Manuale Utente
Versione 1.0.0

Stato	Approvato
Responsabile	Matteo Schievano
Verificatore	Emanuele Artusi Matteo Schievano Samuele Esposito Loris Libralato
Redattori	Samuele Esposito Matteo Schievano Marco Piccoli
Distribuzione	*ALimitedGroup* M31 Prof. Tullio Vardanega Prof. Riccardo Cardin

Descrizione

Questo documento contiene le informazioni utili per l'utilizzo del *Minimum Viable Product* realizzato da *ALimitedGroup*

Registro delle Modifiche

Vers.	Data	Autore	Verificatore	Descrizione
1.0.0	2025-04-04	S. Esposito	L. Libralato	Correzioni minori. Approvazione da parte di M. Schievano.
0.4.0	2025-04-04	M. Schievano	S. Esposito	Cambiamento versione scaricabile.
0.3.0	2025-04-02	M. Schievano	S. Esposito	Redazione installazione e configurazione Sistema, Test, Guide e Telemetria (Sezione 4.2.1, Sezione 3.1, Sezione 5.2, Sezione 6.1).
0.2.0	2025-03-19	M. Piccoli	S. Esposito	Redazione sezione Merci, Tipi di Utenti, Ordini, Trasferimenti e Riassortimenti. Stesura Endpoint. (Sezione 2.1, Sezione 2.3, Sezione 2.4, Sezione 2.5, Sezione 2.6, Sezione 7.12).
0.1.0	2025-02-02	S. Esposito	E. Artusi	Preparazione base documento. Stesura parte introduttiva del documento e requisiti del Sistema (Sezione 1 e Sezione 4.1).

Indice

Lista delle tabelle

Lista delle immagini

1 - Introduzione

1.1 - Scopo del documento

Lo scopo del **Manuale Utente**[G] è quello di introdurre il lettore al Sistema sviluppato da *ALimitedGroup* per il **Capitolato C6: Sistema di Gestione di un Magazzino Distribuito**.

A tale scopo il manuale illustrerà i requisiti minimi necessari ad avviare il Sistema, a partire dai programmi necessari alla sua esecuzione, per poi proseguire a descrivere come configurare il Sistema in base alle proprie esigenze, e come avviare lo stesso.

Successivamente, vengono descritte le operazioni necessarie per la corretta operazione e manutenzione[G] del Sistema, in particolare su periodi di tempo prolungati.

La parte finale di questo manuale ha l'obiettivo di esporre tutte le conoscenze necessarie per un utilizzo efficace del Sistema. Ciò è fatto seguendo i principi di Diátaxis, un "Approccio sistematico alla produzione di documentazione tecnica". In particolare, sono presenti le seguenti sezioni:

- **Concetti**: una spiegazione completa dei concetti chiave che potrebbe essere necessario conoscere per usare il Sistema. Essa segue la forma di documentazione *Explanation*, come indicato dal Diátaxis;
- **Test**[G]: una breve introduzione all'esecuzione dei test[G] di unità, integrazione e accettazione;
- **Installazione**: una spiegazione completa su come scaricare, installare, configurare ed avviare il Sistema;
- **Guide**: delle guide rapide con l'obiettivo di guidare l'utente mentre effettua una particolare azione. Essa segue la forma di documentazione *How-to guides*, come indicato dal Diátaxis;
- **Telemetria**: una guida alla configurazione di Grafana[G] e al suo utilizzo.
- **Riferimento *API***: specifica con il massimo dettaglio ogni aspetto delle chiamate *API*[G] possibili; utile per l'utente che necessita di una specifica dettagliata del funzionamento dell'*API*. Essa segue la forma di documentazione *Reference*, come indicato dal Diátaxis.

1.2 - Glossario

La realizzazione di un Sistema software richiede, ancor prima della scrittura del codice, un'importante operazione di confronto, analisi e progettazione: per supportare e facilitare il lavoro asincrono tutte le informazioni derivate da questa attività saranno appositamente documentate.

È completamente ragionevole tuttavia pensare che tali documenti potrebbero contenere parole e terminologie complesse o comunque non direttamente comprensibili: è stato deciso dunque di realizzare un Glossario, nella quale saranno contenuti le spiegazioni relative a tali termini. Tale documento è in costante aggiornamento ed è reperibile, nella sua versione attuale, all'indirizzo https://alimitedgroup.github.io/glossario.html. È disponibile anche una versione in formato pdf.

