QB SOFTWARE

×

UNIVERSITÀ DEGLI STUDI DI PADOVA

CORSO DI INGEGNERIA DEL SOFTWARE

ANNO ACCADEMICO 2023/2024

Manuale Utente

Contatti: qbsoftware.swe@gmail.com

Registro delle modifiche

V.	Data	Membro	Ruolo	Descrizione
1.0.0	05/05/2024	A. Domuta	Responsabile	Approvazione documento
0.7.0	26/04/2024	A. Bustreo	Verificatore	Controllo qualità
	26/04/2024	S. Rovea	Programmatore	Effettuata revisione
0.6.0	24/04/2024	S. Rovea	Verificatore	Controllo qualità
	23/04/2024	A. Bustreo	Programmatore	Aggiornato la sezione relativa agli stress test (§8)
0.5.0	23/04/2024	R. Fontana	Verificatore	Controllo qualità
	23/04/2024	S. Rovea	Programmatore	Redatto la sezione relativa a Postman (§7)
0.4.0	12/04/2024	A. Domuta	Verificatore	Controllo qualità
	12/04/2024	A. Bustreo	Programmatore	Redatto una prima versione della sezione di stress test (§8)
0.3.0	11/04/2024	A. Giurisato	Verificatore	Controllo qualità
	10/04/2024	A. Feltrin	Programmatore	Redatto la sezione di supporto (§9)
0.2.0	05/04/2024	S. Rovea	Verificatore	Controllo qualità
	04/04/2024	A. Bustreo	Programmatore	Redatto le sezioni di avvio e installazione: (§4), (§5), (§6) e (§3)
0.1.0	21/03/2024	A. Feltrin	Verificatore	Controllo qualità

V.	Data	Membro	Ruolo	Descrizione
	20/03/2024	A. Domuta	Programmatore	Redatto la sezione di introduzione (§1) e requisiti (§2)

Indice

1 Introduzione **6**
1.1 Scopo del documento . . . 6
1.2 A chi è rivolto . . . 6
1.3 Scopo del prodotto . . . 6
1.4 Glossario . . . 6
1.5 Riferimenti . . . 7
1.5.1 Normativi . . . 7
1.5.2 Informativi . . . 7

2 Requisiti per l'installazione **8**
2.1 Attraverso Docker . . . 8
2.1.1 Requisiti di sistema . . . 8
2.2 Manualmente . . . 8
2.2.1 Requisiti tecnici . . . 8

3 Installazione, configurazione e avvio del prodotto **10**
3.1 Installazione attraverso Docker (consigliata) . . . 10
3.1.1 Configurare i parametri d'ambiente con Docker . . . 10
3.2 Installazione manuale del server . . . 11
3.2.1 Recupero del sorgente e compilazione . . . 11
3.2.2 Variabili d'ambiente del sistema . . . 12

4 Avvio del server (installazione con Docker) **13**

5 Avvio del server (installazione manuale) **13**

6 Provare il server con un client **14**
6.1 Esempio con client Ltt.rs - Android . . . 14
6.1.1 Avvio client . . . 14
6.2 Configurazione client e collegamento al server . . . 14
6.3 Screenshot che mostrano alcune funzionalità . . . 16

7 Postman **17**
7.1 Introduzione a Postman . . . 17

7.2 Import dei test su Postman 17
7.3 Generare i dati e avviare i test 18

8 Stress test **22**
8.1 Locust 22
8.2 Descrizione dei test 22
8.2.1 testEmails 22
8.2.2 testIdentities 23
8.3 Esecuzione dei test 23
8.4 Configurazione parametri 23
8.5 Visualizzazione dei risultati 24

9 Supporto tecnico **26**

Elenco delle figure

1 Schermate per la prima configurazione del client e il collegamento al server. 15
2 Alcuni esempio del client in funzione con server JMAP prodotto. 16
3 Per fare l'import dei test bisogna premere il pulsante "import", selezionare i file precedentemente illustrati. 18
3 Schermata di configurazione dei parametri. 24
4 Schermata monitoraggio statistiche. 24
5 Schermata monitoraggio grafici. 25

1 Introduzione

1.1 Scopo del documento

Il gruppo **QB Software**, con il presente documento intende illustrare come utilizzare il *prodotto*$_G$ sviluppato. In particolare vengono riportate le istruzioni su come installare e configurare il *server*$_G$ di posta elettronica e come collegarsi a esso attraverso un *client*$_G$ e-mail con supporto al protocollo *JMAP*$_G$. Fornisce anche una guida all'installazione e all'uso di *Postman*$_G$ che permetterà di verificare la funzionalità del prodotto. In fine viene riporta una guida alla installazione di *Locust*$_G$ per eseguire gli *stress test*$_G$ per verificare l'*efficienza*$_G$ del protocollo.

1.2 A chi è rivolto

Questo manuale è rivolto a tutti coloro che desiderano installare, provare il server di posta elettronica che utilizzi il protocollo JMAP.

1.3 Scopo del prodotto

Lo scopo del prodotto è principalmente esplorativo, l'obiettivo è quello di implementare una server demo di posta elettronica che utilizzi il protocollo JMAP. Inoltre, il prodotto sviluppato deve essere disponibile come container *Docker*$_G$ e deve essere possibile eseguire degli stress test su quest'ultimo. In questo modo sarà possibile per il *committente*$_G$ capire se implementare o meno il protocollo JMAP nel loro attuale sistema *Carbonio*$_G$, che è un sistema di collaborazione on-line orientato all'efficienza dell'organizzazione del team.

