

Manuale Utente

2024-03-21 — v1.0.0



overture.unipd@gmail.com

Destinatari	Prof. Tullio Vardanega Prof. Riccardo Cardin <i>Zextras</i> Gruppo <i>Overture</i>
Responsabile	Eleonora Amadori
Redattori	Michele Bettin Riccardo Bonavigo Francesco Costantino Bulychov Riccardo Fabbian Francesco Furno Alex Vedovato
Verificatori	Michele Bettin Riccardo Bonavigo Riccardo Fabbian Francesco Furno Alex Vedovato

Registro delle modifiche

Versione	Data	Autori	Verificatori	Dettaglio
1.0.0	2024-03-21	Michele Bettin	Riccardo Bonavigo	Approvazione per PB.
0.1.0	2024-03-17	Francesco Furno	Michele Bettin	Aggiunta la sezione 'Guida all'uso di Postman'.
0.0.5	2024-03-10	Francesco Costantino Bulychov	Francesco Furno	Aggiunta la sezione 'Avvio degli stress test'.
0.0.4	2024-03-07	Michele Bettin	Alex Vedovato	Aggiunta la sezione 'Supporto tecnico'.
0.0.3	2024-02-15	Riccardo Fabbian	Francesco Furno	Aggiunta la sezione 'Collegamento di un client di posta elettronica installato su un dispositivo Android'.
0.0.2	2024-02-14	Riccardo Bonavigo, Alex Vedovato	Michele Bettin	Aggiunta la sezione 'Avvio del server di posta elettronica'.
0.0.1	2024-02-12	Francesco Costantino Bulychov	Riccardo Fabbian	Struttura di base del documento e introduzione.

Indice

1) Introduzione	5
1.1) Scopo del prodotto	5
1.2) Scopo del documento	5
1.3) Glossario	5
1.4) Riferimenti	5
1.4.1) Riferimenti normativi	5
1.4.2) Riferimenti informativi	5
2) Avvio del server di posta elettronica	6
2.1) Requisiti tecnici per avviare il server	6
2.1.1) Requisiti opzionali	6
2.2) Download del repository relativo al codice sorgente del server	6
2.3) Spiegazione dei principali file presenti nei sorgenti	7
2.3.1) File: Compose.yml	7
2.3.2) File: Caddyfile	7
2.3.3) File: build.gradle.kts	7
2.3.4) File: startup.jsh	7
2.3.5) File: .env	7
2.3.6) File: .justfile	7
2.4) Duck DNS	8
2.4.1) Guida a Duck DNS	8
2.4.2) Aggiornamento server per Duck DNS	9
2.5) Istruzioni per l'avvio	9
2.6) Istruzioni per lo spegnimento	9
3) Collegamento di un client di posta elettronica installato su un dispositivo Android	10
3.1) Operazioni preliminari	10
3.2) Installazione, avvio del client e collegamento	10
4) Guida all'uso di Postman	12
4.1) Introduzione	12
4.2) Installazione ed accesso	12
4.3) Importazione delle richieste	12
4.4) Visualizzazione dei dettagli della richiesta	13
4.5) Esecuzione dei test ed interpretazione delle risposte	14
5) Avvio degli stress test	15
5.1) Requisiti preliminari	15
5.2) Descrizione dei test	15
5.2.1) test1 - Accesso all'inbox	15
5.2.2) test2 - Invio di email	15
5.2.3) test3 - Gestione delle cartelle	16
5.3) Esecuzione dei test	16
5.3.1) Configurazione dei parametri	17
5.3.2) Monitoraggio dei risultati	18
6) Supporto tecnico	19

Lista della immagini

Figura 1: Maschera per l’inserimento del nome del sottodominio da associare all’Indirizzo IP privato del server in uso.	8
Figura 2: Maschera per l’inserimento dell’Indirizzo IP privato del server in uso.	8
Figura 3: Ping tra il dispositivo mobile e il server avvenuto con successo.	10
Figura 4: Inserimento dell’email con il quale si vuole effettuare l’accesso.	11
Figura 5: Inserimento della password relativa all’account inserito prima.	11
Figura 6: Schermata principale con la lista di tutte le email ricevute.	11
Figura 7: Importazione del file JSON contenente tutte le richieste in Postman.	12
Figura 8: Schermata di visualizzazione dei dettagli della richiesta.	13
Figura 9: Schermata di esecuzione dei test.	14
Figura 10: Visualizzazione del risultato dei test eseguiti sulla richiesta fornita.	14
Figura 11: Schermata di configurazione dei parametri in Locust.	17
Figura 12: Schermata di monitoraggio dei risultati.	18
Figura 13: Schermata di monitoraggio dei grafici.	18

1) Introduzione

1.1) Scopo del prodotto

Lo scopo principale del prodotto è quello di permettere all'azienda *proponente* di poter valutare se ha senso investire tempo e risorse per implementare il *protocollo JMAP* nel loro prodotto di punta chiamato *Carbonio*, una soluzione di collaborazione online che ruota attorno alla gestione delle email. JMAP è difatti un protocollo di comunicazione appositamente progettato per semplificare l'interazione tra client e server nell'ambito delle applicazioni di posta elettronica.

Attualmente, Carbonio fa affidamento su protocolli standard come IMAP, POP e *Exchange Active Sync*. Di conseguenza, l'implementazione di JMAP potrebbe offrire potenzialmente un aumento di *funzionalità* e *efficienza* a un costo inferiore.

1.2) Scopo del documento

Questo documento ha lo scopo di spiegare ai committenti le modalità di utilizzo e di verifica delle funzionalità del sistema informatico che il gruppo Overture ha dovuto sviluppare per adempiere alle richieste fatte in merito allo studio del protocollo JMAP per la posta elettronica.

Al suo interno verranno illustrate tutte le istruzioni per avviare il server di posta elettronica (quindi il nostro backend), le istruzioni per collegarsi ad esso tramite un client di posta elettronica, una breve guida all'uso di Postman per dimostrare il corretto funzionamento del nostro prodotto e le funzionalità che esso espone ed, infine, come avviare gli stress test richiesti dal committente così da poterli modificare a piacimento, con il fine di testare più approfonditamente o in modo diverso alcune parti.

1.3) Glossario

Per evitare ambiguità o incomprensioni riguardanti la terminologia usata nel documento, è stato deciso di adottare un glossario in cui vengono riportate le varie definizioni. In questa maniera in esso verranno riportati tutti i termini specifici del dominio d'uso con relativi significati.

