuale Utente
<b>Versione</b>	v1.0.0
<b>Data approvazione</b>	2019-05-14
<b>Responsabili</b>	Harwinder Singh
<b>Redattori</b>	Federico Rispo Elton Stafa Niccolò Vettorello
<b>Verificatori</b>	Enrico Trinco
<b>Stato</b>	Approvato
<b>Lista distribuzione</b>	Gruppo <i>DStack</i> <i>Prof. Tullio Vardanega</i> <i>Prof. Riccardo Cardin</i> <i>Imola Informatica S.P.A.</i>
<b>Uso</b>	Esterno

## Sommario

Il presente documento contiene la definizione delle funzionalità di Butterfly, ed una guida all'utilizzo di questo software.

# Diario delle Modifiche

Versione	Descrizione	Nominativo	Ruolo	Data
v1.0.0	<b>Approvazione per il rilascio RA</b>	Harwinder Singh	<i>Responsabile di Progetto</i>	2019-05-14
v0.2.0	<b>Verifica superata</b>	Enrico Trinco	<i>Verificatore</i>	2019-05-05
v0.1.4	<b>Incremento §B e §2</b>	Elton Stafa	<i>Programmatore</i>	2019-05-01
v0.1.3	<b>Incremento §D</b>	Elton Stafa	<i>Programmatore</i>	2019-04-27
v0.1.2	<b>Incremento §5</b>	Federico Rispo	<i>Programmatore</i>	2019-04-25
v0.1.1	<b>Aggiunta §4</b>	Niccolò Vettorello	<i>Programmatore</i>	2019-04-24
v0.1.0	<b>Verifica superata</b>	Harwinder Singh	<i>Verificatore</i>	2019-04-06
v0.0.6	<b>Stesura §B</b>	Elton Stafa	<i>Programmatore</i>	2019-04-05
v0.0.5	<b>Stesura §2</b>	Elton Stafa	<i>Programmatore</i>	2019-04-04
v0.0.4	<b>Stesura §3</b>	Niccolò Vettorello	<i>Programmatore</i>	2019-04-03
v0.0.3	<b>Stesura §5</b>	Niccolò Vettorello	<i>Programmatore</i>	2019-04-02
v0.0.2	<b>Stesura §D e §1</b>	Eleonora Signor	<i>Programmatore</i>	2019-04-02
v0.0.1	<b>Creazione scheletro del documento</b>	Enrico Trinco	<i>Amministratore di Progetto</i>	2019-03-25

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento	1
1.2	Scopo del prodotto	1
1.3	Riferimenti informativi	1
1.4	Elenco definizioni utili	2
<b>2</b>	<b>Requisiti minimi di sistema</b>	<b>2</b>
2.1	Prerequisiti	2
2.2	Requisiti hardware	2
2.3	Requisiti software	2
<b>3</b>	<b>Installazione ed esecuzione</b>	<b>2</b>
3.1	Configurazione ambiente per l'esecuzione	3
3.1.1	Installazione Docker	3
3.1.1.1	Windows:	3
3.1.1.2	Linux:	3
3.1.2	Installazione Docker Compose	3
3.1.2.1	Windows:	3
3.1.2.2	Linux:	3
3.2	Installazione Butterfly	3
3.3	Esecuzione dei servizi di terze parti	3
<b>4</b>	<b>Buttercli</b>	<b>5</b>
4.1	Guida all'installazione	5
4.2	Prompt contestuali d'aiuto	5
4.2.1	General help	7
4.2.2	User help	8
4.2.3	User create help	9
4.2.4	User find help	10
4.2.5	User remove help	11
4.2.6	User update help	12
4.2.7	User list help	13
4.2.8	Project help	14
4.2.9	Project create help	15
4.2.10	Project find help	16
4.2.11	Project remove help	17
4.2.12	Project update help	18
4.2.13	Project list help	19
4.2.14	User-contacts help	20
4.2.15	User-contacts create help	21
4.2.16	User-contacts find help	22
4.2.17	User-contacts remove help	23
4.2.18	User-contacts update help	24
4.2.19	Subscriptions help	25
4.2.20	Subscriptions create help	26
4.2.21	Subscriptions find help	27
4.2.22	Subscriptions remove help	28
4.2.23	Subscriptions update help	29

<b>5</b>	<b>Guida all'utilizzo</b>	<b>30</b>
5.1	GET /users	33
5.1.1	cURL	33
5.1.2	Buttercli	33
5.2	POST /users	34
5.2.1	cURL	34
5.2.2	Buttercli	35
5.3	GET /users/{email}	36
5.3.1	cURL	36
5.3.2	Buttercli	36
5.4	PATCH /users/{email}	38
5.4.1	cURL	38
5.4.2	Buttercli	38
5.5	DELETE /users/{email}	40
5.5.1	cURL	40
5.5.2	Buttercli	40
5.6	GET /user-contacts/{userEmail}	41
5.6.1	cURL	41
5.6.2	Buttercli	41
5.7	POST /user-contacts/{contactService}	43
5.7.1	cURL	43
5.7.2	Buttercli	43
5.8	PUT /user-contacts/{userEmail}/{contactService}	45
5.8.1	cURL	45
5.8.2	Buttercli	46
5.9	DELETE /user-contacts/{userEmail}/{contactService}	47
5.9.1	cURL	47
5.9.2	Buttercli	47
5.10	GET /projects	48
5.10.1	cURL	48
5.10.2	Buttercli	48
5.11	POST /projects	49
5.11.1	cURL	49
5.11.2	Buttercli	49
5.12	GET /projects/{projectName}	51
5.12.1	cURL	51
5.12.2	Buttercli	51
5.12.3	PUT /projects/projectName	52
5.12.4	cURL	52
5.12.5	Buttercli	52
5.12.6	DELETE /projects/projectName	54
5.12.7	cURL	54
5.12.8	Buttercli	54
5.12.9	DELETE /projects/projectName/producerService	55
5.12.10	cURL	55
5.12.11	Buttercli	55
5.13	POST /subscriptions	57
5.13.1	cURL	57
5.13.2	Buttercli	58

5.14	GET /subscriptions/users/{userEmail}/projects/{projectName}/event-types/{eventType}	59
5.14.1	cURL	59
5.14.2	Buttercli	60
5.15	PATCH /subscriptions/users/{userEmail}/projects/{projectName}/event-types/{eventType}	61
5.15.1	cURL	61
5.15.2	Buttercli	62
5.16	DELETE /subscriptions/users/{userEmail}/projects/{projectName}/event-types/{eventType}	63
5.16.1	cURL	63
5.16.2	Buttercli	64
<b>6</b>	<b>Esempi notifiche</b>	<b>65</b>
6.1	Esempio di notifica Telegram	65
6.2	Esempio di notifica Slack	65
6.3	Esempio di notifica Email	65
<b>A</b>	<b>Creazione eventi</b>	<b>65</b>
A.1	Apertura issue su Redmine	65
A.2	Aggiornare una issue su Redmine	66
A.3	Apertura issue su Gitlab	66
A.4	Modifica issue su Gitlab	66
A.5	Invio commit su Gitlab	66
A.6	Creazione analisi su Sonarqube	67
A.7	Avvio analisi con Sonarqube	67
<b>B</b>	<b>Risoluzione degli errori</b>	<b>67</b>
<b>C</b>	<b>Segnalazioni agli sviluppatori</b>	<b>68</b>
<b>D</b>	<b>Definizioni utili</b>	<b>69</b>

## Elenco delle figure

1	Esempio di notifica Telegram	65
2	Esempio di notifica Slack	65
3	esempio di notifica Email	65
4	Esempio di issue templatizzata	68

## Elenco delle tabelle

3	Tabella USERS API	30
4	Tabella USER-CONTACTS API	31
5	Tabella PROJECTS API	31
6	Tabella SUBSCRIPTIONS API	32

# 1 Introduzione

## 1.1 Scopo del documento

Questo documento nasce con lo scopo di illustrare tutte le funzionalità di Butterfly, fornendo così all'utente finale indicazioni per il corretto uso del software. La versione attuale rappresenta la prima bozza di un documento che, una volta completato, guiderà il fruitore del prodotto nell'utilizzo.

## 1.2 Scopo del prodotto

Butterfly nasce dall'esigenza di uniformare e accentrare la gestione delle segnalazioni generate a partire da sistemi di terze parti, quali Redmine, GitLab e SonarQube. Questi strumenti sono parte integrante dei processi gestionali, di versionamento e di Continuous Integration dell'azienda committente. La maggior parte di essi fornisce già dei meccanismi di notifica ed inoltro delle possibili segnalazioni, sono configurabili e accessibili da dashboard molto diverse tra loro, di difficile interazione e anche con limitazione di accessibilità. Inoltre, in caso di segnalazioni di bug in ambienti di produzione è fondamentale assicurarsi che gli sviluppatori in grado di risolvere il problema siano segnalati tempestivamente, senza aspettare che loro accedano a qualche dashboard specifica. Il gruppo *DStack* si propone quindi di sviluppare una rete di soluzioni che offrano un'interfaccia condivisa, estendibile per gestire le segnalazioni relative alla pipeline di sviluppo software di *Imola Informatica S.P.A.*. Questa interfaccia deve inoltre permettere una configurazione automatica e personalizzabile di tali segnalazioni.

## 1.3 Riferimenti informativi

- *Docker*<sub>G</sub>:  
<https://www.docker.com/>;
- *Docker Compose*<sub>G</sub>:  
<https://docs.docker.com/compose/>
- *GitLab*<sub>G</sub>:  
<https://about.gitlab.com/>
- *Redmine*<sub>G</sub>:  
<https://www.redmine.org/>
- *SonarQube*<sub>G</sub>:  
<https://www.sonarqube.org/>
- *Telegram*<sub>G</sub>:  
<https://telegram.org/>
- *Slack*<sub>G</sub>:  
<https://slack.com/>

### 1.4 Elenco definizioni utili

Nella parte conclusiva di questo documento, precisamente all'appendice D, viene riportato un breve elenco di vocaboli, corredati dal rispettivo significato, con l'intento di agevolare il lettore nella piena comprensione del testo. Considerata l'utenza finale di Butterfly, il gruppo ha deciso di omettere quei termini che sono noti a chiunque possieda conoscenze basilari di informatica. Tali termini sono marcati con una piccola 'G' posta a pedice.