1.4 Glossario

Al fine di una maggiore chiarezza dei contenuti redatti in questo documento, viene fornito in allegato il *Glossario v2.0.0*, dove vengono definiti tutti i termini con un significato particolare o di rilievo nell'ambito del progetto. Un termine presente nel *Glossario* viene contrassegnato dal testo formattato in corsivo, seguito dalla lettera "G" a pedice.

1.5 Riferimenti

1.5.1 Normativi

- *Norme di Progetto v2.0.0*;
- Lezione PD2- Regolamento del progetto didattico: https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf

  [Online - PDF; ultima visita 21/12/2023]
- Capitolato d'appalto C8:
  - https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C8.pdf

    [Online - PDF; ultima visita 21/12/2023];
  - https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C8p.pdf

    [Online - PDF; ultima visita 21/12/2023].

1.5.2 Informativi

- *Glossario v2.0.0*;
- *Piano di Qualifica v2.0.0*;

2 Requisiti per l'installazione

Per poter costruire ed eseguire il prodotto è necessario che il *dispositivo*$_G$ utilizzato possieda i *requisiti*$_G$ descritti in questa sezione. Scegliendo il proprio metodo di installazione i requisiti cambiano.

2.1 Attraverso Docker

2.1.1 Requisiti di sistema

Il prodotto rilasciato come container Docker è stato studiato per ridurre al minimo la necessità di installare utility di terze (esempio Maven) o runtime (esempio JVM) per installare ed eseguire il server. Per poter costruire, installare e utilizzare prodotto bisogna avere nel sistema:

- *Docker*$_G$ come software per la gestione container, per approfondire vedere la guida per l'installazione fornita dal sito ufficiale [Online - Doc; ultima visita 20/03/2024];

2.2 Manualmente

È possibile procedere con una installazione manuale del prodotto. In questo caso l'utente deve essere esperto, e deve scegliere in modo autonomo:

- dove posizionare nel sistema l'eseguibile del server (il file *.jar*$_G$);
- dove posizionare nel sistema il database, e configurarlo secondo le proprie esigenze;
- come configurare il server con le proprie impostazioni;
- come eseguire il server.

2.2.1 Requisiti tecnici

Per costruire il prodotto, configurarlo, installarlo e usarlo manualmente bisogna avere installato nel proprio sistema:

- *Docker*$_G$ come software per la gestione container, per approfondire vedere la guida per l'installazione fornita dal sito ufficiale [Online - Doc; ultima visita 20/03/2024];

- *Java JDK 21 LTS* come runtime (JRE) e come kit di sviluppo (JDK);
- *Maven* versione 3.8.7, come sistema di automation build, per installare Maven seguire la guida del sito ufficiale [Online - Doc; ultima visita 20/03/2024];
- una istanza configurata e pronta all'utilizzo del database *MongoDB* versione minima consigliata 7.0, per installare e configurare MongoDB consigliamo di seguire la documentazione disponibile nel sito ufficiale [Online - Doc; ultima visita 20/03/2024].

3 Installazione, configurazione e avvio del prodotto

In questo capitolo vengono riportare le procedure per l'installazione del prodotto, se necessario la sua configurazione. e l'avvio del server.

3.1 Installazione attraverso Docker (consigliata)

La procedura di installazione del prodotto attraverso Docker è quella che consigliamo, in quanto permette di avere il server pronto da eseguire con il minimo sforzo possibile. Nel caso si voglia avere più libertà su come deve essere installato e configurato consigliamo di procedere con l'installazione manuale, vedi la sezione 3.2.
Seguire i passi elencati per installare il prodotto:

1. scaricare il sorgente dal pagina GitHub della repository del prodotto [Online - GitHub; ultima visita 04/04/2024];
2. (opzionale) configurare le variabili d'ambiente di *Docker Compose*, vedi la sezione 3.1.1 per maggiori dettagli;
3. dentro alla root del progetto avviare Docker Compose per costruire il prodotto, per fare ciò eseguire il comando:

   ```
   $docker compose up --build
   ```

   Docker inizierà a costruire il server, e a scaricare le dipendenze necessarie. In fine avvierà il prodotto già configurato e pronto all'utilizzo.

3.1.1 Configurare i parametri d'ambiente con Docker

Per cambiare la configurazione del server/database è necessario modificare i parametri dentro al file `compose.yaml` nella root del progetto. Service contente il server:

- *ports*: porte esposte dal server, di default al porta è 9999, per poter configurare un'altra porta inserire come parametro la seguente stringa, sostituendo x con la porta desiderata:

  ```
  9999:x
  ```

- *DB_CONNECTION_STRING*: definisce la stringa di connessione al server MongoDB, per cambiare URL di connessione al database sostituire la stringa del parametro con quella desiderata, maggiori dettagli sul manuale di MongoDB sugli URL di connessione al database [Online - Doc; ultima visita 04/04/2024];

Service che mette a disposizione il database:

- *ports*: porte esposte dal database verso l'esterno, di default al porta è 27017, per poter configurare un'altra porta inserire come parametro la seguente stringa, sostituendo x con la porta desiderata:

  ```
  27017:x
  ```

  ATTENZIONE: la porta esposta dal container non è utilizzata dal server, il quale si collega internamente, ma serve per collegare il database a un software esterno per la gestione/controllo dei database (esempio: *MongoDB Compass*$_G$); *MONGO_INITDB_ROOT_USERNAME*: nome dell'admin per il database MongoDB; *MONGO_INITDB_ROOT_PASSWORD*: password dell'admin per il database.