La presenza di un termine all'interno del Glossario viene indicata applicando *questo stile*.

1.4) Riferimenti

1.4.1) Riferimenti normativi

- Norme di Progetto vw.0.0:
https://overture-unipd.github.io/docs/rtb/interni/norme_di_progetto_v2.0.0.pdf
- PD2 - Regolamento del progetto didattico
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf>
- Capitolato d'appalto C8: JMAP, il nuovo protocollo standard per la comunicazione email
<https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C8.pdf>

1.4.2) Riferimenti informativi

- Glossario v2.0.0:
https://overture-unipd.github.io/docs/rtb/interni/glossario_v2.0.0.pdf

2) Avvio del server di posta elettronica

2.1) Requisiti tecnici per avviare il server

In questa sezione andiamo a definire la lista dei requisiti che un server a livello di software debba avere per poter procedere all'installazione e all'avvio del codice sorgente del server fornito.

Come prima cosa il dispositivo in questione deve aver installato Docker e Docker Compose per la gestione di applicazioni Docker multi-container. Di seguito la guida ufficiale per l'installazione: <https://docs.docker.com/engine/install>.

Successivamente il sistema deve aver installato al suo interno Java in una sua versione uguale o successiva alla 21. Di seguito la guida ufficiale per l'installazione: <https://docs.oracle.com/en/java/javase/21/install/#Java-Platform%2C-Standard-Edition>.

Infine si richiede l'installazione di Gradle che è il sistema di build scelto. Di seguito la guida ufficiale per l'installazione: <https://gradle.org/install/>.

2.1.1) Requisiti opzionali

2.1.1.1) Just e guida per l'installazione

Ai fini di un'avvio del server che richieda il minor sforzo da parte dell'utente è possibile installare il pacchetto Just per evitare di copiare e incollare manualmente i comandi necessari all'avvio presenti nel file .justile come spiegato nella sezione **File: justfile**. Just è uno strumento di automazione leggero e flessibile che semplifica l'esecuzione di comandi e task attraverso l'utilizzo di un file di configurazione chiamato "Justfile".

Per installare questo tool è necessario recarsi sul seguente repository di github <https://github.com/casey/just> e seguire la guida ufficiale in base al sistema operativo della macchina sulla quale si vuole effettuare l'installazione.

Per verificare che Just sia stato installato correttamente, digita il seguente comando nel terminale: `just --version`. Se l'installazione è avvenuta con successo, vedrai la versione di Just attualmente installata.

2.1.1.1.1) Guida per l'utilizzo

- Passo 1: Naviga nella Directory del Progetto: apri il terminale e spostati nella directory del progetto e assicurati di essere nella stessa directory del file ".justfile".
- Passo 2: Esegui i Task: una volta nella directory del progetto, puoi eseguire i task definiti nel ".justfile" con il comando just seguito dal nome del task. Ad esempio, se esiste un task chiamato "build" nel ".justfile", digita: `just build`.
- Passo 3: Visualizza la Lista dei Task: se si vuole visualizzare la lista dei task disponibili nel ".justfile", puoi eseguire il comando just senza argomenti: `just`. Questo mostrerà un elenco di tutti i task definiti nel ".justfile".
- Passo 4: Ulteriori Opzioni: just offre molte opzioni e funzionalità avanzate. Si può esplorare ulteriori opzioni eseguendo il comando `just --help` per visualizzare la documentazione.

2.2) Download del repository relativo al codice sorgente del server

Per scaricare i sorgenti relativi al codice sorgente del server basta scaricare la cartella zip presente nel seguente repository dedicata al seguente link: <https://github.com/overture-unipd/jmap.git>.

Alternativamente, se si ha installato il sistema di versionamento Git sul sistema dove si vuole effettuare il download dei sorgenti è possibile clonare direttamente il repository su una cartella a piacere digitando il seguente comando: `git clone https://github.com/overture-unipd/jmap.git`.

2.3) Spiegazione dei principali file presenti nei sorgenti

2.3.1) File: **Compose.yml**

Il file `Compose.yml` è un file di configurazione utilizzato per definire e gestire i servizi di un'applicazione Docker. Docker Compose è uno strumento che semplifica il processo di definizione, configurazione e orchestrazione di più contenitori Docker come parte di un'applicazione complessa.

Al suo interno troviamo la definizione dei seguenti servizi (quindi l'istanziamento delle seguenti immagini):

- `overture-unipd/jmap:latest`: il nostro server `jmap` sotto forma di container, dunque un vero e proprio web server istanziato con il codice sorgente del repository;
- `rethinkdb:2.4.2-bullseye-slim`: il nostro database `Rethink Db`;
- `overture-unipd/caddy:latest`: il servizio `caddy` necessario qualora un client necessiti di `https` per avviare una comunicazione con il server (feature ricorrente nei client android).

2.3.2) File: **Caddyfile**

Un file dedicato di `Caddy` che definisce le regole per impostare un reverse proxy (nel nostro caso abbiamo scelto `duckdns`) al fine di arginare il problema di sicurezza che alcuni client sollevano se manca `https` nella comunicazione.

2.3.3) File: **build.gradle.kts**

Il file `build.gradle.kts` è un file di configurazione per progetti basati su `Gradle`, un sistema di automazione della compilazione e gestione delle dipendenze ampiamente utilizzato per progetti Java, Kotlin e altri linguaggi. La notazione `.kts` indica che il file è scritto utilizzando il linguaggio di scripting `Kotlin`, invece del tradizionale formato `Groovy` utilizzato nei file `build.gradle`. Questo file contiene istruzioni per la configurazione del progetto, inclusi dettagli come le dipendenze del progetto, i plugin da utilizzare, le impostazioni del compilatore, i task di build e altre configurazioni specifiche del progetto.

2.3.4) File: **startup.jsh**

Il file `startup.jsh` è un file di configurazione contenente tutti i comandi necessari per lanciare la shell di Java.

2.3.5) File: **.env**

Il file `.env` è un file di configurazione utilizzato per immagazzinare variabili d'ambiente in un progetto. Quest'ultime possono contenere informazioni sensibili come chiavi API, password, configurazioni di database o qualsiasi altra informazione che non dovrebbe essere inclusa direttamente nel codice sorgente o nei file di configurazione. Le variabili d'ambiente memorizzate nel file `.env` possono poi essere utilizzate dal programma per accedere a queste informazioni senza doverle hardcodificare nel codice. Questo è particolarmente utile quando si lavora in ambienti diversi come sviluppo locale, test e produzione, in cui le configurazioni possono variare.