## 2 Requisiti minimi di sistema

Per poter utilizzare Butterfly sono richiesti i seguenti requisiti minimi di sistema:

### 2.1 Prerequisiti

È necessario che sia abilitata la virtualizzazione a livello software e/o hardware<sup>1</sup>.

### 2.2 Requisiti hardware

- Almeno 16 GB di RAM.

### 2.3 Requisiti software

- **Sistema operativo:** Windows, MacOS o GNU/Linux;
- **Docker:** ad una versione maggiore od uguale alla 18.09.0;
- **Docker Compose:** ad una versione maggiore o uguale alla 1.24.0.

## 3 Installazione ed esecuzione

Questa sezione serve ad elencare i passi da completare per poter utilizzare l'intero sistema<sup>2</sup>. Sono necessarie multiple istanze di terminale: vanno infatti eseguiti separatamente i moduli *Java<sub>G</sub>*, lo *User Manager<sub>G</sub>* e i servizi di terze parti. Per poter eseguire il software va clonata la *repository<sub>G</sub>* del progetto sul proprio computer. Essa è raggiungibile mediante l'URL:

<https://github.com/dstack-group/Butterfly>

L'operazione di clonazione avviene tramite il comando:

```
git clone https://github.com/dstack-group/Butterfly.git
```

---

<sup>1</sup>Il procedimento da utilizzare per abilitare l'opzione di virtualizzazione è dipendente sia dal sistema operativo che dalla macchina in uso: la sua illustrazione va oltre lo scopo del presente *Manuale Utente*.

<sup>2</sup>In caso ci si trovi in ambiente GNU/Linux si rimanda il lettore alla pagina <https://docs.docker.com/install/linux/linux-postinstall/> per configurare opzioni di post-installazione del software *Docker*.

## 3.1 Configurazione ambiente per l'esecuzione

### 3.1.1 Installazione Docker

Nel caso in cui Docker non soddisfi la versione richiesta oppure non è presente nel sistema, puoi fare riferimento a questi link per scaricare Docker nel tuo sistema operativo.

**3.1.1.1 Windows:** <https://runnable.com/docker/install-docker-on-windows-10>

**3.1.1.2 Linux:** <https://runnable.com/docker/install-docker-on-linux>

### 3.1.2 Installazione Docker Compose

Nel caso in cui Docker non soddisfi la versione richiesta oppure non è presente nel sistema, puoi fare riferimento a questi link per scaricare Docker nel tuo sistema operativo.

Prerequisito fondamentale è la presenza di **Docker** già installato sul proprio dispositivo. Se non è ancora installato fare riferimento alla sotto sezione precedente.

**3.1.2.1 Windows:** <https://docs.docker.com/compose/install/>

**3.1.2.2 Linux:** <https://linuxize.com/post/how-to-install-and-use-docker-compose-on-ubuntu-18-04/>

## 3.2 Installazione Butterfly

Per installare le dipendenze di Butterfly nel proprio ambiente, e per generare il programma eseguibile del terminale, è necessario eseguire il seguente comando all'interno della root directory del progetto:

```
./run.sh install
```

Per l'esecuzione dei servizi invece, eseguire il seguente comando:

```
./run.sh --build prod
```

## 3.3 Esecuzione dei servizi di terze parti

1. Spostarsi nella cartella *third-party-services* con il comando:

```
cd third-party-services
```

2. Per ognuno dei tre servizi (*gitlab*, *sonarqube*, *redmine*) sono disponibili i seguenti comandi:

- `./third-party-service.sh start`  
Manda in esecuzione il servizio (ad esempio `./gitlab.sh start` per *GitLab*);
- `./third-party-service.sh stop`  
Ferma l'esecuzione del servizio (ad esempio `./redmine.sh stop` per *Redmine*);



- `./third-party-service.sh reset`  
Resetta il servizio (ad esempio `./sonarqube.sh reset` per *SonarQube*);
- `./third-party-service.sh logs`  
Mostra le informazioni di *log* per il servizio (ad esempio `./gitlab.sh logs` per *GitLab*)

3. Una volta che i servizi sono stati mandati in esecuzione, è possibile accedervi:

- **Redmine:** aprendo <http://localhost:15000> ed inserendo le credenziali:
  - Username: admin;
  - Password: admin.
- **GitLab:** aprendo <http://localhost:10443>: lo Username è root, mentre la Password deve essere impostata al primo avvio;
- **SonarQube:** aprendo <http://localhost:15002> ed inserendo le credenziali:
  - Username: admin;
  - Password: admin.

## 4 Buttercli

Il progetto *Butterfly* mette a disposizione una interfaccia a linea di comando, denominata *Buttercli*, che permette di interagire con il sistema in maniera più facile ed intuitiva. Tale interfaccia è stata realizzata con lo scopo di agevolare l'utente che si accinge ad utilizzare il software, nascondendo quei dettagli ritenuti troppo tecnici e/o non strettamente necessari.

Nei paragrafi seguenti è riportata una spiegazione della modalità di installazione dell'interfaccia ed una breve presentazione dei prompt di aiuto contestuale, che forniscono indicazioni di dettaglio sui parametri e sul loro utilizzo. Ulteriori illustrazioni relative al funzionamento della *cli* e a caratteristiche prettamente tecniche esulano dai fini di questo Manuale e sono state pertanto tralasciate.

### 4.1 Guida all'installazione

Presentiamo l'elenco dei passi da seguire per installare *Buttercli*:

1. Posizionarsi nella root directory del progetto;
2. Eseguire da terminale il comando:

```
./run.sh install
```

3. Posizionarsi nella cartella `./user-manager/buttercli`;
4. All'interno di tale cartella sarà presente il file eseguibile `buttercli`, che non richiede alcuna dipendenza esterna.

### 4.2 Prompt contestuali d'aiuto

L'interfaccia *Buttercli* fornisce dei prompt esplicativi che consentono di avere maggiori informazioni sui comandi utilizzabili e sui parametri accettati, obbligatori o meno. Viene presentata una lista riassuntiva di tutte le schermate d'aiuto disponibili: tale elenco esprime la natura gerarchica degli stessi prompt, e, per ogni voce, rimanda alla sezione apposita.

- General help
  - User help
    - \* User create help
    - \* User find help
    - \* User remove help
    - \* User update help
    - \* User list help
  - Project help
    - \* Project create help
    - \* Project find help
    - \* Project remove help
    - \* Project list help
    - \* Project update help
  - User-contacts help

- \* User-contacts create help
- \* User-contacts find help
- \* User-contacts remove help
- \* User-contacts update help
- Subscriptions help
  - \* Subscriptions create help
  - \* Subscriptions find help
  - \* Subscriptions remove help
  - \* Subscriptions update help

### 4.2.1 General help

Informazioni di carattere generale sull'interfaccia e sulle operazioni messe a disposizione.

A CLI to interact with the Butterfly User Manager REST API

#### USAGE

```
$ buttercli [COMMAND]
```

#### COMMANDS

config	Manage buttercli settings
contact	Perform contact related operations
help	display help for buttercli
init	Initialize the db and set all the server settings
project	Perform project related operations
sub	Perform subscription related operations
user	Perform user related operations

A CLI to interact with the Butterfly User Manager REST API

### 4.2.2 User help

Informazioni sui comandi che permettono di effettuare operazioni sugli utenti.

Perform user related operations

#### USAGE

```
$ buttercli user:COMMAND
```

#### COMMANDS

user:create	Create a new user
user:find	Find a specific user identified by email
user:list	Show a list of all users
user:remove	Remove a specific user identified by email
user:update	Update an existing user identified by email

### 4.2.3 User create help

Informazioni di dettaglio relative alla creazione di un utente.

Create a new user

#### USAGE

```
$ buttercli user:create
```

#### OPTIONS

<code>-a, --[no-]available</code>	the new user is currently available (default is true)
<code>-e, --email=email</code>	(required) new user email address
<code>-f, --firstname=firstname</code>	(required) new user first name (max 30 character)
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-l, --lastname=lastname</code>	(required) new user last name (max 30 characters)
<code>-x, --extended</code>	show extra columns

#### 4.2.4 User find help

Informazioni di dettaglio relative alla ricerca di un utente.

Find a specific user identified by email

##### USAGE

```
$ buttercli user:find
```

##### OPTIONS

<code>-e, --email=email</code>	(required) user email address
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-x, --extended</code>	show extra columns

#### 4.2.5 User remove help

Informazioni di dettaglio relative alla rimozione di un utente.

Remove a specific user identified by email

##### USAGE

```
$ buttercli user:remove
```

##### OPTIONS

<code>-e, --email=email</code>	(required) user email address
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format



#### 4.2.6 User update help

Informazioni di dettaglio relative alla modifica di un utente.

Update an existing user identified by email

##### USAGE

```
$ buttercli user:update
```

##### OPTIONS

<code>-a, --[no-]available</code>	the user is currently available (default is true)
<code>-e, --email=email</code>	(required) user email address
<code>-f, --firstname=firstname</code>	new first name (max 30 characters)
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-l, --lastname=lastname</code>	new last name (max 30 characters)
<code>-x, --extended</code>	show extra columns

#### 4.2.7 User list help

Informazioni di dettaglio relative alla ricerca degli utenti registrati.

Show a list of all users

##### USAGE

```
$ buttercli user:list
```

##### OPTIONS

<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-x, --extended</code>	show extra columns

### 4.2.8 Project help

Informazioni sui comandi che permettono di effettuare operazioni sui progetti.

Perform project related operations

#### USAGE

```
$ buttercli project:COMMAND
```

#### COMMANDS

<code>project:create</code>	Create a new project specifying one or more service urls
<code>project:find</code>	Find a specific project identified by name
<code>project:list</code>	Show a list of all projects
<code>project:remove</code>	Remove an existing project or only the service urls specified
<code>project:update</code>	Update an existing service url of the project identified by name

#### 4.2.9 Project create help

Informazioni di dettaglio relative alla creazione di un progetto.