Le parole che possiedono un riferimento nel Glossario saranno indicate nel modo seguente:

parola[G]

1.3 - Riferimenti

1.3.1 - Riferimenti normativi

- **Norme di Progetto[G] versione 2.0.0**

  https://alimitedgroup.github.io/NP%20v2.0.0.pdf

  Ultimo Accesso 4 Aprile 2025

- **Capitolato C6: Sistema di Gestione di un Magazzino Distribuito**

  https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C6.pdf

  Ultimo Accesso 4 Aprile 2025

1.3.2 - Riferimenti informativi

- **Glossario**

  https://alimitedgroup.github.io/Gloss%D0%B0rio.pdf

  Ultimo Accesso 4 Aprile 2025

2 - Concetti

2.1 - Tipi di utenti

Nel Sistema che *ALimitedGroup* realizzerà ed implementerà, sono presenti tre tipi di utenti:

- Il **Cliente**
- L'**Admin Locale**
- L'**Admin Globale**.

2.1.1 - Cliente

Il **Cliente** è la persona che utilizzerà il prodotto finale ed usufruirà dei servizi offerti dal Sistema con, chiaramente, delle limitazioni riguardo certe operazioni. Questo tipo di utente potrà effettuare le seguenti azioni all'interno del Sistema:

- **Autenticare**, immettendo il ruolo tramite l'apposita *API*[G] offerta;
- **Creare un ordine**[G] inserendo il nome, il destinatario dell'ordine e l'indirizzo di spedizione;
- **Selezionare la merce per l'ordine**[G] inserendo la quantità, per ognuna delle merci aggiunte all'ordine;
- **Confermare l'ordine**[G] ed in questo modo far partire la richiesta per l'ordine, in attesa che venga elaborata;
- **Visualizzare la lista degli ordini non confermati** e, per ognuno di essi, visualizzarne le informazioni come:
  - L'**ID** dell'ordine;
  - La **Data di creazione** dell'ordine;
  - Il **Nome** dell'ordine;
  - La **Lista delle merci** nell'ordine e, per ognuna di questa:
    - Il **Nome** della merce aggiunta all'ordine;
    - La **Quantità** della merce nell'ordine.
- **Visualizzare la lista delle merci disponibili nel Sistema** e, per ognuna di essa, visualizzare:
  - Il **Nome** della merce;
  - L'**ID** della merce;
  - La **Descrizione** della merce;
  - La **Quantità Complessiva** in tutti i magazzini presenti nel Sistema;
  - La **Quantità** disponibile nel magazzino più vicino al Cliente.

2.1.2 - Admin Locale

L'**Admin Locale** è la persona responsabile[G] per un singolo magazzino, con un interesse "locale" alla situazione di quest'ultimo. Inoltre, rispetto al Cliente, l'Admin Locale ha un maggior potere decisionale sulle azioni da intraprendere all'interno del magazzino. Le operazioni che questo tipo di utente ha la possibilità di effettuare, sono le seguenti:

- **Aggiungere** ed aumentare lo *stock*[G] disponibile per una merce, inserendone la quantità.

2.1.3 - Admin Globale

L'**Admin Globale** ha la responsabilità di supervisionare tutti i magazzini presenti, ha quindi un interesse "globale" sul Sistema e tutte le sue componenti. Questa tipologia di utente è la più importante, dal punto di vista decisionale, ma soprattutto la più delicata richiedendo che

ci sia un'elevata attenzione ad ogni azione presa. La lista di operazioni disponibili per questo tipo di utente è la seguente:

- **Creare** un trasferimento[G] di merci da confermare, scegliendo per ognuno:
  - Il **magazzino mittente** del trasferimento[G]
  - Il **magazzino destinatario** del trasferimento[G].
- **Aggiungere** la merce ad un trasferimento[G] non confermato, a patto che:
  - Il **magazzino mittente** del trasferimento[G], abbia la quantità di merce sufficiente nello *stock*[G]
  - **Esista** un trasferimento[G] precedentemente creato.
- **Confermare** un trasferimento[G] così da procedere con la richiesta, a patto che:
  - Il magazzino mittente del trasferimento[G], abbia la quantità di merce sufficiente nello *stock*[G]
  - Esista un trasferimento[G] non confermato nel Sistema.
- **Cancellare** un trasferimento[G] non ancora confermato, a patto che ne esista uno;
- **Visualizzare** l'elenco di tutti trasferimenti presenti nel Sistema, visualizzando per ognuno:
  - L'**ID** del trasferimento[G]
  - Lo **Stato** del trasferimento[G].
- **Visualizzare** nel dettaglio un particolare trasferimento[G], visualizzando:
  - L'**ID** del trasferimento[G]
  - Lo **Stato** del trasferimento[G]
  - Il **Magazzino mittente** del trasferimento[G]
  - Il **Magazzino destinatario** del trasferimento[G]
  - L'**Elenco delle merci** interessate nel trasferimento[G], e per ognuna di queste:
    - Il **Nome** della singola merce nel trasferimento[G]
    - La **Quantità** della singola merce nel trasferimento[G].
- **Visualizzare** l'elenco delle notifiche di rifornimento, a patto che ne esistano, e per ognuna visualizzare:
  - L'**ID** della notifica di rifornimento;
  - Lo **Stato** della notifica di rifornimento.
- **Visualizzare** nel dettaglio una notifica di rifornimento, visualizzando:
  - Lo **Stato** della notifica;
  - L'**ID** della notifica;
  - Il **Magazzino** di destinazione del rifornimento;
  - L'**Elenco delle merci** che è consigliato rifornire, e per ognuna delle merci è possibile visualizzare:
    - L'**ID** della merce;
    - La **Quantità** consigliata da rifornire della merce;
    - Il **Nome** della merce.
- **Accettare** una notifica di rifornimento, purchè ne esista una;
- **Rifiutare** una notifica di rifornimento, a condizione che ne esista una;
- **Visualizzare** l'elenco dei microservizi, e per ognuno di essi visualizzare:
  - Il **Numero di richieste al secondo** del microservizio;

    - Il **Nome** del microservizio.
- **Esportare** gli ordini eseguiti, purché ce ne siano, in un *file* .csv;
- **Esportare** il *report* dell'inventario, a patto che ce ne esista uno, in un *file* .csv;
- **Creare** una soglia minima, a condizione che sia valida, in una determinata merce;
- **Creare** una nuova merce, inserendo:
    - Il **Nome** della nuova merce;
    - La **Descrizione** della nuova merce.
- **Aggiornare** le informazioni su una determinata merce.

2.1.4 - Riepilogo degli Utenti presenti

Tipo di Utente	Descrizione	Operazioni
Cliente	È la persona che utilizzerà il prodotto finale ed usufruirà dei servizi offerti dal Sistema, con delle limitazioni.	Può effettuare ordini? Sì. Può visualizzare le merci nel Sistema? Sì. Può confermare gli ordini? Sì. Può visualizzare gli ordini effettuati? Sì. Può aumentare lo stock di una certa merce? No. Può creare una nuova merce? No. Può creare un trasferimento tra magazzini? No.
Admin Locale	È la persona responsabile di un singolo magazzino, con un interesse «locale» verso di esso.	Può effettuare ordini? No. Può visualizzare le merci nel Sistema? Sì. Può confermare gli ordini? No. Può visualizzare gli ordini effettuati? No. Può aumentare lo stock di una certa merce? Sì. Può creare una nuova merce? No. Può creare un trasferimento tra magazzini? No.
Admin Globale	È la persona che ha la responsabilità di supervisionare su tutti i magazzini presenti nel Sistema; è la persona più importante e più delicata dal punto di vista decisionale.	Può effettuare ordini? No. Può visualizzare le merci nel Sistema? No. Può confermare gli ordini? No. Può visualizzare gli ordini effettuati? No. Può aumentare lo stock di una certa merce? No. Può creare una nuova merce? Sì. Può creare un trasferimento tra magazzini? Sì.

Tabella 1: Riepilogo degli Utenti presenti nel Sistema

2.2 - Autenticazione ed autorizzazione

Per mantenere un Sistema sicuro, e per evitare azioni accidentali o volute da parte di utenti non autorizzati a compierle, molti sistemi richiedono agli utenti di identificarsi, prima di svolgere molte delle operazioni. Questo processo viene chiamato **autenticazione**. Il fatto che un utente sia autenticato, però, non garantisce che esso possa effettuare ogni operazione implementata dal Sistema: il processo che controlla se l'utente ha i privilegi di effettuare le operazioni che richiede si chiama **autorizzazione**.