In questo file noi abbiamo memorizzato anche gli utenti di default registrati nel server di posta elettronica. Tipicamente è una cosa sbagliata da fare, ma ai fini del prodotto che dobbiamo realizzare (non dobbiamo realizzare un prodotto finito, ma una demo per testare una nuova tecnologia) è accettabile.

2.3.6) File: **justfile**

Qui troviamo tutti i comandi specificati nella sezione successiva di questo documento (**Istruzioni per l'avvio**), necessari per avviare concretamente i sorgenti del server appena installato.

2.4) Duck DNS

Dalla lista di client che ci sono stati forniti da Zextras in un repository pubblico, abbiamo scoperto che alcuni di questi necessitano una comunicazione https per potersi collegare al server (senza https per questioni di sicurezza rifiutano le risposte del server).

Per questo motivo, considerando il contesto in cui dobbiamo lavorare (dove questo grado di sicurezza non ci serve e non abbiamo neanche a disposizione un server in rete con un certificato https) abbiamo deciso di bypassare questo problema utilizzando Duck DNS, il quale ci consente di generare dei certificati https per qualunque sottodominio duckdns.org preferiamo.

Quindi nel nostro caso abbiamo creato il sottodominio `overture1.duckdns.org` e gli abbiamo associato l'indirizzo IP privato del server all'interno della LAN. A questo punto, qualora un client volesse collegarsi al server lo dovrà fare con il sottodominio `overture1.duckdns.org`. Così facendo, esso non darà nessun errore di sicurezza in quanto Duck DNS fornirà un certificato https valido per quel sottodominio che in realtà sarà il nostro server situato nella stessa LAN.

2.4.1) Guida a Duck DNS

Come prima cosa dobbiamo capire quale è l'indirizzo IP privato del server dove vogliamo eseguire l'installazione nella LAN (il comando per capirlo differisce in base al sistema operativo). A questo punto occorre recarsi sul sito di Duck DNS (<https://www.duckdns.org/domains>) e registrarsi.

Dovremmo vedere una maschera come quella in foto:



Figura 1: Maschera per l'inserimento del nome del sottodominio da associare all'Indirizzo IP privato del server in uso.

Quindi inseriamo il nome del sottodominio da associare all'indirizzo IP privato del server in uso (in questo caso abbiamo scelto il nome: `overture1`) e clicchiamo sul bottone verde: `add domain`.

Ora dovremmo vedere la seguente maschera:



Figura 2: Maschera per l'inserimento dell'Indirizzo IP privato del server in uso.

Quindi inseriamo l'Indirizzo IP privato del server in uso (in questo caso 192.168.1.10) e clicchiamo sul bottone arancione: update ip.

2.4.2) Aggiornamento server per Duck DNS

Per far funzionare il server in seguito a questa modifica è necessario recarsi sul file .env e svolgere le seguenti modifiche:

- Eliminare il record DOMAIN e sostituirlo con DOMAIN= (nel nostro caso DOMAIN=overture1.duckdns.org);
- Eliminare il record DUCKDNS e sostituirlo con DUCKDNS= (dove il token lo si trova sul sito Duck DNS subito dopo essersi autenticati).

2.5) Istruzioni per l'avvio

Se sono stati rispettati tutti i requisiti della sezione **Requisiti tecnici per avviare il server** e si è in possesso del codice sorgente del server si potrà procedere con l'avvio.

Come prima cosa assicuriamoci che:

- Docker sia attivo nel computer in uso con il relativo comando (`sudo systemctl status docker` in ambiente Linux e `docker info` in ambiente Windows);
- Java sia stato installato correttamente (quindi ne verifichiamo la versione con il comando `java -version`);
- Gradle sia stato installato correttamente (quindi ne verifichiamo la versione con il comando `gradle -version`).

A questo punto se si desidera avviare il server di posta elettronica si devono digitare manualmente i comandi presenti nel file .justfile (ovviamente questo passaggio può essere evitato qualora si abbia installato Just e di conseguenza si riesca ad avviare automaticamente tutti i comandi dal justfile fornito). I comandi per l'avvio del server quindi sono i seguenti:

- `gradle dockerBuildImage`: per creare l'immagine docker per l'applicazione;
- `docker build -t overture-unipd/caddy:latest -f caddy.dockerfile .`: per creare l'immagine di caddy, in quanto nel file caddy.dockerfile ci sono le istruzioni per creare la nostra immagine personalizzata di caddy che richiede l'integrazione con duckdns per avere il certificato https;
- `docker compose up`: per avviare tutti i container.

Qualora ci fossero degli errori, essi verranno mostrati nel terminale nel quale si hanno digitato i comandi.

Come prova del nove per verificare che il server sia attivo possiamo aprire un browser a scelta e digitare nella barra di ricerca: localhost:9000. Se possiamo vedere il pannello di gestione del database in genere significa che tutto è andato a buon fine.

2.6) Istruzioni per lo spegnimento

Per spegnere il server avviato seguendo le direttive specificate nella sezione **Istruzioni per l'avvio** possiamo usare il seguente comando (anch'esso presente nel file .justfile): `docker compose down`.

3) Collegamento di un client di posta elettronica installato su un dispositivo Android

3.1) Operazioni preliminari

Per collegare un client di posta elettronica installato su un dispositivo (come ad esempio un cellulare) Android è necessario prima eseguire delle verifiche per aumentare il successo dell'operazione.

Prima di tutto dobbiamo assicurarci che il dispositivo in questione sia collegato alla stessa LAN del server e che possa comunicare con quest'ultimo. Per verificare questo dobbiamo installare un software di ping sul dispositivo Android e provare ad effettuare un DNS Lookup per il sottodominio configurato precedentemente sulla sezione: **Duck DNS**.

Qualora tutto fosse installato correttamente questo dovrebbe essere il risultato:

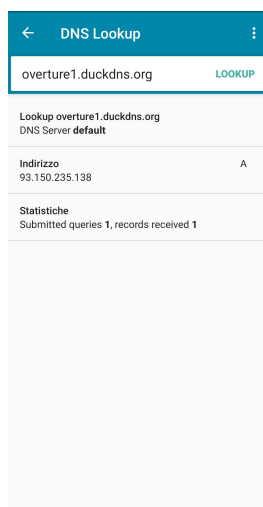


Figura 3: Ping tra il dispositivo mobile e il server avvenuto con successo.