Create a new project specifying one or more service urls

##### USAGE

```
$ buttercli project:create
```

##### OPTIONS

<code>-g, --gitlab=gitlab</code>	gitlab url of the new project
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-n, --name=name</code>	(required) project name (max 50 characters)
<code>-r, --redmine=redmine</code>	redmine url of the new project
<code>-s, --sonarqube=sonarqube</code>	sonarqube url of the new project
<code>-x, --extended</code>	show extra columns

#### 4.2.10 Project find help

Informazioni di dettaglio relative alla ricerca di un progetto.

Find a specific project identified by name

##### USAGE

```
$ buttercli project:find
```

##### OPTIONS

<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-n, --name=name</code>	(required) project name (max 50 characters)
<code>-x, --extended</code>	show extra columns

#### 4.2.11 Project remove help

Informazioni di dettaglio relative alla rimozione di un progetto.

Remove an existing project or only the service urls specified

##### USAGE

```
$ buttercli project:remove
```

##### OPTIONS

<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-n, --name=name</code>	(required) project name (max 50 characters)
<code>-s, --service=service</code>	(required) remove service between GITLAB, REDMINE, SONARQUBE or ALL of them
<code>-x, --extended</code>	show extra columns

#### 4.2.12 Project update help

Informazioni di dettaglio relative alla modifica di un progetto.

Update an existing service url of the project identified by name

##### USAGE

```
$ buttercli project:update
```

##### OPTIONS

<code>-g, --gitlab=gitlab</code>	new gitlab url of the project
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-n, --name=name</code>	(required) project name (max 50 characters)
<code>-r, --redmine=redmine</code>	new redmine url of the project
<code>-s, --sonarqube=sonarqube</code>	new sonarqube url of the project
<code>-x, --extended</code>	show extra columns

#### 4.2.13 Project list help

Informazioni di dettaglio relative alla ricerca degli progetti registrati.

Show a list of all projects

##### USAGE

```
$ buttercli project: list
```

##### OPTIONS

<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-x, --extended</code>	show extra columns



#### 4.2.14 User-contacts help

Informazioni sui comandi che permettono di effettuare operazioni sulle informazioni di contatto.

Perform contact related operations

##### USAGE

```
$ buttercli contact:COMMAND
```

##### OPTIONS

<code>contact:create</code>	Create a new user contact
<code>contact:find</code>	Find all contacts of a specific user identified by email
<code>contact:remove</code>	Remove an existing user contact specified by user email and contact service
<code>contact:update</code>	Update an existing user contact account specified by user email and contact platform

#### 4.2.15 User-contacts create help

Informazioni di dettaglio relative all'aggiunta di informazioni di contatto per un utente.

Create a new user contact

##### USAGE

```
$ buttercli contact:create
```

##### OPTIONS

<code>-a, --account=account</code>	(required) new identifier for the service specified
<code>-e, --email=email</code>	(required) user email address
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-p, --platform=platform</code>	(required) choose the contact platform between SLACK, EMAIL. The TELEGRAM contact can only be set from the Telegram Bot
<code>-x, --extended</code>	show extra columns

#### 4.2.16 User-contacts find help

Informazioni di dettaglio relative alla ricerca di informazioni di contatto di un utente.

Find all contacts of a specific user identified by email

##### USAGE

```
$ buttercli contact:find
```

##### OPTIONS

<code>-e, --email=email</code>	(required) user email address
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-x, --extended</code>	show extra columns

#### 4.2.17 User-contacts remove help

Informazioni di dettaglio relative alla rimozione di informazioni di contatto di un utente.

Remove an existing user contact specified by user email and contact platform

##### USAGE

```
$ buttercli contact:remove
```

##### OPTIONS

<code>-e, --email=email</code>	(required) user email address
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-p, --platform=platform</code>	(required) choose the contact platform to delete between SLACK, EMAIL, TELEGRAM

#### 4.2.18 User-contacts update help

Informazioni di dettaglio relative alla modifica di informazioni di contatto di un utente.

Update an existing user contact account specified by user email and contact platform

##### USAGE

```
$ buttercli contact:update
```

##### OPTIONS

<code>-a, --account=account</code>	(required) new identifier for the service specified
<code>-e, --email=email</code>	(required) user email address
<code>-h, --help</code>	show CLI help
<code>-j, --json</code>	display results in json format
<code>-p, --platform=platform</code>	(required) choose the contact platform to update between SLACK, EMAIL, TELEGRAM
<code>-x, --extended</code>	show extra columns

#### 4.2.19 Subscriptions help

Informazioni sui comandi che permettono di effettuare operazioni sulle sottoscrizioni.

Perform subscription related operations

##### USAGE

```
$ buttercli sub:COMMAND
```

##### COMMANDS

sub:create	Create a new subscription
sub:find	Find an existing subscription
sub:remove	Remove an existing subscription
sub:update	Update an existing subscription

#### 4.2.20 Subscriptions create help

Informazioni di dettaglio relative alla creazione di una sottoscrizione.

Create a new subscription

##### USAGE

```
$ buttercli sub:create
```

##### OPTIONS

- `-P, --priority=priority` (required) set the priority between LOW, MEDIUM, HIGH
  
- `-c, --contact=contact` (required) choose the contact service between EMAIL, SLACK, TELEGRAM where the notification will be sent
  
- `-e, --email=email` (required) user email address
  
- `-h, --help` show CLI help
  
- `-j, --json` display results in json format
  
- `-k, --keyword=keyword` (required) keyword compared with the contents of the future events
  
- `-p, --project=project` (required) project name
  
- `-t, --event-type=event-type` (required) set the event type between  
GITLAB\_COMMIT\_CREATED  
GITLAB\_ISSUE\_CREATED  
GITLAB\_ISSUE\_EDITED  
GITLAB\_MERGE\_REQUEST\_CREATED  
GITLAB\_MERGE\_REQUEST\_EDITED  
GITLAB\_MERGE\_REQUEST\_MERGED  
GITLAB\_MERGE\_REQUEST\_CLOSED  
REDMINE\_TICKET\_CREATED  
REDMINE\_TICKET\_EDITED  
SONARQUBE\_PROJECT\_ANALYSIS\_COMPLETED
  
- `-x, --extended` show extra columns

#### 4.2.21 Subscriptions find help

Informazioni di dettaglio relative alla ricerca di una sottoscrizione.

Find an existing subscription

USAGE

```
$ buttercli sub:find
```

OPTIONS

```
-e, --email=email          (required) user email address

-h, --help                  show CLI help

-j, --json                  display results in json format

-p, --project=project      (required) project name

-t, --event-type=event-type (required) set the event type between
                             GITLAB_COMMIT_CREATED
                             GITLAB_ISSUE_CREATED
                             GITLAB_ISSUE_EDITED
                             GITLAB_MERGE_REQUEST_CREATED
                             GITLAB_MERGE_REQUEST_EDITED
                             GITLAB_MERGE_REQUEST_MERGED
                             GITLAB_MERGE_REQUEST_CLOSED
                             REDMINE_TICKET_CREATED
                             REDMINE_TICKET_EDITED
                             SONARQUBE_PROJECT_ANALYSIS_COMPLETED

-x, --extended              show extra columns
```



#### 4.2.22 Subscriptions remove help

Informazioni di dettaglio relative alla rimozione di una sottoscrizione.

Remove an existing subscription

##### USAGE

```
$ buttercli sub:remove
```

##### OPTIONS

```
-e, --email=email          (required) user email address

-h, --help                  show CLI help

-j, --json                  display results in json format

-p, --project=project      (required) project name

-t, --event-type=event-type (required) set the event type between
                             GITLAB_COMMIT_CREATED
                             GITLAB_ISSUE_CREATED
                             GITLAB_ISSUE_EDITED
                             GITLAB_MERGE_REQUEST_CREATED
                             GITLAB_MERGE_REQUEST_EDITED
                             GITLAB_MERGE_REQUEST_MERGED
                             GITLAB_MERGE_REQUEST_CLOSED
                             REDMINE_TICKET_CREATED
                             REDMINE_TICKET_EDITED
                             SONARQUBE_PROJECT_ANALYSIS_COMPLETED

-x, --extended              show extra columns
```

### 4.2.23 Subscriptions update help

Informazioni di dettaglio relative alla modifica di una sottoscrizione.

Update an existing subscription

#### USAGE

```
$ buttercli sub:update
```

#### OPTIONS

- `-P, --priority=priority` set the priority  
between LOW, MEDIUM, HIGH
  
- `-c, --contact=contact` choose the contact  
service between EMAIL, SLACK,  
TELEGRAM where the notification  
will be sent
  
- `-e, --email=email` (required) user email address
  
- `-h, --help` show CLI help
  
- `-j, --json` display results in json format
  
- `-k, --keyword=keyword` keyword compared with  
the contents of the future events
  
- `-p, --project=project` (required) project name
  
- `-t, --event-type=event-type` (required) set the event type between  
GITLAB\_COMMIT\_CREATED  
GITLAB\_ISSUE\_CREATED  
GITLAB\_ISSUE\_EDITED  
GITLAB\_MERGE\_REQUEST\_CREATED  
GITLAB\_MERGE\_REQUEST\_EDITED  
GITLAB\_MERGE\_REQUEST\_MERGED  
GITLAB\_MERGE\_REQUEST\_CLOSED  
REDMINE\_TICKET\_CREATED  
REDMINE\_TICKET\_EDITED  
SONARQUBE\_PROJECT\_ANALYSIS\_COMPLETED
  
- `-x, --extended` show extra columns

## 5 Guida all'utilizzo

Lo scopo di questo paragrafo è fornire una guida per l'utilizzo del sistema Butterfly. Il software fa uso di *API REST<sub>G</sub>* per interagire con l'utente: egli ha la possibilità di invocarle direttamente mediante strumenti appositi ed esterni a *Butterfly*<sup>3</sup>, oppure può scegliere di utilizzare *Buttercli*, la command line interface messa a disposizione dal sistema stesso. Per ogni tipo di API segue una tabella riassuntiva cliccabile, che rimanda alla rispettiva spiegazione in dettaglio<sup>4</sup>.