3.2 Installazione manuale del server

L'installazione manuale è sconsigliata alle persone meno esperte.

3.2.1 Recupero del sorgente e compilazione

1. scaricare il sorgente dal pagina GitHub della repository del prodotto [Online - GitHub; ultima visita 04/04/2024];
2. utilizzare Maven per generare l'eseguibile del server:

   ```
   $mvn clean package
   ```

   Siccome il progetto è abbastanza grande è necessario un po' di tempo per eseguire tutti i test, compilare e creare il pacchetto jar. È possibile utilizzare più core per ridurre il tempo attraverso il parametro `-Tx` dove x è il numero di core che si desidera utilizzare;
3. il pacchetto jar è disponibile nella cartella `target/demo_jmap_server-1.0.0-jar-with-dependencies.jar`, spostare l'eseguibile nel path desiderato.

3.2.2 Variabili d'ambiente del sistema

Prima di poter eseguire il prodotto è necessario definire le variabili d'ambiente:

- *DB_CONNECTION_STRING*: con la stringa di connessione al server MongoDB, maggiori dettagli sul manuale di MongoDB sugli URL di connessione al database [Online - Doc; ultima visita 04/04/2024].

4 Avvio del server (installazione con Docker)

Per avviare il server usare:

```
$docker compose up
```

Avviato il server è possibile collegarsi utilizzando la stringa di connessione:

```
http://{ip_server}:{porta_server_jmap}/.well-known/jmap
```

Di default la porta è `9999`, e se il server è in localhost allora si può usare l'IP `127.0.0.1`.

5 Avvio del server (installazione manuale)

Per avviare il server assicurasi di:

- aver configurato il database MongoDB e che sia raggiungibile dal sistema in cui viene eseguito il server;
- di aver configurato le variabili d'ambiente come indicato nella sezione .

Per avviare il server usare:

```
$java -jar demo_jmap_server-1.0.0-jar-with-dependencies.jar
```

oppure sostituire con il nome dell'eseguibile nel caso in cui sia stato rinominato. Avviato il server è possibile collegarsi utilizzando la stringa di connessione:

```
http://{ip_server}:{porta_server_jmap}/.well-known/jmap
```

6 Provare il server con un client

È possibile utilizzare un server con qualunque client e-mail che supporti gli standard JMAP (RF8620 [Online - IETF; ultima visita 04/04/2024] e RF8621 [Online - IETF; ultima visita 04/04/2024]). Inoltre, dato che si tratta di un server di demo, è necessario che il client supporti anche la connessione con HTTP (non HTTPS), in quanto il server non metti a disposizione HTTPS in quanto non è pensato per andare in produzione. Se si desidera scegliere un client diverso da quello che proporremo nella prossima sezione allora consultare il sito ufficiale di JMAP [Online - Sito; ultima visita 04/04/2024] per verificare quelli compatibili.

6.1 Esempio con client Ltt.rs - Android

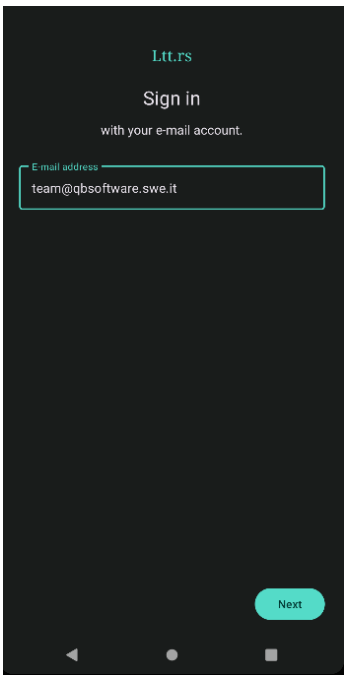
In questa sezione riportiamo un esempio di connessione al server di posta elettronica attraverso il client di posta open source per Android: Ltt.rs (pronunciato "Letters"), è possibile ottenere maggiori informazioni sulla pagina Codeberg del progetto [Online - *Codeberg*$_G$; ultima visita 04/04/2024].

6.1.1 Avvio client

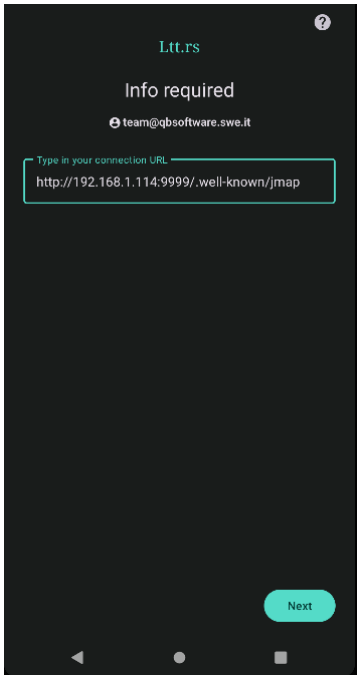
Per iniziare questa sezione si deve aver installato sul proprio dispositivo il client proposto, per l'installazione seguire la presente nel README.md della pagina Codeberg [Online - Codeberg; ultima visita 04/04/2024]. Consigliamo di compilare da sorgente con *Android Studio*$_G$ e di creare l'*apk*$_G$ in modalità debug (non release, in quanto disattiva la possibilità di connettersi a server che usano HTTP). Inoltre modificare l'indirizzo IP nel file `network_security_config.xml` in modo che corrisponda all'indirizzo IP del server.