3.2) Installazione, avvio del client e collegamento

Come prima cosa dobbiamo scegliere uno dei client presenti nel seguente repository pubblico nel sito ufficiale di jmap: <https://jmap.io/software.html>. Fatto questo, dobbiamo installarlo nel dispositivo in uso seguendo la guida ufficiale associata al README presente su ogni client scelto nella pagina github relativa.

Nel nostro caso noi abbiamo scelto di utilizzare il client Ltt.rs (<https://github.com/iNPUTmice/lttrs-android>).

Arrivati a questo punto avviamo il client ed effettuiamo l'accesso con uno degli account presenti nel file .env del server installato precedentemente. A partire dall'indirizzo email dell'account con il quale si vuole effettuare l'accesso il client riconoscerà automaticamente il dominio (a partire da alice@overture.duckdns.org, il client saprà che dovrà collegarsi al server presente al dominio overture.duckdns.org e quindi all'indirizzo IP relativo al record DNS fornito da Duck DNS).

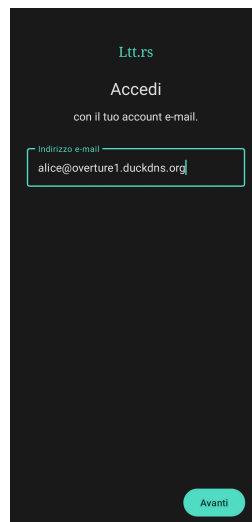


Figura 4: Inserimento dell'email con il quale si vuole effettuare l'accesso.

Successivamente inseriamo la password associata a quell'account (la troviamo nel file .env) e clicchiamo su "Avanti":

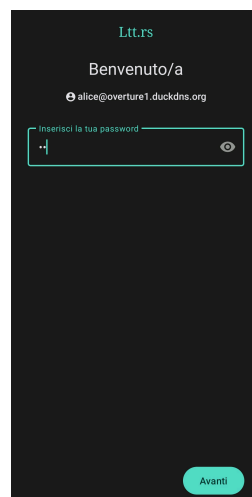


Figura 5: Inserimento della password relativa all'account inserito prima.

Se tutto è stato eseguito correttamente avremo il seguente risultato:

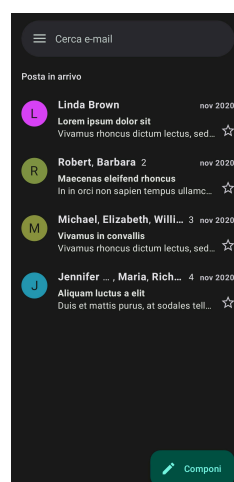


Figura 6: Schermata principale con la lista di tutte le email ricevute.

4) Guida all'uso di Postman

4.1) Introduzione

Questa sezione ha lo scopo di guidare l'utente attraverso il processo di importazione delle richieste, la visualizzazione dei dettagli di ciascuna richiesta, l'esecuzione dei test e l'interpretazione delle risposte utilizzando l'applicazione Postman.

4.2) Installazione ed accesso

Per iniziare, il primo passo è scaricare e installare l'applicazione sul proprio dispositivo. Puoi scegliere tra una versione gratuita e una a pagamento, ma per le nostre necessità, la versione gratuita sarà più che sufficiente. Puoi trovare il link per il download qui: <https://www.postman.com/downloads/>. Una volta completata l'installazione, procedi creando un account oppure accedi se ne possiedi già uno. Questa operazione ti permetterà di accedere alle funzionalità necessarie per raggiungere il nostro scopo.

4.3) Importazione delle richieste

Dopo aver effettuato l'accesso, all'interno della pagina principale di Postman si troveranno varie opzioni. L'operazione che ci interessa svolgere è l'importazione del file JSON contenente tutte le richieste. Questa può essere fatta selezionando il tasto "Import" situato in alto a sinistra, accanto al nome del nostro Workspace.

Una volta importato il file JSON, verrà creata una cartella denominata "JMAP" contenente tutte le richieste gestite dal server. Ogni richiesta è denominata con il suo specifico nome e al suo interno si trova una serie di dettagli. Per semplicità, è stato incluso un solo esempio per ciascuna richiesta.