Questa guida presuppone che vi sia un'istanza del sistema in esecuzione con la quale interagire.

Le richieste devono essere indirizzate verso l'endpoint-root:

<http://localhost:5000>

Sono stati individuati i seguenti macro-gruppi che permettono di dividere le API per tipologia:

- **USERS:** API che mettono a disposizione operazioni sugli utenti:

<a href="#">GET /users</a>	Lista di tutti gli utenti registrati	<a href="#">cURL</a>	<a href="#">Buttercli</a>
<a href="#">POST /users</a>	Inserimento di un nuovo utente	<a href="#">cURL</a>	<a href="#">Buttercli</a>
<a href="#">PATCH /users/{email}</a>	Modifica informazioni di uno specifico utente	<a href="#">cURL</a>	<a href="#">Buttercli</a>
<a href="#">DELETE /users/{email}</a>	Cancellazione utente	<a href="#">cURL</a>	<a href="#">Buttercli</a>
<a href="#">GET /users/{email}</a>	Ricerca utente specifico	<a href="#">cURL</a>	<a href="#">Buttercli</a>

**Tabella 3:** Tabella USERS API

- **USER-CONTACTS:** API che permettono di effettuare operazioni sui sistemi di contatto associati ad un utente:

---

<sup>3</sup>Gli esempi qui riportati sono realizzati mediante l'utilizzo del software cURL. Questo software è pre-installato sia nella totalità delle distribuzioni GNU/Linux sia nella Windows Git Bash: si assume pertanto che l'utente ne possieda una conoscenza almeno basilare

<sup>4</sup>Nell'illustrazione dei vari *request bodies* utilizziamo asterischi per indicare campi non obbligatori

GET /user-contacts/ {userEmail}	Lista di tutte le informazioni di contatto associate all'utente ricercato	cURL	Buttercli
POST /user-contacts/ {contactService}	Creazione di una nuova associazione tra un utente esistente ed una piattaforma di contatto	cURL	Buttercli
PUT /user-contacts/ {userEmail}/ {contactService}	Modifica delle informazioni di contatto relative ad uno specifico utente	cURL	Buttercli
DELETE /user-contacts/ {userEmail}/ {contactService}	Cancellazione di informazioni di contatto relative ad uno specifico utente	cURL	Buttercli

**Tabella 4:** Tabella USER-CONTACTS API

- **PROJECTS:** API che consentono di effettuare operazioni sui progetti:

GET /projects	Lista di tutti i progetti registrati	cURL	Buttercli
POST /projects	Creazione di un nuovo progetto	cURL	Buttercli
GET /projects/ {projectName}	Ricerca di un progetto per nome	cURL	Buttercli
PUT /projects/ {projectName}	Modifica delle informazioni di un progetto ricercato per nome	cURL	Buttercli
DELETE /projects/ {projectName}	Rimozione di un progetto ricercato per nome	cURL	Buttercli
DELETE /projects/ {projectName}/ {producerService}	Cancellazione dell'URL associato ad un progetto	cURL	Buttercli

**Tabella 5:** Tabella PROJECTS API

- **SUBSCRIPTIONS:** API per gestire le sottoscrizioni:

GET /subscriptions/ users/ {userEmail}/ projects/ {projectName}/ event-types/ {eventType}	Informazioni sulla sottoscrizione che lega il dato utente, progetto e tipo di eventi	cURL	Buttercli
POST /subscriptions	Creazione di una nuova sottoscrizione	cURL	Buttercli
PATCH /subscriptions/ users/ {userEmail}/ projects/ {projectName}/ event-types/ {eventType}	Modifica delle informazioni relative ad una sottoscrizione	cURL	Buttercli
DELETE /subscriptions/ users/ {userEmail}/ projects/ {projectName}/ event-types/ {eventType}	Cancellazione di una sottoscrizione	cURL	Buttercli

**Tabella 6:** Tabella SUBSCRIPTIONS API

## 5.1 GET /users

Permette di ottenere la lista degli utenti registrati.

- Non richiede alcun parametro;
- Non vi è alcun request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine.

### 5.1.1 cURL

Esempio di utilizzo:

```
curl -X GET http://localhost:5000/users
```

Esempio di risposta:

```
1 {
2   "data": [
3     {
4       "userId": "1",
5       "email": "dstackgroup@gmail.com",
6       "firstname": "DStack",
7       "lastname": "Group",
8       "enabled": true,
9       "created": "2019-05-06T13:32:15.681Z",
10      "modified": null
11    }
12  ]
13 }
```

### 5.1.2 Buttercli

Esempio di utilizzo:

```
buttercli user:list -j
```

Esempio di risposta:

```
1 [
2   {
3     "userId": "1",
4     "email": "dstackgroup@gmail.com",
5     "firstname": "DStack",
6     "lastname": "Group",
7     "enabled": true,
8     "created": "2019-05-06T13:32:15.681Z",
9     "modified": null
10   }
11 ]
```

## 5.2 POST /users

Permette di inserire un nuovo utente.

- Non richiede alcun parametro;
- Il corpo della richiesta è strutturato nel seguente modo:

```
1 {
2     "email": "dstackgroup@gmail.com",
3     "firstname": "DStack",
4     "lastname": "Group",
5     "enabled(*)": true
6 }
```

Il campo non obbligatorio *enabled*, che può risultare non immediatamente comprensibile, è usato per indicare l'eventuale disponibilità dell'utente: di default esso assume valore *true*;

- Il sistema può rispondere con:
  - **Status Code 201**: l'operazione è andata a buon fine;
  - **Status Code 409**: errore nell'inserimento: esiste un altro utente registrato con la stessa email.

### 5.2.1 cURL

Esempio di utilizzo:

```
curl -X POST "http://localhost:5000/users" -H "accept:
application/json" -H "Content-Type: application/json" -d
"{\"email\": \"dstackgroup@gmail.com\", \"firstname\": \"DStack\",
  \"lastname\": \"Group\"}"
```

Esempio di risposta:

```
1 {
2     "data": [
3         {
4             "userId": "1",
5             "email": "dstackgroup@gmail.com",
6             "firstname": "DStack",
7             "lastname": "Group",
8             "enabled": true,
9             "created": "2019-05-06T13:32:15.681Z",
10            "modified": null
11        }
12    ]
13 }
```

### 5.2.2 Buttercli

Esempio di utilizzo:

```
buttercli user:create -j -e dstackgroup@gmail.com -f DStack -l Group
```

Esempio di risposta:

```
1  [  
2    {  
3      "userId": "1",  
4      "email": "dstackgroup@gmail.com",  
5      "firstname": "DStack",  
6      "lastname": "Group",  
7      "enabled": true,  
8      "created": "2019-05-06T13:32:15.681Z",  
9      "modified": null  
10   }  
11  ]
```



### 5.3 GET /users/{email}

Permette di ricercare un utente per mail.

- Come parametro è obbligatoriamente richiesta la stringa che rappresenta l'e-mail dell'utente;
- Non vi è alcun request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: l'utente cercato non esiste.

#### 5.3.1 cURL

Esempio di utilizzo:

```
curl -X GET "http://localhost:5000/users/dstackgroup@gmail.com" -H
"accept: application/json"
```

Esempio di risposta:

```
1 {
2   "data": [
3     {
4       "userId": "1",
5       "email": "dstackgroup@gmail.com",
6       "firstname": "DStack",
7       "lastname": "Group",
8       "enabled": true,
9       "created": "2019-05-06T13:32:15.681Z",
10      "modified": null
11    }
12  ]
13 }
```

#### 5.3.2 Buttercli

Esempio di utilizzo:

```
buttercli user:find -j -e dstackgroup@gmail.com
```

Esempio di risposta:

```
1 [
2   {
3     "userId": "1",
4     "email": "dstackgroup@gmail.com",
5     "firstname": "DStack",
6     "lastname": "Group",
7     "enabled": true,
8     "created": "2019-05-06T13:32:15.681Z",
```

```
9     "modified": null
10   }
11 ]
```

## 5.4 PATCH /users/{email}

Permette di modificare le informazioni relative ad un utente.

- Come parametro è obbligatoriamente richiesta la stringa che rappresenta l'e-mail dell'utente;
- Il corpo della richiesta è strutturato nel seguente modo:

```
1 {
2     "email": "dstackgroup@gmail.com",
3     "firstname(*)": "DStack",
4     "lastname(*)": "Group",
5     "enabled(*)": true
6 }
```

- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: l'utente cercato non esiste.

### 5.4.1 cURL

Esempio di utilizzo:

```
curl -X PATCH "http://localhost:5000/users/dstackgroup@gmail.com" -H
"accept: application/json" -H "Content-Type: application/json" -d
"{\"email\":\"dstackgroup@gmail.com\",\"enabled\":false}"
```

Esempio di risposta:

```
1 {
2     "data": [
3         {
4             "userId": "1",
5             "email": "dstackgroup@gmail.com",
6             "firstname": "DStack",
7             "lastname": "Group",
8             "enabled": true,
9             "created": "2019-05-06T13:32:15.681Z",
10            "modified": null
11        }
12    ]
13 }
```

### 5.4.2 Buttercli

Esempio di utilizzo:

```
buttercli user:update -j -e dstackgroup@gmail.com -f Doge
```

Esempio di risposta:

```
1  [  
2    {  
3      "userId": "1",  
4      "email": "dstackgroup@gmail.com",  
5      "firstname": "DStack",  
6      "lastname": "Group",  
7      "enabled": true,  
8      "created": "2019-05-06T13:32:15.681Z",  
9      "modified": null  
10   }  
11  ]
```

## 5.5 DELETE /users/{email}

Permette di eliminare un utente.