6.2 Configurazione client e collegamento al server

Appena avviato il client sarà necessario inserire l'email con la quale si intende accedere e successivamente l'indirizzo URL del server come indicato nella sezione 4.



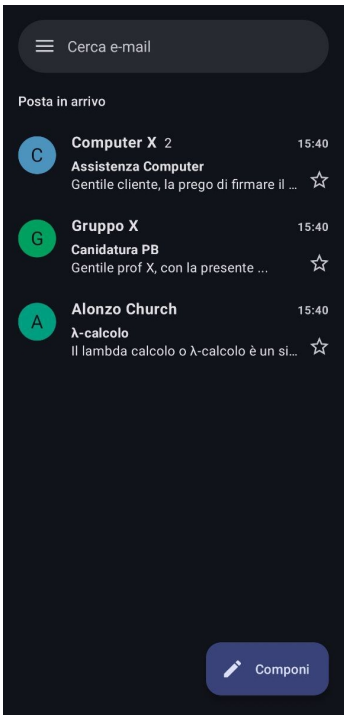
(a) *Esempio dell'inserimento di un indirizzo email che si vuole gestire.*



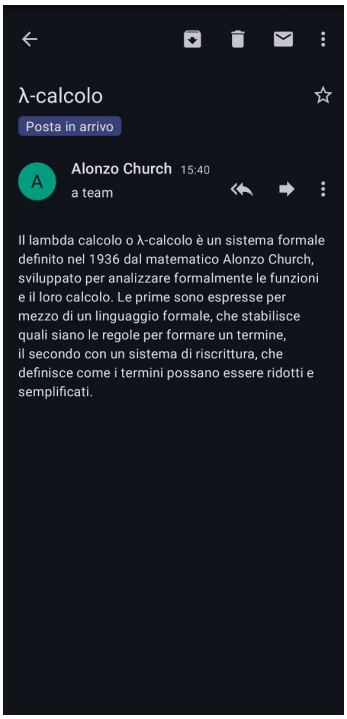
(b) *Esempio dell'inserimento dell'URL del server a cui ci si vuole connettere.*

Figura 1: Schermate per la prima configurazione del client e il collegamento al server.

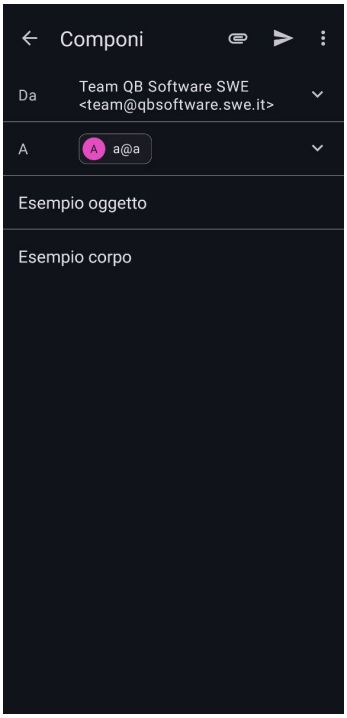
6.3 Screenshot che mostrano alcune funzionalità



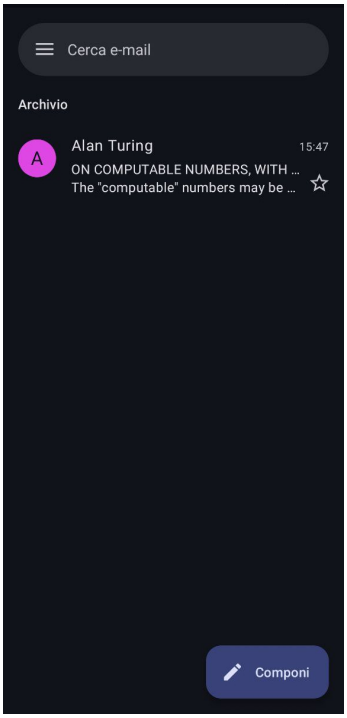
(a) *Esempio della cartella "Posta in arrivo".*



(b) *Esempio della lettura di una email in "Posta in arrivo".*



(c) *Esempio della creazione di una nuova email da inviare.*



(d) *Esempio della cartella "Archivio".*

Figura 2: Alcuni esempio del client in funzione con server JMAP prodotto.

7 Postman

7.1 Introduzione a Postman

Postman è un'applicazione software utilizzata per provare, sviluppare e documentare le API_G. Tra le sue principali caratteristiche, vi sono la capacità di creare e organizzare collezioni di richieste, l'automazione dei test delle API, la gestione delle variabili per facilitare la personalizzazione delle richieste e la generazione di documentazione automatica delle API.

Lo scopo di questa sezione è utilizzare Postman con il server di posta elettronica. Con Postman è possibile provare il funzionamento degli endpoint e funzionalità messe a disposizione del nostro prodotto. Permette anche l'esecuzione dei test e l'interpretazione delle relative risposte. Per l'installazione le istruzioni sono presenti nella pagina di download del sito ufficiale di Postman [Online - Doc; ultima visita 23/04/2024].