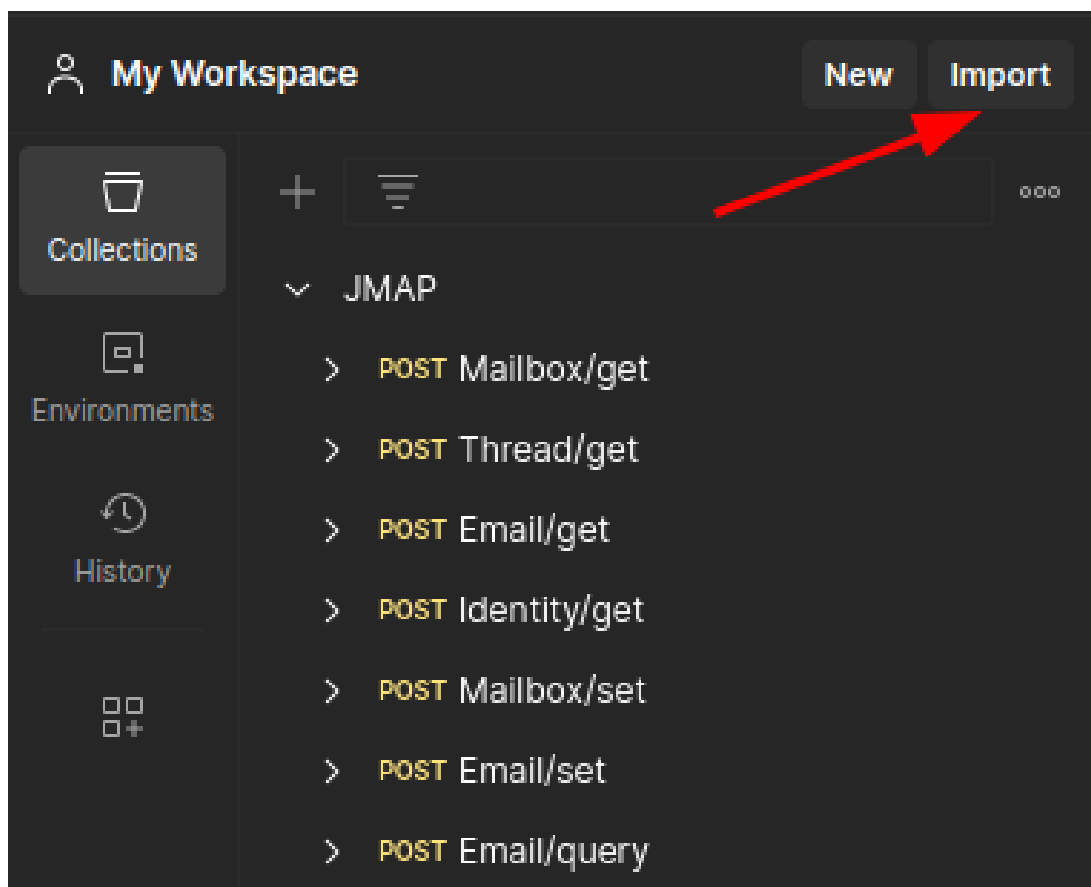
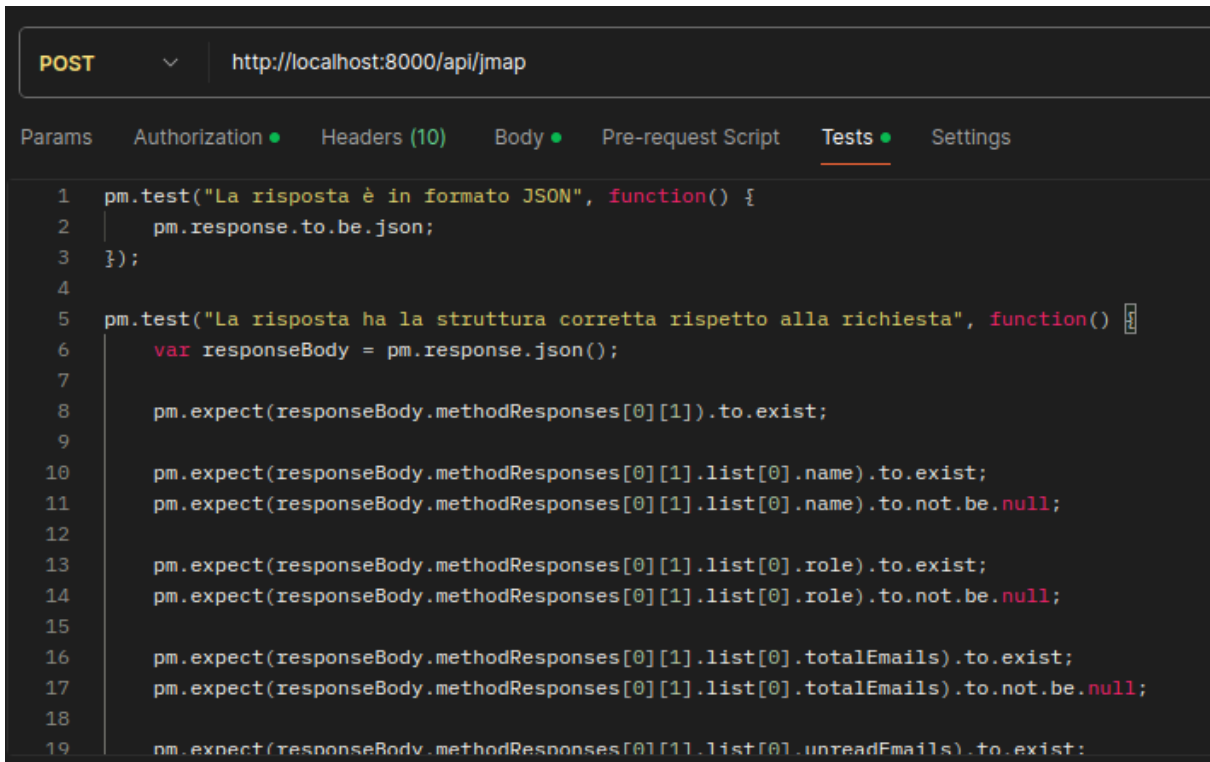


Figura 7: Importazione del file JSON contenente tutte le richieste in Postman.

4.4) Visualizzazione dei dettagli della richiesta

Quando si seleziona una richiesta è possibile visualizzarne i dettagli, compreso l'indirizzo a cui viene inviata, i parametri, l'autorizzazione, gli headers, il body ed i test da eseguire. Le schede rilevanti sono contrassegnate da un cerchio verde, indicando che sono state modificate rispetto al valore predefinito. Ad esempio, è possibile vedere l'autorizzazione eseguita a nome dell'utente "alice", il corpo della richiesta in formato JSON e i test JavaScript che verranno eseguiti sulla risposta del server per verificare la correttezza della struttura e dei parametri ricevuti.



```
1 pm.test("La risposta è in formato JSON", function() {
2   pm.response.to.be.json;
3 });
4
5 pm.test("La risposta ha la struttura corretta rispetto alla richiesta", function() {
6   var responseBody = pm.response.json();
7
8   pm.expect(responseBody.methodResponses[0][1]).to.exist;
9
10  pm.expect(responseBody.methodResponses[0][1].list[0].name).to.exist;
11  pm.expect(responseBody.methodResponses[0][1].list[0].name).to.not.be.null;
12
13  pm.expect(responseBody.methodResponses[0][1].list[0].role).to.exist;
14  pm.expect(responseBody.methodResponses[0][1].list[0].role).to.not.be.null;
15
16  pm.expect(responseBody.methodResponses[0][1].list[0].totalEmails).to.exist;
17  pm.expect(responseBody.methodResponses[0][1].list[0].totalEmails).to.not.be.null;
18
19  pm.expect(responseBody.methodResponses[0][1].list[0].unreadEmails).to.exist;
```

Figura 8: Schermata di visualizzazione dei dettagli della richiesta.