- Come parametro è obbligatoriamente richiesta la stringa che rappresenta l'e-mail dell'utente;
- Non vi è alcun request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: l'utente cercato non esiste.

### 5.5.1 cURL

Esempio di utilizzo:

```
curl -X DELETE "http://localhost:5000/users/dstackgroup@gmail.com" -H  
"accept: application/json"
```

Esempio di risposta:

```
1 {  
2   data: null  
3 }
```

### 5.5.2 Buttercli

Esempio di utilizzo:

```
buttercli user:remove -e dstackgroup@gmail.com
```

## 5.6 GET /user-contacts/{userEmail}

Permette di elencare tutte le piattaforme di contatto associate all'utente ricercato tramite mail:

- Come parametro è obbligatoriamente richiesta l'email dell'utente;
- Non vi è alcun request body;
- Il sistema può rispondere con:
  - **Status Code 404**: si è verificato un errore: l'utente non esiste;
  - **Status Code 200**: l'operazione è andata a buon fine.

### 5.6.1 cURL

Esempio di utilizzo:

```
curl -X GET
"http://localhost:5000/user-contacts/dstackgroup@gmail.com/contacts"
-H "accept: application/json"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "TELEGRAM": {
4       "userContactId": "23",
5       "userId": "1",
6       "contactType": "TELEGRAM",
7       "contactRef": "12739814"
8     },
9     "EMAIL": {
10      "userContactId": "24",
11      "userId": "1",
12      "contactType": "EMAIL",
13      "contactRef": "dstackgroup@gmail.com"
14    }
15  }
16 }
```

### 5.6.2 Buttercli

Esempio di utilizzo:

```
buttercli contact:find -j -e dstackgroup@gmail.com
```

Esempio di risposta:

```
1 {
2   "TELEGRAM": {
3     "userContactId": "23",
4     "userId": "1",
5     "contactType": "TELEGRAM",
```

```
6   "contactRef": "12739814"
7 },
8 "EMAIL": {
9   "userContactId": "24",
10  "userId": "1",
11  "contactType": "EMAIL",
12  "contactRef": "dstackgroup@gmail.com"
13 }
14 }
```

## 5.7 POST /user-contacts/{contactService}

Permette di aggiungere nuove informazioni di contatto ad un utente esistente.

- Come parametro è obbligatoriamente richiesta la stringa che rappresenta il servizio di contatto che si vuole aggiungere. I valori disponibili sono:

- TELEGRAM
- SLACK
- EMAIL

- Il request body è nella forma:

```
1 {  
2   "userEmail": "dstackgroup@gmail.com",  
3   "contactRef": "12739814"  
4 }
```

- Il sistema può rispondere con:
  - **Status Code 404**: si è verificato un errore: l'utente non esiste;
  - **Status Code 201**: l'operazione è andata a buon fine;
  - **Status Code 409**: si è verificato un errore: esiste già l'associazione tra lo specifico utente ed il servizio di contatto;
  - **Status Code 422**: si è verificato un errore: il servizio di contatto non è supportato.

### 5.7.1 cURL

Esempio di utilizzo:

```
curl -X POST "http://localhost:5000/user-contacts/TELEGRAM" -H  
"accept: application/json" -H "Content-Type: application/json" -d  
"{\"userEmail\":\"dstackgroup@gmail.com\",\"contactRef\":\"12739814\"}"
```

Esempio di risposta:

```
1 {  
2   "data": {  
3     "userContactId": "23",  
4     "userId": "1",  
5     "contactService": "TELEGRAM",  
6     "contactRef": "12739814"  
7   }  
8 }
```

### 5.7.2 Buttercli

Esempio di utilizzo:

```
buttercli contact:create -j -e dstackgroup@gmail.com -p SLACK -a 4r446jhf
```



Esempio di risposta:

```
1 {  
2   "userContactId": "23",  
3   "userId": "1",  
4   "contactService": "TELEGRAM",  
5   "contactRef": "12739814"  
6 }
```

## 5.8 PUT /user-contacts/{userEmail}/{contactService}

Permette di modificare le informazioni di contatto di un utente esistente.

- Come parametro sono obbligatoriamente richieste:
  1. La stringa che rappresenta il servizio di contatto che si vuole aggiungere.  
I valori disponibili sono:
    - TELEGRAM
    - SLACK
    - EMAIL
  2. L'email dell'utente di interesse.

- Il request body è nella forma:

```
1 {  
2   "contactRef(*)": "12739814"  
3 }
```

- Il sistema può rispondere con:
  - **Status Code 404:** si è verificato un errore: l'utente o l'associazione data non esiste;
  - **Status Code 200:** l'operazione è andata a buon fine;
  - **Status Code 422:** si è verificato un errore: il servizio di contatto non è supportato.

### 5.8.1 cURL

Esempio di utilizzo:

```
curl -X PUT  
"http://localhost:5000/user-contacts/dstackgroup@gmail.com/TELEGRAM"  
-H "accept: application/json" -H "Content-Type: application/json" -d  
"{\"contactRef\": \"74287343\"}"
```

Esempio di risposta:

```
1 {  
2   "data": {  
3     "userContactId": "23",  
4     "userId": "1",  
5     "contactService": "TELEGRAM",  
6     "contactRef": "12739814"  
7   }  
8 }
```

### 5.8.2 Buttercli

Esempio di utilizzo:

```
buttercli contact:update -j -e dstackgroup@gmail.com -p SLACK -a 32944924
```

Esempio di risposta:

```
1 {  
2   "userContactId": "23",  
3   "userId": "1",  
4   "contactService": "TELEGRAM",  
5   "contactRef": "12739814"  
6 }
```

## 5.9 DELETE /user-contacts/{userEmail}/{contactService}

Permette di rimuovere informazioni di contatto associate all'utente ricercato:

- Come parametro sono obbligatoriamente richieste:
  1. La stringa che rappresenta il servizio di contatto che si vuole aggiungere.  
I valori disponibili sono:
    - TELEGRAM
    - SLACK
    - EMAIL
  2. L'email dell'utente di interesse.
- Non vi è alcun request body;
- Il sistema può rispondere con:
  - **Status Code 404**: si è verificato un errore: l'associazione data non esiste;
  - **Status Code 200**: l'operazione è andata a buon fine.

### 5.9.1 cURL

Esempio di utilizzo:

```
curl -X DELETE
"http://localhost:5000/user-contacts/dstackgroup@gmail.com/TELEGRAM"
-H "accept: application/json"
```

Esempio di risposta:

```
1 {
2   data: null
3 }
```

### 5.9.2 Buttercli

Esempio di utilizzo:

```
buttercli contact:remove -e dstackgroup@gmail.com -p TELEGRAM
```

### 5.10 GET /projects

Permette di ottenere la lista dei progetti registrati:

- Non sono richiesti parametri;
- Non vi sono request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine.

#### 5.10.1 cURL

Esempio di utilizzo:

```
curl -X GET "http://localhost:5000/projects" -H "accept: application/json"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "projectId": "2",
4     "projectName": "Butterfly",
5     "projectURL": {
6       "REDMINE": "http://redmine.dstack.unipd.it",
7       "GITLAB": "http://gitlab.dstack.unipd.it",
8       "SONARQUBE": "http://sonarqube.dstack.unipd.it"
9     }
10  }
11 }
```

#### 5.10.2 Buttercli

Esempio di utilizzo:

```
buttercli project:list -j
```

Esempio di risposta:

```
1 {
2   "projectId": "2",
3   "projectName": "Butterfly",
4   "projectURL": {
5     "REDMINE": "http://redmine.dstack.unipd.it",
6     "GITLAB": "http://gitlab.dstack.unipd.it",
7     "SONARQUBE": "http://sonarqube.dstack.unipd.it"
8   }
9 }
```

### 5.11 POST /projects

Permette di creare un nuovo progetto.

- Non sono richiesti parametri;
- Il request body deve seguire lo schema:

```
1 {
2   "projectName": "Butterfly",
3   "projectURL": {
4     "SONARQUBE": "http://sonarqube.dstack.unipd.it"
5   }
6 }
```

- Il sistema può rispondere con:
  - **Status Code 201**: l'operazione è andata a buon fine;
  - **Status Code 409**: si è verificato un errore: esiste un altro progetto con lo stesso nome.

#### 5.11.1 cURL

Esempio di utilizzo:

```
curl -X POST "http://localhost:5000/projects" -H "accept:
application/json" -H "Content-Type: application/json" -d
"{\"projectName\":\"Butterfly\",\"projectURL\":{\"REDMINE\":\
"http://redmine.dstack.unipd.it\",\"GITLAB\":\
"http://gitlab.dstack.unipd.it\"}}"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "projectId": "2",
4     "projectName": "Butterfly",
5     "projectURL": {
6       "REDMINE": "http://redmine.dstack.unipd.it",
7       "GITLAB": "http://gitlab.dstack.unipd.it",
8       "SONARQUBE": "http://sonarqube.dstack.unipd.it"
9     }
10  }
11 }
```

#### 5.11.2 Buttercli

Esempio di utilizzo:

```
buttercli project:create -j -n Butterfly -g https://gitlab.dstack.unipd.it
```

Esempio di risposta:

```
1 {  
2   "projectId": "2",  
3   "projectName": "Butterfly",  
4   "projectURL": {  
5     "REDMINE": "http://redmine.dstack.unipd.it",  
6     "GITLAB": "http://gitlab.dstack.unipd.it",  
7     "SONARQUBE": "http://sonarqube.dstack.unipd.it"  
8   }  
9 }
```

### 5.12 GET /projects/{projectName}

Permette di cercare un progetto tramite il suo nome.

- Come parametro è richiesta la stringa che rappresenta il nome del progetto;
- Non è richiesto alcun request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: il progetto cercato non esiste.