7.2 Import dei test su Postman

Scaricando il sorgente nostro prodotto dalla pagina di GitHub è possibile trovare nella path `postman/` partendo dalla root del progetto tutti i file JSON contente i test per Postman. I file sono due:

- *MethodCall JMAP Test.json* contiene tutti i test riguardanti le chiamate JMAP disponibili nel server. Lo scopo di questo pacchetto di test è verificare il corretto funzionamento delle API;
- *Mock Data Gen.json* permette di popolare il server con un insieme di dati da poi utilizzare durante i test delle API.

Attenzione: *MethodCall JMAP Test.json* dipende da *Mock Data Gen.json*, dunque è obbligatorio fare l'import di *Mock Data Gen.json* per poter utilizzare il pacchetto di test per l'API.

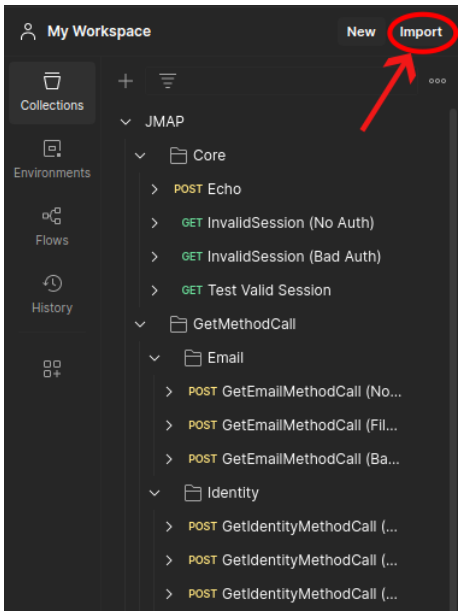
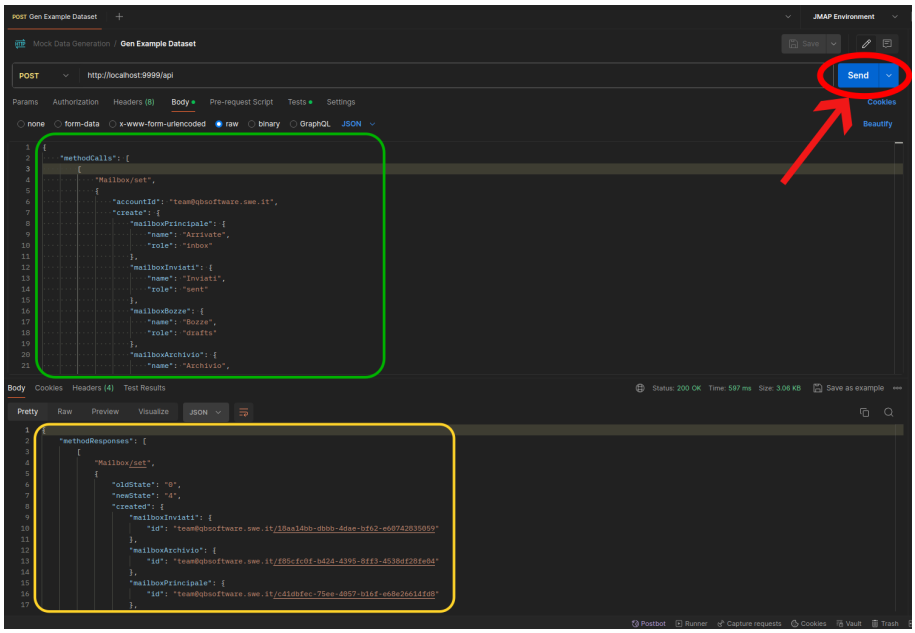


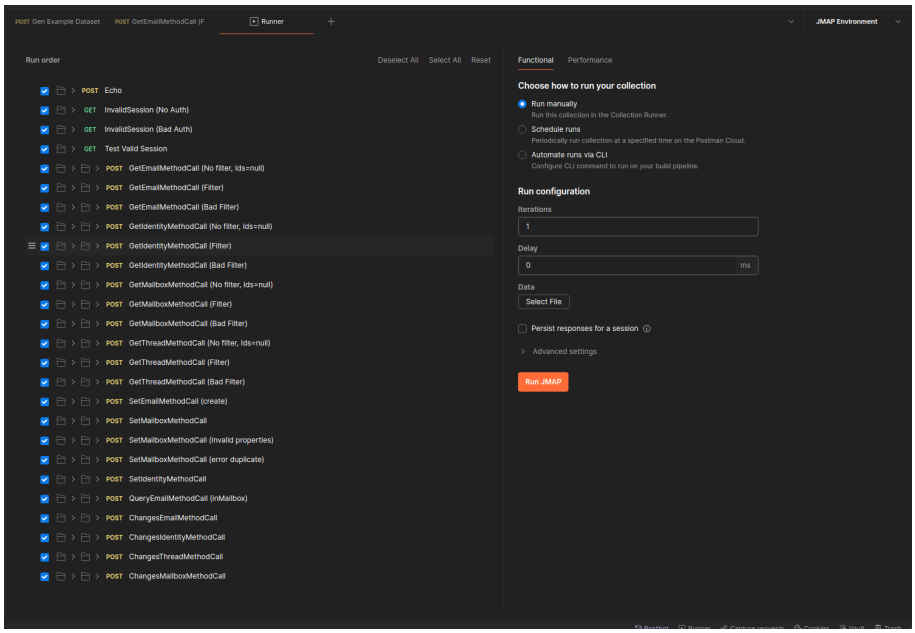
Figura 3: Per fare l'import dei test bisogna premere il pulsante "import", selezionare i file precedentemente illustrati.