4.5) Esecuzione dei test ed interpretazione delle risposte

Tornando alla lista delle richieste sulla sinistra, ciascuna è associata a una risposta con lo stesso nome, contenente un esempio di risposta corretta. Selezionando una risposta è possibile cliccare su “Try” in alto a destra per effettuare effettivamente la richiesta. Il risultato verrà visualizzato nella parte inferiore della schermata, mostrando il corpo della nuova risposta, eventuali cookies, gli headers ed i risultati dei test. Quest’ultimi includono una preview dei test eseguiti e di quelli che sono passati correttamente. Selezionando questa sezione, verranno visualizzati i nomi dei test effettuati ed i relativi risultati.

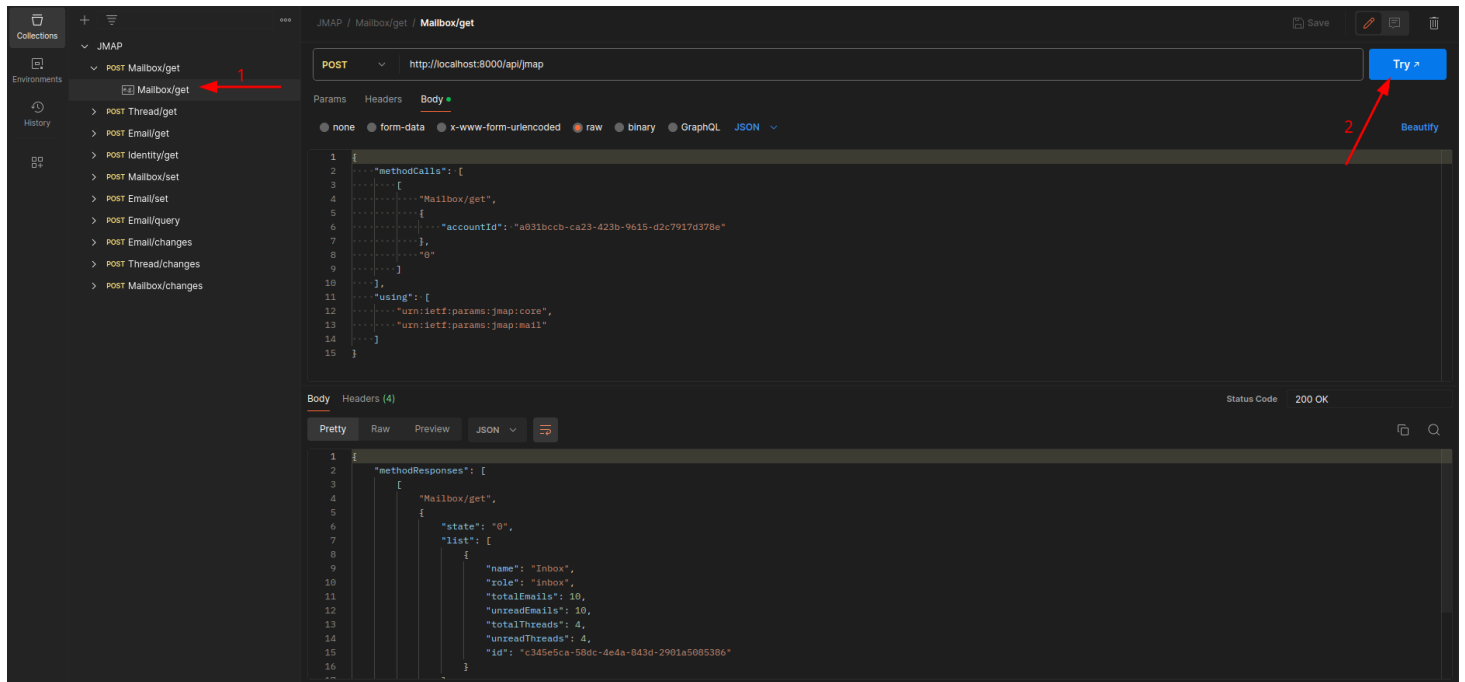


Figura 9: Schermata di esecuzione dei test.

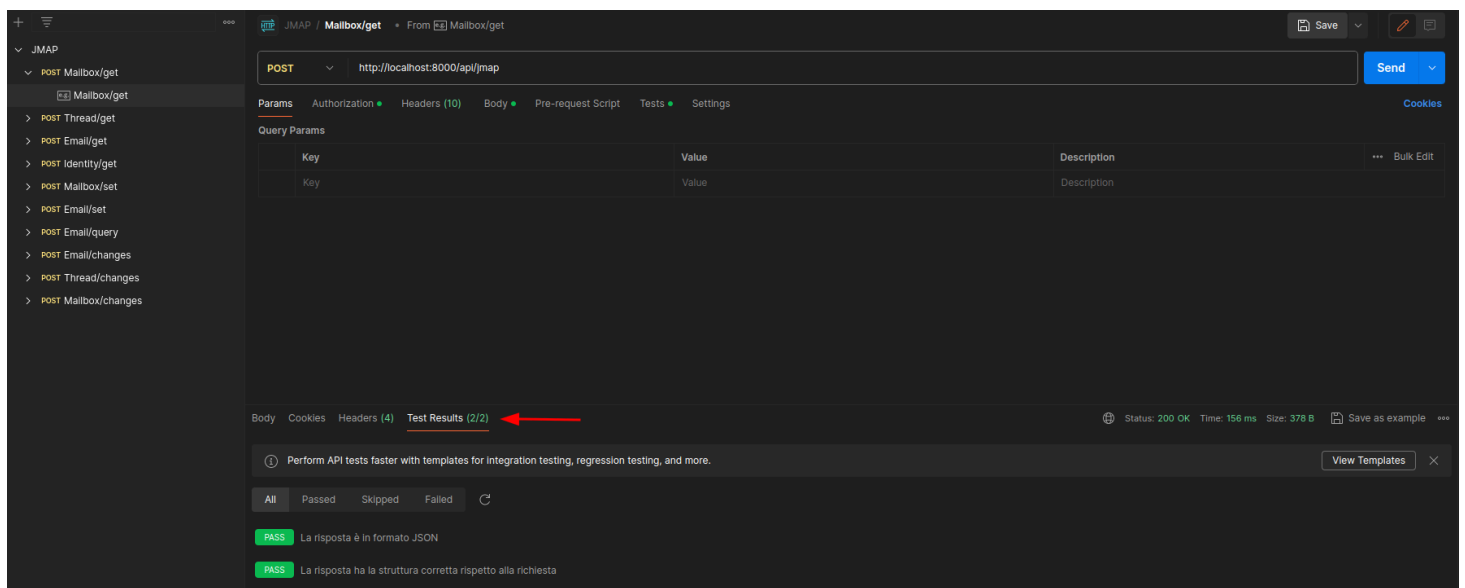


Figura 10: Visualizzazione del risultato dei test eseguiti sulla richiesta fornita.