#### 5.12.1 cURL

Esempio di utilizzo:

```
curl -X GET "http://localhost:5000/projects/Butterfly" -H "accept: application/json"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "projectId": "2",
4     "projectName": "Butterfly",
5     "projectURL": {
6       "REDMINE": "http://redmine.dstack.unipd.it",
7       "GITLAB": "http://gitlab.dstack.unipd.it",
8       "SONARQUBE": "http://sonarqube.dstack.unipd.it"
9     }
10  }
11 }
```

#### 5.12.2 Buttercli

Esempio di utilizzo:

```
buttercli project:find -j -n Butterfly
```

Esempio di risposta:

```
1 {
2   "projectId": "2",
3   "projectName": "Butterfly",
4   "projectURL": {
5     "REDMINE": "http://redmine.dstack.unipd.it",
6     "GITLAB": "http://gitlab.dstack.unipd.it",
7     "SONARQUBE": "http://sonarqube.dstack.unipd.it"
8   }
9 }
```



### 5.12.3 PUT /projects/projectName

Permette di modificare le proprietà di un progetto ricercato per nome.

- Come parametro è richiesta la stringa che rappresenta il nome del progetto;
- Il request body deve seguire lo schema:

```
1 {
2   "projectName": "Butterfly",
3   "projectURL": {
4     "SONARQUBE": "http://sonarqube.dstack.unipd.it"
5   }
6 }
```

- Il sistema può rispondere con:
  - **Status Code 200:** l'operazione è andata a buon fine;
  - **Status Code 404:** si è verificato un errore: il progetto cercato non esiste.

### 5.12.4 cURL

Esempio di utilizzo:

```
curl -X PUT "http://localhost:5000/projects/Butterfly" -H "accept: application/json" -H "Content-Type: application/json" -d '{"projectName": "Butterfly", "projectURL": {"SONARQUBE": "http://sonarqube.dstack.unipd.it"}}'
```

Esempio di risposta:

```
1 {
2   "data": {
3     "projectId": "2",
4     "projectName": "Butterfly",
5     "projectURL": {
6       "REDMINE": "http://redmine.dstack.unipd.it",
7       "GITLAB": "http://gitlab.dstack.unipd.it",
8       "SONARQUBE": "http://sonarqube.dstack.unipd.it"
9     }
10  }
11 }
```

### 5.12.5 Buttercli

Esempio di utilizzo:

```
buttercli project:update -j -n Butterfly -s http://sonarqube.dstack.unipd.it
```

Esempio di risposta:

```
1 {  
2   "projectId": "2",  
3   "projectName": "Butterfly",  
4   "projectURL": {  
5     "REDMINE": "http://redmine.dstack.unipd.it",  
6     "GITLAB": "http://gitlab.dstack.unipd.it",  
7     "SONARQUBE": "http://sonarqube.dstack.unipd.it"  
8   }  
9 }
```

### 5.12.6 DELETE /projects/projectName

Permette di rimuovere un progetto ricercandolo per nome.

- Come parametro è richiesta la stringa che rappresenta il nome del progetto;
- Non è richiesto alcun request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: il progetto cercato non esiste.

### 5.12.7 cURL

Esempio di utilizzo:

```
curl -X DELETE "http://localhost:5000/projects/Butterfly" -H "accept:
application/json"
```

Esempio di risposta:

```
1 {
2   data: null
3 }
```

### 5.12.8 Buttercli

Esempio di utilizzo:

```
buttercli project:remove -n Butterfly -s ALL
```

### 5.12.9 DELETE /projects/projectName/producerService

Permette di rimuovere uno specifico URL associato al progetto di cui si è effettuata la ricerca per nome.

- Come parametro sono richieste:
  1. La stringa che rappresenta il nome del progetto;
  2. La stringa che rappresenta il producer specifico. Può assumere i seguenti valori:
    - SONARQUBE
    - REDMINE
    - GITLAB
- Non è richiesto alcun request body;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 422**: si è verificato un errore: il producer fornito non è valido;
  - **Status Code 404**: si è verificato un errore: il progetto cercato non esiste.

### 5.12.10 cURL

Esempio di utilizzo:

```
curl -X DELETE "http://localhost:5000/projects/Butterfly/REDMINE" -H  
"accept: application/json"
```

Esempio di risposta:

```
1 {  
2   "data": {  
3     "projectId": "2",  
4     "projectName": "Butterfly",  
5     "projectURL": {  
6       "REDMINE": "http://redmine.dstack.unipd.it",  
7       "GITLAB": "http://gitlab.dstack.unipd.it",  
8       "SONARQUBE": "http://sonarqube.dstack.unipd.it"  
9     }  
10  }  
11 }
```

### 5.12.11 Buttercli

Esempio di utilizzo:

```
buttercli project:remove -j -n Butterfly -s GITLAB
```

Esempio di risposta:

```
1 {  
2   "projectId": "2",  
3   "projectName": "Butterfly",  
4   "projectURL": {  
5     "REDMINE": "http://redmine.dstack.unipd.it",  
6     "GITLAB": "http://gitlab.dstack.unipd.it",  
7     "SONARQUBE": "http://sonarqube.dstack.unipd.it"  
8   }  
9 }
```

### 5.13 POST /subscriptions

Permette di creare una nuova sottoscrizione tra un utente, un progetto ed un tipo d'evento.

- Non sono richiesti parametri;
- Il request body deve conformarsi allo schema:

```
1 {
2   "userEmail": "dstackgroup@gmail.com",
3   "projectName": "Butterfly",
4   "eventType": "GITLAB\_COMMIT\_CREATED",
5   "contactServices": [
6     "TELEGRAM",
7     "SLACK"
8   ],
9   "userPriority": "HIGH",
10  "keywords": [
11    "bug",
12    "fix",
13    "performance"
14  ]
15 }
```

- Il sistema può rispondere con:
  - **Status Code 201**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: l'associazione tra l'utente e il sistema di contatto è assente, oppure non esiste alcun progetto con tale nome. o l'evento selezionato non è supportato;
  - **Status Code 409**: si è verificato un errore: la sottoscrizione che si desidera creare esiste già.

#### 5.13.1 cURL

Esempio di utilizzo:

```
curl -X POST "http://localhost:5000/subscriptions" -H "accept: application/json" -H "Content-Type: application/json" -d '{"userEmail": "dstackgroup@gmail.com", "projectName": "Butterfly", "eventType": "GITLAB_COMMIT_CREATED", "contactServices": ["TELEGRAM", "SLACK"], "userPriority": "HIGH", "keywords": ["bug", "fix", "performance"]}'
```

Esempio di risposta:

```
1 {
2   "data": {
3     "subscriptionId": "123",
4     "userEmail": "1",
```

```

5     "projectName": "2",
6     "eventType": "GITLAB\_COMMIT\_CREATED",
7     "contactTypes": {
8         "TELEGRAM": "1232398",
9         "EMAIL": "dstackgroup@gmail.com"
10    },
11    "userPriority": "HIGH",
12    "keywords": [
13        "bug",
14        "fix",
15        "performance"
16    ]
17 }
18 }
```

### 5.13.2 Buttercli

Esempio di utilizzo:

```
buttercli sub:create -j -e dstackgroup@gmail.com -p Butterfly -P HIGH
-k fix -k bug -c SLACK -c EMAIL -t GITLAB_COMMIT_CREATED
```

Esempio di risposta:

```

1 {
2     "subscriptionId": "123",
3     "userEmail": "1",
4     "projectName": "2",
5     "eventType": "GITLAB\_COMMIT\_CREATED",
6     "contactTypes": {
7         "TELEGRAM": "1232398",
8         "EMAIL": "dstackgroup@gmail.com"
9     },
10    "userPriority": "HIGH",
11    "keywords": [
12        "bug",
13        "fix",
14        "performance"
15    ]
16 }
```

### 5.14 GET /subscriptions/users/{userEmail}/projects/{projectName}/event-types/{eventType}

Permette di visualizzare informazioni relative ad una sottoscrizione.

- Sono richiesti i seguenti parametri:
  1. L'email dell'utente;
  2. Il nome del progetto;
  3. Il tipo di evento, che può assumere i seguenti valori:
    - GITLAB\_COMMIT\_CREATED
    - GITLAB\_ISSUE\_CREATED
    - GITLAB\_ISSUE\_EDITED
    - GITLAB\_MERGE\_REQUEST\_CREATED
    - GITLAB\_MERGE\_REQUEST\_EDITED
    - GITLAB\_MERGE\_REQUEST\_MERGED
    - GITLAB\_MERGE\_REQUEST\_CLOSED
    - REDMINE\_TICKET\_CREATED
    - REDMINE\_TICKET\_EDITED
    - SONARQUBE\_PROJECT\_ANALYSIS\_COMPLETED
- Il request body non è richiesto;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: il dato utente non ha mai effettuato tale sottoscrizione;
  - **Status Code 422**: si è verificato un errore: l'email dell'utente non è valida, o si è verificato un problema con il nome del progetto, oppure il tipo d'evento specificato non è supportato.

#### 5.14.1 cURL

Esempio di utilizzo:

```
curl -X GET
"http://localhost:5000/subscriptions/users/dstackgroup@gmail.com/projects/Butterfly/event-
-H "accept: application/json"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "subscriptionId": "123",
4     "userEmail": "1",
5     "projectName": "2",
6     "eventType": "GITLAB\\_COMMIT\\_CREATED",
7     "contactTypes": {
8       "TELEGRAM": "1232398",
9       "EMAIL": "dstackgroup@gmail.com"
```



```
10     },
11     "userPriority": "HIGH",
12     "keywords": [
13         "bug",
14         "fix",
15         "performance"
16     ]
17 }
18 }
```

#### 5.14.2 Buttercli

Esempio di utilizzo:

```
buttercli sub:find -j -e dstackgroup@gmail.com -p Butterfly -t
GITLAB_COMMIT_CREATED
```

Esempio di risposta:

```
1 {
2     "subscriptionId": "123",
3     "userEmail": "1",
4     "projectName": "2",
5     "eventType": "GITLAB\_COMMIT\_CREATED",
6     "contactTypes": {
7         "TELEGRAM": "1232398",
8         "EMAIL": "dstackgroup@gmail.com"
9     },
10    "userPriority": "HIGH",
11    "keywords": [
12        "bug",
13        "fix",
14        "performance"
15    ]
16 }
```

### 5.15 PATCH /subscriptions/users/{userEmail}/projects/{projectName}/event-types/{eventType}

Permette di modificare informazioni relative ad una sottoscrizione.