7.3 Generare i dati e avviare i test

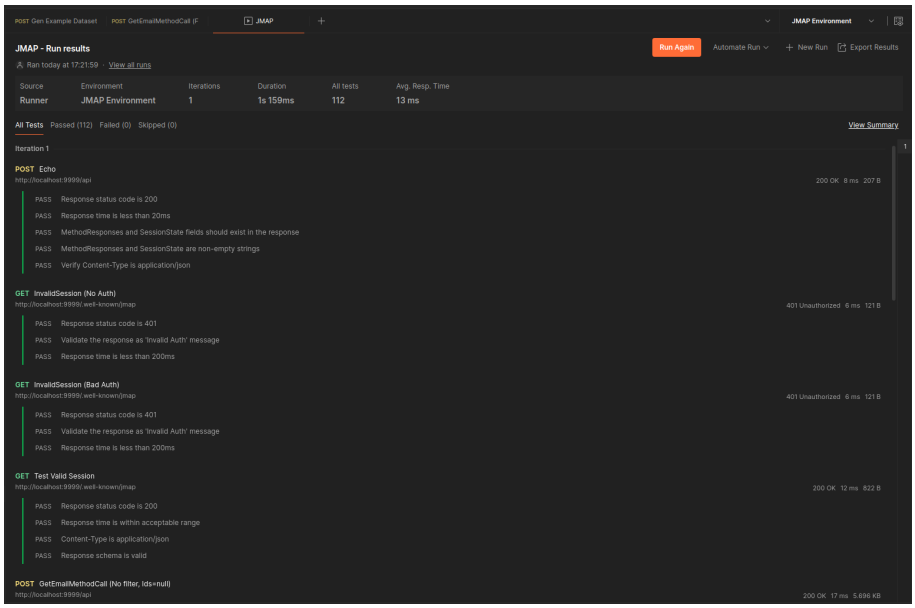
Per generare i dati e avviare i test delle API è necessario avere il server configurato e pronto all'uso, inoltre il database non deve aver nessuna informazione caricata nell'account target che di default è "team@qbsoftware.swe.it". La procedura è illustrata dai seguenti passaggi:



(1) *Avviare la generazione dei dati. Nel "rettangolo" verde vediamo il payload della chiamata JMAP in formato JSON, e nel "rettangolo" giallo vediamo il payload della risposta da parte del server sempre in formato JSON.*



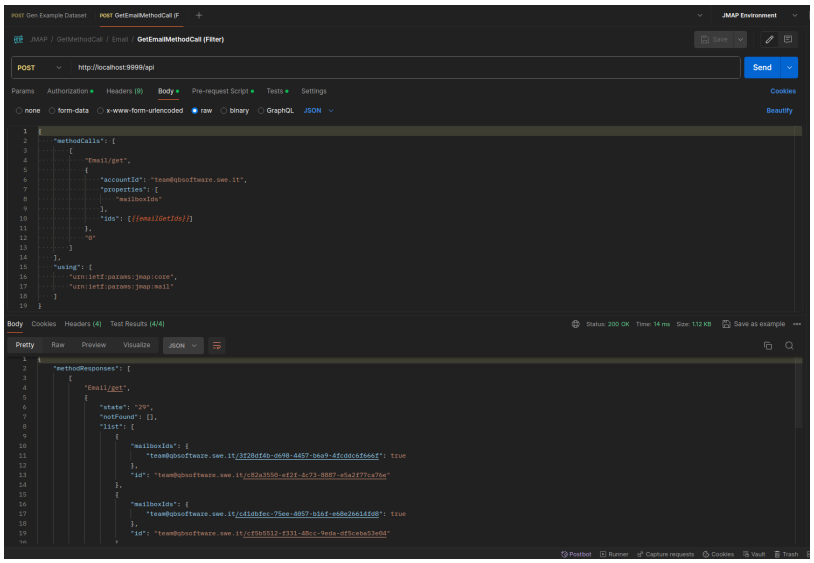
(2) *Andare nella cartella che contiene tutti i test e avviare un "run collection" e poi premere il pulsante in arancione "run JMAP" per avviare i test.*



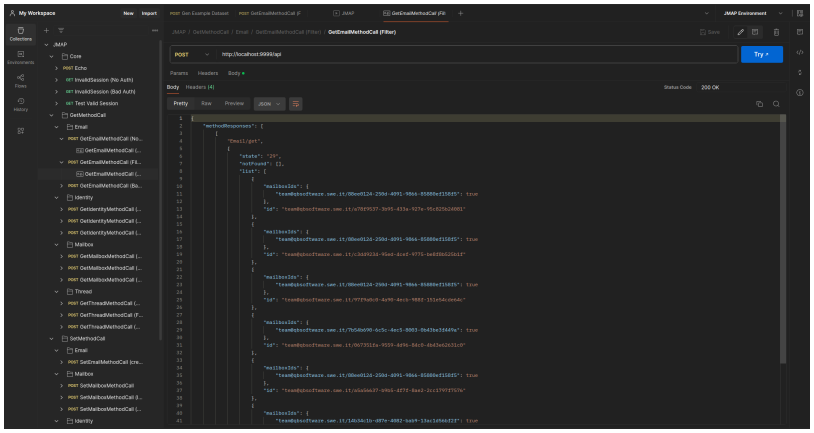
(3) *Finita l'esecuzione di tutti i test verrà generata una schermata con i risultati e i test superati.*

Attenzione: se si desidera ripetere i test è consigliato fare il reset del server e ricreare tutti i dati rifacendo l'intero procedimento, questo è un passaggio obbligato dovuto al fatto che alcuni test eseguono delle operazioni irreversibili (come cancellare una email) e dunque a lungo andare potrebbero finire tutte l'email disponibile e dunque fallisce anche il test correlato.