5) Avvio degli stress test

Questa sezione fornisce istruzioni dettagliate su come eseguire gli stress test utilizzando il framework Locust, attraverso il quale è possibile simulare il carico di lavoro su un sistema e valutarne le prestazioni.

I test sono organizzati in file Python denominati `test1`, `test2` e `test3`, con i dettagli delle richieste contenuti nei file JSON presenti nella cartella `requests`.

5.1) Requisiti preliminari

Assicurarsi di avere Python installato sul sistema e di disporre del framework Locust. Nel caso in cui quest'ultimo non sia presente nel vostro sistema, per installarlo eseguire il seguente comando sul terminale: `pip install locust`. Di seguito la guida ufficiale per l'installazione: <https://docs.locust.io/en/stable/installation.html>.

5.2) Descrizione dei test

5.2.1) `test1` - Accesso all'inbox

Simula una grande quantità di utenti che accedono all'inbox contemporaneamente. Questo test è finalizzato a valutare le prestazioni del server durante l'accesso simultaneo di numerosi utenti alla propria inbox. Il codice è stato sviluppato per simulare l'intero processo di visualizzazione dell'inbox, dalla fase di apertura dell'applicazione fino alla visualizzazione effettiva delle email, al fine di ricreare uno scenario realistico.

All'inizio del codice sono definite variabili che consentono la personalizzazione dei dati del server, come il dominio e gli utenti. Successivamente vengono aperti e memorizzati i payload delle richieste da inviare successivamente.

Il codice, eseguito ripetutamente da Locust, seleziona casualmente un utente tra quelli disponibili e invia una richiesta GET per ottenere l'ID dell'account, fondamentale per le richieste future. Successivamente vengono inviate solo richieste POST contenenti i payload memorizzati in precedenza. Dopo un'ulteriore verifica tramite una richiesta `Identity/get`, viene visualizzato l'elenco delle caselle di posta disponibili, da cui viene estratto l'ID dell'inbox, necessario per la visualizzazione delle email. In seguito viene ottenuto lo stato della sessione corrente, utilizzato insieme all'ID dell'account e dell'inbox per richiedere la visualizzazione delle email contenute nell'inbox. Al termine, viene eseguita una richiesta per aggiornare lo stato della sessione, in preparazione a eventuali future richieste.

5.2.2) `test2` - Invio di email

Crea uno scenario in cui molti utenti inviano email contemporaneamente, variando il numero di destinatari e la dimensione dell'email per testare la capacità del server di gestire carichi di lavoro diversi. Il test simula lo scenario in cui un utente, dopo essersi autenticato nell'applicazione e aver visualizzato l'inbox, decide di inviare una email, al fine di valutare il comportamento del server in questa operazione.

Il codice presenta molte analogie con il test precedente, in quanto l'utente visualizza la pagina di inbox prima di poter eseguire qualsiasi altra operazione.

Dopo l'autenticazione e l'ottenimento dell'ID dell'utente, viene effettuata una richiesta per visualizzare le caselle di posta disponibili, tra cui quella delle email inviate, che sarà necessaria per l'invio della nuova email. Nel caso in cui non siano state inviate email in precedenza e di conseguenza la casella di posta non esista ancora, questa verrà creata successivamente. Dopo aver visualizzato l'inbox come nel test precedente, viene creata la casella di posta per le email inviate, se non esiste ancora, e viene inviata effettivamente la richiesta per inviare l'email. In questo caso, l'oggetto dell'email rimane identico per

tutte le richieste poiché ininfluente, mentre il numero di destinatari viene scelto casualmente per un massimo di 3 e il contenuto dell'email ha una lunghezza variabile tra 16 e 3200 caratteri, al fine di testare la capacità del server di gestire differenti carichi di lavoro.

5.2.3) test3 - Gestione delle cartelle

Questo test è progettato per valutare le prestazioni del server durante la creazione ed eliminazione simultanea di cartelle da parte di numerosi utenti. Anche in questo caso il test simula lo scenario in cui un utente si autentica ed accede alla propria inbox, dopodiché procede alla creazione di una cartella e alla sua eliminazione.

Esattamente come nel test di Accesso all'Inbox, l'utente si autentica e, dopo aver ricevuto tutti i parametri necessari come descritto nei test precedenti, esegue la richiesta per la visualizzazione dell'Inbox. A questo punto è come se l'utente fosse davanti alla schermata iniziale del suo client e proseguisse con la creazione di una nuova cartella chiamata "Test". Dopo aver ottenuto l'identificativo di quest'ultima, procede alla sua eliminazione.

5.3) Esecuzione dei test

- Passo 1 - Modifica dei parametri:
 - Passo 1.1: Aprire il file Python relativo al test desiderato;
 - Passo 1.2: Modificare le informazioni necessarie nelle prime righe del file per adattare il test al server e agli utenti desiderati. I valori corretti verranno sostituiti automaticamente lungo tutto il codice.
- Passo 2 - Avvio del test:
 - Passo 2.1: Aprire il terminale;
 - Passo 2.2: Eseguire il comando seguente:

```
locust -f test.py --host=http://localhost:8000
```