- Sono richiesti i seguenti parametri:
  1. L'email dell'utente;
  2. Il nome del progetto;
  3. Il tipo di evento, che può assumere i seguenti valori:
    - GITLAB\_COMMIT\_CREATED
    - GITLAB\_ISSUE\_CREATED
    - GITLAB\_ISSUE\_EDITED
    - GITLAB\_MERGE\_REQUEST\_CREATED
    - GITLAB\_MERGE\_REQUEST\_EDITED
    - GITLAB\_MERGE\_REQUEST\_MERGED
    - GITLAB\_MERGE\_REQUEST\_CLOSED
    - REDMINE\_TICKET\_CREATED
    - REDMINE\_TICKET\_EDITED
    - SONARQUBE\_PROJECT\_ANALYSIS\_COMPLETED
- Il request body deve conformarsi al seguente schema:

```
1 {
2   "contactTypes(*)": [
3     "EMAIL",
4     "SLACK"
5   ],
6   "userPriority(*)": "HIGH",
7   "keywords(*)": [
8     "bug",
9     "fix",
10    "performance"
11  ]
12 }
```

- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: il dato utente non ha mai effettuato tale sottoscrizione;
  - **Status Code 422**: si è verificato un errore: l'email dell'utente non è valida, o si è verificato un problema con il nome del progetto, oppure il tipo d'evento specificato non è supportato.

#### 5.15.1 cURL

Esempio di utilizzo:

```
curl -X PATCH
"http://localhost:5000/subscriptions/users/dstackgroup@gmail.com/projects/Butterfly/event
-H "accept: application/json" -H "Content-Type: application/json" -d
"{\"contactTypes\": [\"EMAIL\", \"SLACK\"], \"userPriority\": \"HIGH\", \"keywords\": [\"bug
\", \"fix\", \"performance\"]}"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "subscriptionId": "123",
4     "userEmail": "1",
5     "projectName": "2",
6     "eventType": "GITLAB\_COMMIT\_CREATED",
7     "contactTypes": {
8       "TELEGRAM": "1232398",
9       "EMAIL": "dstackgroup@gmail.com"
10    },
11    "userPriority": "HIGH",
12    "keywords": [
13      "bug",
14      "fix",
15      "performance"
16    ]
17  }
18 }
```

### 5.15.2 Buttercli

Esempio di utilizzo:

```
buttercli sub:update -j -e dstackgroup@gmail.com -p Butterfly -P
MEDIUM -t GITLAB_ISSUE_CREATED -k performance
```

Esempio di risposta:

```
1 {
2   "subscriptionId": "123",
3   "userEmail": "1",
4   "projectName": "2",
5   "eventType": "GITLAB\_COMMIT\_CREATED",
6   "contactTypes": {
7     "TELEGRAM": "1232398",
8     "EMAIL": "dstackgroup@gmail.com"
9   },
10  "userPriority": "HIGH",
11  "keywords": [
12    "bug",
13    "fix",
14    "performance"
15  ]
16 }
```

### 5.16 DELETE /subscriptions/users/{userEmail}/projects/{projectName}/event-types/{eventType}

Permette di rimuovere una sottoscrizione.

- Sono richiesti i seguenti parametri:
  1. L'email dell'utente;
  2. Il nome del progetto;
  3. Il tipo di evento, che può assumere i seguenti valori:
    - REDMINE\_TICKET\_CREATED
    - REDMINE\_TICKET\_EDITED
    - GITLAB\_COMMIT\_CREATED
    - GITLAB\_ISSUE\_CREATED
    - GITLAB\_ISSUE\_EDITED
    - GITLAB\_MERGE\_REQUEST\_CREATED
    - GITLAB\_MERGE\_REQUEST\_EDITED
    - GITLAB\_MERGE\_REQUEST\_MERGED
    - GITLAB\_MERGE\_REQUEST\_CLOSED
    - SONARQUBE\_PROJECT\_ANALYSIS\_COMPLETED
- Il request body non è richiesto;
- Il sistema può rispondere con:
  - **Status Code 200**: l'operazione è andata a buon fine;
  - **Status Code 404**: si è verificato un errore: la sottoscrizione non è stata trovata;
  - **Status Code 422**: si è verificato un errore: l'email dell'utente non è valida, o si è verificato un problema con il nome del progetto, oppure il tipo d'evento specificato non è supportato.

#### 5.16.1 cURL

Esempio di utilizzo:

```
curl -X DELETE
"http://localhost:5000/subscriptions/users/dstackgroup@gmail.com/projects/Butterfly/event-
types/GITLAB_MERGE_REQUEST_CREATED"
-H "accept: application/json"
```

Esempio di risposta:

```
1 {
2   "data": {
3     "subscriptionId": "123",
4     "userEmail": "1",
5     "projectName": "2",
6     "eventType": "GITLAB_MERGE_REQUEST_CREATED",
7     "contactTypes": {
8       "TELEGRAM": "1232398",
9       "EMAIL": "dstackgroup@gmail.com"
```

```
10     },
11     "userPriority": "HIGH",
12     "keywords": [
13         "bug",
14         "fix",
15         "performance"
16     ]
17 }
18 }
```

### 5.16.2 Buttercli

Esempio di utilizzo:

```
buttercli sub:remove -e dstackgroup -p Butterfly -t
GITLAB_COMMIT_CREATED
```

## 6 Esempi notifiche

Seguono gli esempi delle notifiche inviate dal sistema *Butterfly*.

### 6.1 Esempio di notifica Telegram

A screenshot of a Telegram notification. It starts with 'Hi Alberto Schiabel, here's a new event for you.' followed by 'Progetto: Butterfly' and a bolded line 'Some bug written in VSCode by a Doge'. Below that is 'Breaking doge that affects the Middleware Dispatcher.' and a timestamp '17:56' in the bottom right corner.

Hi Alberto Schiabel, here's a new event for you.

Progetto: Butterfly

**Some bug written in VSCode by a Doge**

Breaking doge that affects the Middleware Dispatcher.

17:56

Figura 1: Esempio di notifica Telegram

### 6.2 Esempio di notifica Slack

A screenshot of a Slack notification from a bot named 'butterflystackbot'. It includes a small dog icon, the bot name, 'APP', and the time '9:01 PM'. The message text is: 'Hi Alberto Schiabel, here's a new event for you.', 'Progetto: [Butterfly](https://localhost:10443/dstack/butterfly)', 'AAAAAAAAAAAA DOGE', and 'Breaking issue that affects the Middleware Dispatcher.'.

 **butterflystackbot** APP 9:01 PM

Hi Alberto Schiabel, here's a new event for you.

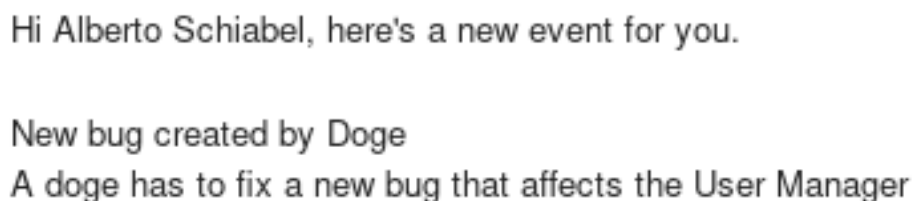
Progetto: [Butterfly](https://localhost:10443/dstack/butterfly)

**AAAAAAAAAAAA DOGE**

Breaking issue that affects the Middleware Dispatcher.

Figura 2: Esempio di notifica Slack

### 6.3 Esempio di notifica Email

A screenshot of an email notification. The text reads: 'Hi Alberto Schiabel, here's a new event for you.', 'New bug created by Doge', and 'A doge has to fix a new bug that affects the User Manager'.

Hi Alberto Schiabel, here's a new event for you.

New bug created by Doge

A doge has to fix a new bug that affects the User Manager

Figura 3: esempio di notifica Email

## A Creazione eventi

### A.1 Apertura issue su Redmine

Sono qui elencati i passi da eseguire per poter aprire una *issue* su *Redmine*:

1. Spostarsi nel *repository* del progetto per il quale si desidera aprire una issue;
2. Cliccare la voce *New Issue*;

3. Compilare i campi richiesti con le informazioni opportune;
4. Confermare la creazione cliccando sul pulsante *Create*.

## A.2 Aggiornare una issue su Redmine

Per poter aggiornare una *issue* su *Redmine* l'utente deve:

1. Aprire la pagina dedicata alle issues;
2. Selezionare la issue da modificare;
3. Cliccare sul pulsante *Update* situato in sopra o sotto la pagina della issue;
4. Editare i valori interessati;
5. Confermare la modifica;

## A.3 Apertura issue su Gitlab

Sono qui elencati i passi da eseguire per poter aprire una *issue* su *GitLab*:

1. Spostarsi nel *repository* del progetto per il quale si desidera aprire una issue;
2. Cliccare sulla voce *Issues*, presente nel menù a sinistra;
3. Cliccare sul bottone riportante la voce *New issue*;
4. Compilare i campi richiesti con le informazioni opportune;
5. Cliccare sul pulsante *Submit issue* per confermare l'inserimento della issue.

## A.4 Modifica issue su Gitlab

Segue l'elenco dei passi da completare per modificare con successo una issue su *GitLab*:

1. Entrare nella pagina relativa alla issue che si desidera modificare;
2. Sono possibili diversi tipi di modifiche:
  - Per modificare *titolo* o *descrizione*, cliccare sul pulsante con il simbolo della matita. Successivamente, salvare le modifiche con l'apposito pulsante.
  - Per modificare altre opzioni (*assegnatari*, *milestones*, ecc...), cliccare sulla dicitura *edit*. La modifica sarà in effetto fin da subito, non occorre premere alcun pulsante di conferma.

## A.5 Invio commit su Gitlab

Oltre alla classica interfaccia da riga di comando, è possibile effettuare commit anche dalla pagina web apposita. Sarà poi sufficiente completare il form con il messaggio di commit ed il branch d'interesse, e confermare premendo il bottone.