Nel Sistema descritto dal presente manuale, i processi di Autenticazione e di Autorizzazione sono parzialmente separati mediante l'utilizzo di *token*, stringhe di testo che solo il Sistema può generare, utilizzate appunto per identificare l'utente. Ogni utente è associato a uno e un solo **ruolo**, ed ogni utente assegnato ad uno stesso ruolo può effettuare le stesse operazioni.

L'interazione tra utente non autenticato e Sistema può quindi essere riassunta nelle seguenti fasi:

1. L'utente effettua una richiesta di login;
2. Il Sistema controlla le credenziali fornite, restituendo un token se sono valide.

Ottenuto il *token*, l'utente potrà interagire con il Sistema nel seguente modo:

1. L'utente effettua una richiesta al Sistema, allegando il proprio *token*
2. Il Sistema controlla che la richiesta abbia un *token* associato, e che sia valido; se non lo è, restituisce un messaggio d'errore;
3. Il Sistema recupera il ruolo dell'utente autenticato, controlla se esso ha l'autorizzazione di effettuare l'operazione richiesta, e consente o nega l'operazione di conseguenza.

2.2.1 - *Token*

Il Sistema utilizza dei *token* con un formato ben conosciuto: si tratta di *token* JWT firmati con algoritmo ES256. I JWT sono tipicamente trasmessi sotto forma di una stringa composta da tre parti separate da punti: `xxxxx.yyyyy.zzzzz`. Le prime due parti sono oggetti JSON, codificati con *encoding* `Base64Url`, mentre l'ultima è una firma crittografica.

La prima parte viene chiamata *header*. Essa contiene l'algoritmo di firma utilizzato e il tipo di token. Nel caso del Sistema, l'*header* ha questa struttura:

```
{
  "alg": "ES256",
  "typ": "JWT"
}
```

La seconda parte viene chiamata *payload*, e contiene vari campi utili per identificare l'utente. Nel caso del Sistema, i campi presenti sono i seguenti:

```
{
    "sub": "username",
    "role": "local_client",
    "exp": 1355310732,
    "iss": "31c111ab-9ed4-49da-b4c0-8641514a4589",
}
```

I due campi `sub` e `role` contengono rispettivamente il nome utente ed il ruolo. Il campo `exp` contiene una data di scadenza del token, espressa come numero di secondi passati dal 1°

gennaio 1970 (una convenzione comunemente chiamata "*Unix Timestamp*"). Il campo `iss`, invece, contiene informazioni su chi ha emesso il JWT. Data la natura distribuita del Sistema, infatti, più componenti dello stesso tipo possono generare i *token*, ognuno firmando con una propria chiave privata. Il campo `iss` è fondamentale per identificare quale tra questi abbia emesso il JWT, così da poter recuperare la chiave pubblica necessaria per verificarne la validità.

La terza e ultima parte, invece, è una firma crittografica che attesta la validità del *token*. Senza entrare troppo nei dettagli, è usato un Sistema di **crittografia asimmetrica**, in cui è presente una coppia di chiavi definite "pubblica" e "privata". Utilizzando la chiave privata, è possibile generare una "firma" digitale che garantisce che i dati firmati non siano stati alterati rispetto a quando la firma è stata generata. È successivamente possibile verificare la firma anche se si è in possesso solo della chiave pubblica. In questo modo, il Sistema può inserire dati pubblicamente disponibili all'interno del *token* (ricordiamo che le sezioni *header* e *payload* sono pubblicamente leggibili, essendo codificate ma non crittografate), senza paura che vengano alterati (l'utente non possiede la chiave privata, quindi non può generare una firma valida per il *token* modificato).

Qualora il lettore desiderasse più dettagli, si rimanda alla lettura di https://jwt.io/introduction, sito che fornisce una rapida introduzione ai JWT. Qualsiasi dettaglio più specifico può essere reperito all'interno dell'RFC 7519, il documento che descrive i JWT, e all'RFC 7518, il quale descrive `ES256`, l'algoritmo di firma che il Sistema utilizza.