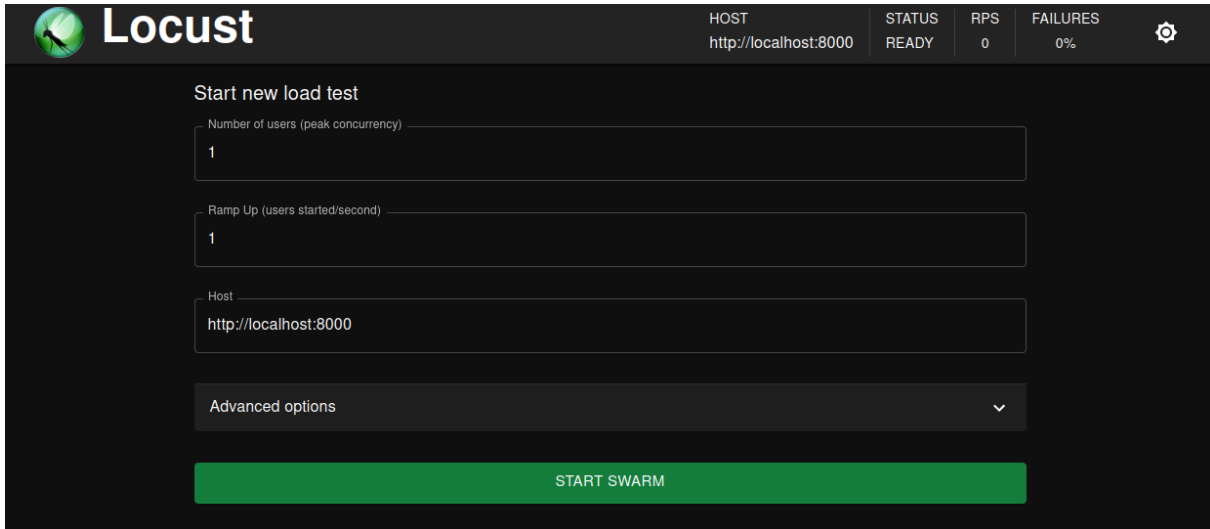
Sostituendo "test.py" con il nome del file del test da eseguire e "<http://localhost:8000>" con l'indirizzo IP del server.

- Passo 3 - Visualizzazione dell'interfaccia utente:
 - Passo 3.1: Aprire qualsiasi browser e accedere all'indirizzo <http://localhost:8089>. Da ora sarà possibile interagire con Locust attraverso l'interfaccia grafica.

5.3.1) Configurazione dei parametri

Durante l'utilizzo dell'interfaccia utente, saranno richieste le seguenti informazioni:

- Numero di utenti (frequenza massima): indica il numero massimo di utenti simulati durante il test. Ad esempio, impostando il numero di utenti a 100, Locust simulerà l'attività di 100 utenti che interagiscono con il sistema contemporaneamente;
- Aumento graduale (utenti inizializzati al secondo): definisce il tasso di aggiunta di nuovi utenti simulati nel tempo. Ad esempio, impostando l'aumento graduale a 10 utenti al secondo e il numero di utenti a 100, Locust inizierà con 10 utenti e ne aggiungerà altri 10 ogni secondo fino al raggiungimento del numero massimo di 100 utenti.



The screenshot shows the Locust web interface. At the top, there's a header with the Locust logo on the left and a status bar on the right. The status bar includes: HOST (http://localhost:8000), STATUS (READY), RPS (0), FAILURES (0%), and a settings gear icon. Below the header, the main area is titled 'Start new load test'. It contains three input fields: 'Number of users (peak concurrency)' with the value '1', 'Ramp Up (users started/second)' with the value '1', and 'Host' with the value 'http://localhost:8000'. Below these fields is a dropdown menu labeled 'Advanced options'. At the bottom of the form is a large green button labeled 'START SWARM'.

Figura 11: Schermata di configurazione dei parametri in Locust.

5.3.2) Monitoraggio dei risultati

Una volta avviata la simulazione, verrà presentata un'interfaccia intuitiva che fornisce una vasta gamma di informazioni. Queste includono il tipo di richiesta, il suo nome e il numero di richieste effettuate, insieme ad altri dettagli tecnici pertinenti. In alto a destra è possibile regolare i parametri della simulazione in tempo reale senza interrompere il processo. È inoltre possibile anche interrompere e ripristinare la simulazione in qualsiasi momento.

Navigando tra le schede successive si avrà accesso ad ulteriori dettagli sulla simulazione in corso. Lì si troveranno grafici chiari che mostrano le richieste totali inviate al secondo, i tempi di risposta e il numero di utenti attivi. Ulteriori schede offriranno informazioni ancora più dettagliate, consentendo inoltre di scaricare i dati generati durante la simulazione e visualizzare i log forniti da Locust.

Per ulteriori informazioni e dettagli sul framework Locust, si consiglia di consultare la documentazione ufficiale: <https://docs.locust.io/en/stable/>.

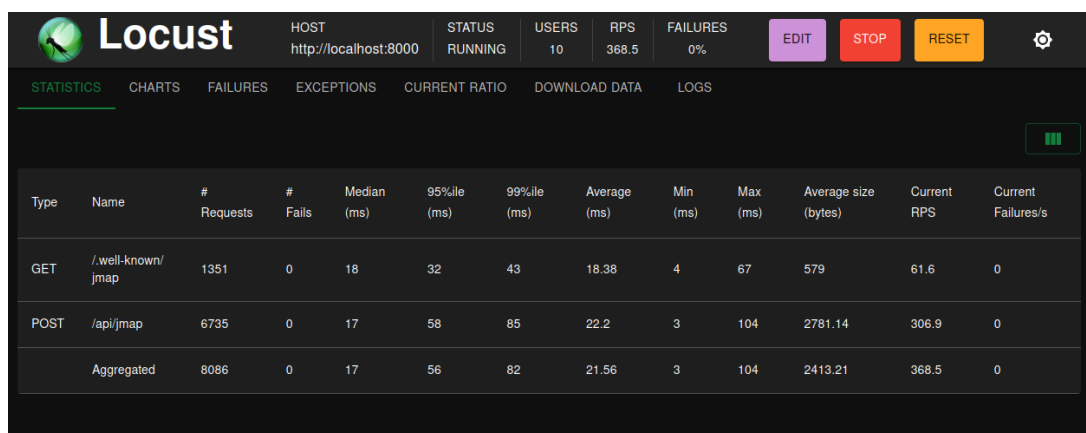


Figura 12: Schermata di monitoraggio dei risultati.



Figura 13: Schermata di monitoraggio dei grafici.

6) Supporto tecnico

Il nostro team è disponibile per assisterti in caso di problemi o domande sul nostro prodotto. Dunque se riscontri difficoltà tecniche, oppure hai dei dubbi riguardo alle funzionalità o all'installazione, non esitare a contattarci inviando un'email all'indirizzo: overture.unipd@gmail.com.

Assicurati di includere una descrizione dettagliata del problema o della domanda, eventuali messaggi di errore che hai ricevuto, la data in cui il malfunzionamento è stato riscontrato e qualsiasi altro dettaglio che ci possa permettere di assisterti nel miglior modo possibile. Il nostro team farà del suo meglio per risolvere rapidamente qualsiasi problema e fornirti le informazioni necessarie per utilizzare al meglio il frutto del nostro lavoro.