## A.6 Creazione analisi su Sonarqube

## A.7 Avvio analisi con Sonarqube

Uno degli eventi che catturati da Butterfly è l'esecuzione dell'analisi statica del codice caricato in una repository e sincronizzato con sonarqube.

Per avviare un'analisi attraverso sonarqube basta eseguire da terminale il seguente comando

```
mvn sonar:sonar.
```

L'output dell'analisi sarà visibile nel server locale.

Per mostrare online l'output è necessario inserire ulteriori flag dopo il comando appena riportato:

- `-Dsonar.projectKey=<project key>`
- `-Dsonar.organization=<my org>`
- `-Dsonar.host.url=https://sonarcloud.io`
- `-Dsonar.login=<token>`

## B Risoluzione degli errori

Riportiamo di seguito una lista di errori comuni, corredati dalla loro soluzione:

- **ERRORE:**

```
"Driver failed programming external connectivity on endpoint"
```

**SOLUZIONE:**

Indica che il precedente processo Docker è stato interrotto da cause esterne a Butterfly. Per risolverlo è necessario:

1. Chiudere tutti i servizi Docker in esecuzione con il comando:

```
docker stop $(docker ps -a -q)
```

2. Riprovare a lanciare il comando appena fallito.

- **ERRORE:**

```
"Windows named pipe error: The system cannot find the file  
specified. (code: 2)"
```

**SOLUZIONE:**

È un errore comune, soprattutto su Windows, e sta ad indicare che il servizio Docker non è in esecuzione. Per risolverlo basta avviare il suddetto servizio;



## C Segnalazioni agli sviluppatori

Nonostante tutto l'impegno profuso nella realizzazione del sistema, esiste la possibilità che al suo interno permangano imperfezioni e/o bachi accidentalmente trascurati durante lo sviluppo. È anche possibile che l'utente di Butterfly desideri che vengano apportate modifiche, estensioni ed ampliamenti al software stesso. In entrambi i casi è possibile contattare gli sviluppatori:

1. Aprendo una *issue* su *GitHub* nella repository del progetto:

<https://github.com/dstack-group/Butterfly/issues>

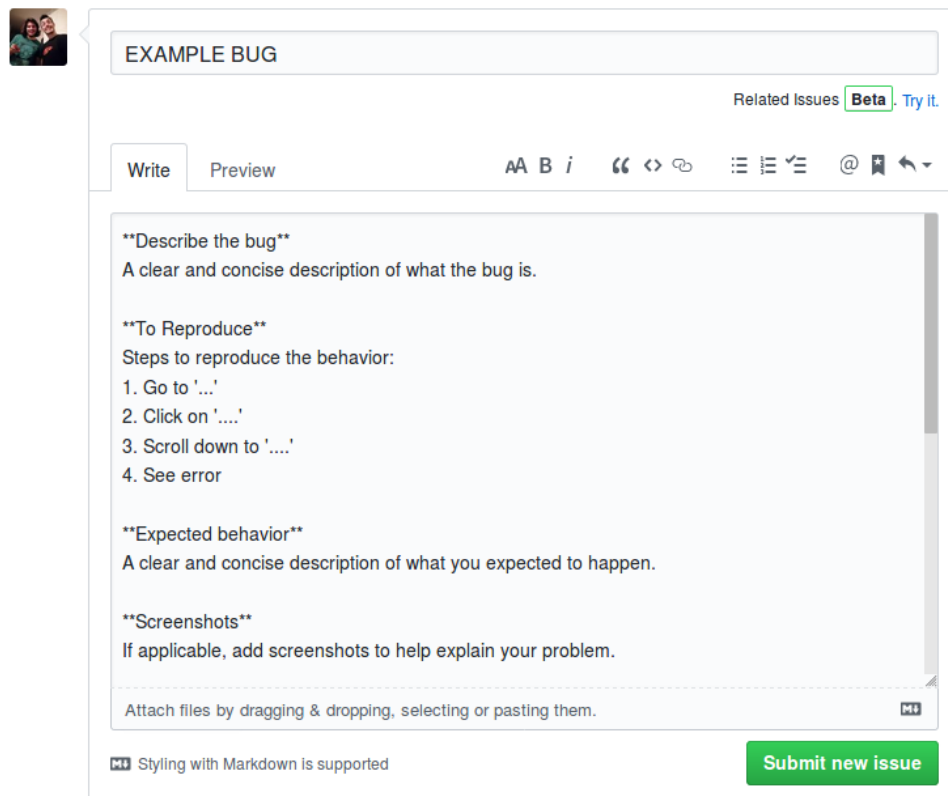
La issue deve essere esplicativa e diretta, in modo tale da permettere agli sviluppatori di capire immediatamente il problema. Essa deve seguire il template messo a disposizione, il quale prevede una iniziale scelta del tipo di segnalazione:

- *BUG CODICE*
- *BUG DOCUMENTAZIONE*
- *RICHIESTA AGGIUNTA FUNZIONALITÀ*

Successivamente è richiesta una opportuna compilazione dei campi necessari: per un esempio si veda la figura 4

### Issue: Bug report

Create a report to help us improve and correct a bug in the code. If this doesn't look right, [choose a different type](#).



The screenshot shows the GitHub 'New issue' form for a bug report. At the top, there's a header 'Issue: Bug report' and a sub-header 'Create a report to help us improve and correct a bug in the code. If this doesn't look right, [choose a different type](#).' Below this is a profile picture of a person and a text input field containing 'EXAMPLE BUG'. To the right of the input field, it says 'Related Issues **Beta** Try it.' Below the input field are two tabs: 'Write' (selected) and 'Preview'. To the right of the tabs is a rich text editor toolbar with icons for bold, italic, link, list, quote, and other formatting options. The main text area contains a template for a bug report with the following sections:   
- '\*\*Describe the bug\*\*': A clear and concise description of what the bug is.   
- '\*\*To Reproduce\*\*': Steps to reproduce the behavior:   
1. Go to '...'   
2. Click on '...'   
3. Scroll down to '...'   
4. See error   
- '\*\*Expected behavior\*\*': A clear and concise description of what you expected to happen.   
- '\*\*Screenshots\*\*': If applicable, add screenshots to help explain your problem.   
At the bottom of the text area, there's a dashed line and the text 'Attach files by dragging & dropping, selecting or pasting them.' with a small icon. Below the text area, there's a small icon and the text 'Styling with Markdown is supported'. At the bottom right, there's a green button labeled 'Submit new issue'.

Figura 4: Esempio di issue templatizzata

2. Scrivendo all'indirizzo email:

`dstackgroup@gmail.com`

## D Definizioni utili

Di seguito viene riportata una lista di termini che possono agevolare l'utente nel utilizzo del sistema:

- **API-REST**: acronimo di **R**epresentational **S**tate **T**ransfer, fornisce un protocollo di comunicazione stateless che si basa sul concetto di locazione delle risorse a cui è possibile accedere tramite un identificatore globale unico (URI), utilizzando le richieste HTTP GET, POST, PUT, DELETE come verbi che indicano rispettivamente richieste di ottenimento/lettura di una risorsa, inserimento/scrittura di una risorsa, modifica/aggiornamento di una risorsa e cancellazione della risorsa;
- **CRUD**: sono quattro operazioni di base che si possono effettuare su dati persistenti. Esse consistono in: create (inserimento nuovi dati), read (lettura dei dati), update (aggiornamento dei dati), delete (eliminazione dei dati);
- **Docker**: è un progetto *open source* che automatizza la consegna o rilascio al cliente di applicazioni all'interno di contenitori software, fornendo un'astrazione mediante la virtualizzazione a livello di sistema operativo;
- **Docker Compose**: è un sistema che permette di definire una lista di servizi definiti da immagini Docker da istanziare in una macchina host. Esso permette di definire variabili d'ambiente da passare ai container, specificare le porte da esporre, le eventuali network virtuali e altre configurazioni;
- **GitLab**: è pari a *GitHub* come scopo e funzioni con l'aggiunta di alcune migliorie. Alcune di esse riguardano il livello di autenticazione, l'interfaccia utente, la condivisione da parte degli utenti di singole piccole parti del progetto, *milestone* a livello di gruppo, rami più protetti, la possibilità di allegare file, ecc;
- **Publisher/Subscriber**: design pattern che permette di implementare un sistema di messaggistica molti a molti tra un numero arbitrario di oggetti. Il pattern prevede la presenza di mittenti e destinatari di messaggi che dialogano tra loro attraverso un tramite, dispatcher o broker. Il funzionamento del pattern è il seguente:
  1. Il mittente di un messaggio (publisher) si occupa esclusivamente di pubblicare (publish) il proprio messaggio al dispatcher, non essendo consapevole di quale è l'identità dei destinatari (subscriber).
  2. I destinatari (subscriber) si rivolgono, a loro volta, al dispatcher abbonandosi (subscribe) alla ricezione dei messaggi.
  3. A questo punto il dispatcher inoltra ogni messaggio inviato da un publisher a tutti i subscriber interessati a quel messaggio.

Il meccanismo di sottoscrizione consente ai subscriber di precisare a quali messaggi sono interessati. Tale pattern implica che ai publisher non sia noto quanti

e quali subscriber ci sono e viceversa, contribuendo in questo modo ad ottenere un sistema scalabile;

- **Redmine:** applicazione web che permette la gestione di progetti;
- **Slack:** è uno strumento di collaborazione aziendale che permette l'invio dei messaggi in modo istantaneo ai membri di un team. Da la possibilità di organizzare la comunicazione del team su canali specifici, che possono essere accessibili od a tutto il team o solo da qualche membro;
- **SonarQube:** strumento che permette di inseguire la qualità del codice attraverso revisioni automatiche con analisi statiche di quest'ultimo;
- **Telegram:** è un servizio di messaggistica istantanea basato su *cloud* ed erogato senza fini di lucro dalla società Telegram LLC, che permette lo scambio di messaggi multimediali e testo.