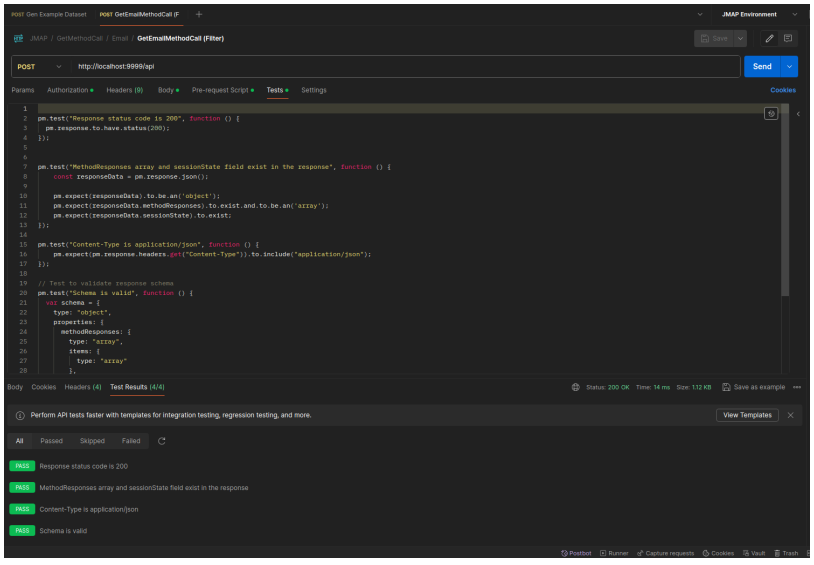
È possibile avviare anche i test manualmente e allegati ad ogni singolo test c'è un esempio di risposta generata dal server.



(d) *Esempio test manuale della chiamata JMAP per ottenere una o più email (Email/Get).*



(e) *Esempio di un data example della Email/Get generata dal server.*



(f) *Esempio dell'esecuzione dei test sulla risposta generata dal server.*

8 Stress test

8.1 Locust

Locust è un framework open-source utilizzato per testare le prestazioni delle applicazioni web. È scritto in Python e consente agli sviluppatori di simulare il comportamento di un gran numero di utenti simultanei che interagiscono con un'applicazione. Utilizzando Locust, è possibile definire scenari di test personalizzati e offre anche strumenti per monitorare e analizzare le prestazioni del sistema durante i test, consentendo agli sviluppatori di identificare e risolvere eventuali problemi. Per installare Locust vedi la guida ufficiale per l'installazione [Online - Sito; ultima visita 23/04/2024].

Lo scopo di questa sezione è utilizzare Locust per permetterci di eseguire degli stress test soddisfacenti e verificare la performance della nostra applicazione, simulando appunto un gran numero di operazioni successive. Le istruzioni per l'installazione sono presenti nella sezione 2.

8.2 Descrizione dei test

8.2.1 testEmails

In questo test di carico, si simula l'invio di un numero elevato di email da parte di un utente. Ecco cosa fa nel dettaglio:

- **Autenticazione**: All'inizio del test, l'utente si autentica sul server di posta;
- **Generazione delle caselle di posta**: Dopo l'autenticazione, il test genera delle caselle di posta sul server;
- **Creazione della casella di posta "sent"**: Controlla se esiste una casella di posta per le email inviate "sent". Se non esiste, la crea;
- **Generazione di email**: Infine, il test simula l'invio di email da molteplici utenti. Ogni email viene generata con un indirizzo email del mittente casuale, scelto casualmente da una pool predefinita.

8.2.2 testIdentities

In questo test di carico si simula la creazione di un numero elevato di identità. Ecco cosa fa nel dettaglio:

- **Autenticazione**: All'inizio del test, l'utente si autentica sul server di posta;
- **Ottenimento dell'ID dell'account**: Dopo l'autenticazione, il test ottiene l'ID dell'account dell'utente;
- **Generazione di identità**: Il test consiste in una task che genera una nuova identità. Questa identità ha un nome casuale scelto da un pool predefinito namePool e un indirizzo email casuale scelto da un altro pool predefinito emailPool.

8.3 Esecuzione dei test

Per eseguire correttamente i test bisogna seguire i seguenti passaggi:

- **Avviare il server di posta**: Prima di eseguire i test, è necessario avviare il server di posta. Per farlo, aprire un terminale e spostarsi nella cartella del progetto. Eseguire il comando: `docker-compose up --build`;
- **Avviare il test**: Aprire un altro terminale e spostarsi nella cartella del progetto. Eseguire il comando:
  `python -m locust -f .\locust\test.py --host=http://localhost:9999`. Bisogna sostituire `test.py` con il nome del test che si vuole eseguire e anche modificare l'host se il server di posta è in esecuzione su un altro indirizzo.
- **Visualizzazione interfaccia utente**: Aprire un browser e andare all'indirizzo `http://localhost:8089`. Qui è possibile visualizzare i risultati dei test in tempo reale utilizzando l'interfaccia utente di locust.

8.4 Configurazione parametri

Tramite l'interfaccia utente di Locust sarà necessario impostare i seguenti parametri:

- **Number of users (peak concurrency)**: Numero di utenti massimo che si vuole simulare durante il test;

- **Ramp up**: Definisce il tasso di aggiunta di nuovi utenti simulati nel tempo. Ad esempio, impostando il ramp up a 3 utenti al secondo e il numero di utenti a 100, Locust inizierà con 3 utenti e ne aggiungerà altri 3 ogni secondo fino al raggiungimento del numero massimo di 100 utenti.

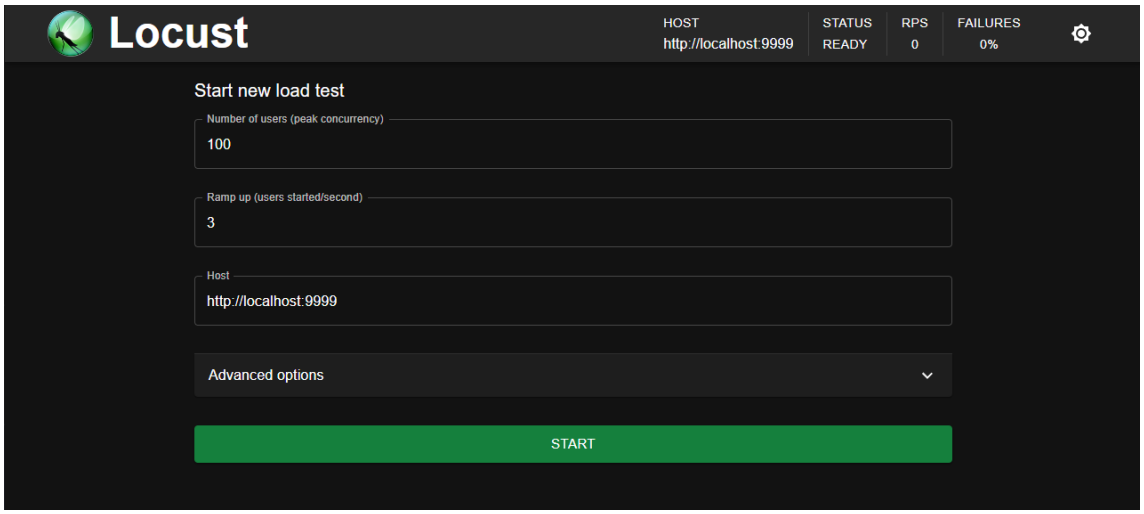


Figura 3: Schermata di configurazione dei parametri.

8.5 Visualizzazione dei risultati

Una volta impostati i parametri, è possibile avviare il test premendo il pulsante "START". Durante l'esecuzione del test, è possibile visualizzare i risultati in tempo reale nell'interfaccia utente di Locust. Una volta terminato il test, è possibile visualizzare i risultati completi e le statistiche dettagliate.

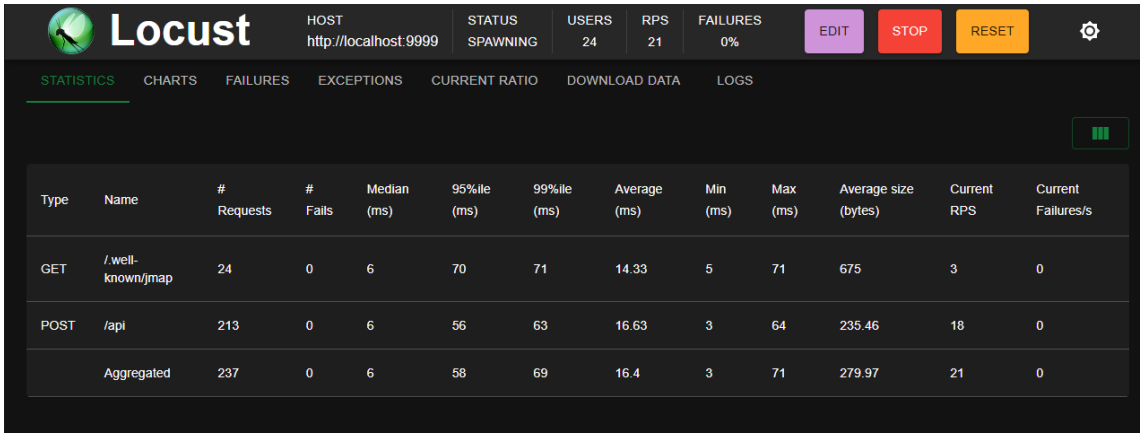


Figura 4: Schermata monitoraggio statistiche.

Figura 5: Schermata monitoraggio grafici.

9 Supporto tecnico

Per qualsiasi dubbio o problema contattare il nostro team all'e-mail qbsoftware.swe@gmail.com assicurandosi di aver fornito una accurata descrizione del problema. In caso questo non sia risolvibile tramite e-mail è possibile programmare un meeting in cui guideremo passo passo l'utente all'avvio della nostra applicazione e alla risoluzione di eventuali problemi.