2.2.2 - Procedura di autenticazione

L'utente che volesse autenticarsi presso il Sistema dovrà inviare una richiesta `POST` al percorso `/api/v1/login`, allegando le proprie credenziali. Il Sistema, dopo aver effettuato le verifiche necessarie, risponderà con un *token* che l'utente dovrà conservare.

Per le richieste successive, qualora sia richiesta autenticazione, l'utente dovrà includere un *header* `Authorization` nella richiesta, con valore `Bearer <token>` (rimpiazzando `<token>` con il valore ottenuto durante il *login*).

2.3 - Merce

Nel *Minimum Viable Product*[G], che *ALimitedGroup* ha progettato ed implementato, e nei documenti annessi viene fatto riferimento ad una serie di termini, uno tra questi è: la **Merce**.

Viene subito da porsi la seguente domanda: *Cosa si intende per Merce? E cosa ALimitedGroup intende con Merce?*

La risposta è la seguente: *ALimitedGroup* con il termine "Merce", più appropriamente chiamato *Good*[G], intende una t-upla, ovvero una sequenza di valori specifici ed univoci, che identifichi tale Merce su cui stiamo lavorando. All'interno di questa sequenza di valori possiamo trovare:

- L'**Identificatore Univoco** della merce;
- Il **Nome** della merce;
- La **Descrizione** associata alla merce.

All'interno dell'MVP sono presenti dei metadati extra, quest'ultimi vengono codificati come stringa tramite lo standard *Unicode UTF-8* questi metadati sono utili per l'utilizzo corretto

dell'MVP realizzato. Di seguito sono riportati alcuni esempi pratici su alcuni usi utili di questi metadati:

- **Salvare** l'identificatore unico che ha una determinata merce nel sito del fornitore, e da cui devo rifornirmi per aumentare lo *stock*[G] nel magazzino;
- **Salvare** l'ID che ha una merce nel sito di un cliente, per effettuare un trasferimento[G] a quest'ultimo;
- **Salvare** quando è avvenuto l'ultimo *restock* nel magazzino di una merce selezionata.

Questi sono solo alcuni esempi, ma potremmo scriverne ancora per comprenderne ulteriormente l'importanza. Perciò, *ALimitedGroup* consiglia di salvare i metadati presenti all'interno della merce in un *file* .json.

2.4 - Ordini

Nel *Minimum Viable Product*[G], che *ALimitedGroup* ha progettato ed implementato, e nei documenti annessi viene fatto riferimento ad una serie di termini, uno tra questi è: l'**Ordine**.

Viene subito da chiedersi: *Che cos'è un "Ordine"? E cosa intende ALimitedGroup per "Ordine"?*.

Un "Ordine" viene inteso come un qualsiasi invio di merci, unilateralmente, tra due entità; questo invio deve avere queste informazioni al suo interno:

- Un **Identificativo Unico** che rappresenta questo invio;
- La **Quantità** di merce che viene inviata da parte del mittente;
- L'**Indirizzo del Mittente** dell'invio di merci;
- L'**Indirizzo del Destinatario** dell'invio di merce.

Queste sono le informazioni che devono essere presenti nell'invio, perché senza di queste l'Ordine perderebbe di significato.

Inoltre, a differenza del "Trasferimento" e del "Riassortimento", quando viene preso in carico un Ordine[G], con i dati sopra citati, quest'ultimo deve essere convalidato **se e solo se** sia presente la quantità richiesta di merce (*Good*[G]) dall'Ordine. In caso negativo, l'Ordine verrà cancellato interamente e dovrà essere ripetuto nuovamente dall'Utente. Infatti, solamente quando l'Utente confermerà l'Ordine precedentemente inserito, quest'ultimo verrà eseguito e preso in carico dal Sistema.

2.5 - Riassortimenti

Nel *Minimum Viable Product*[G], che *ALimitedGroup* ha progettato ed implementato, e nei documenti annessi viene fatto riferimento ad una serie di termini, uno tra questi è: il **Riassortimento**.

Viene subito da porsi la seguente domanda: *Cosa si intende per Riassortimento? E cosa ALimitedGroup intende con Riassortimento?*

Con il termine "Riassortimento" si intende il processo per cui, tramite il Sistema, si viene a conoscenza della scarsità di una determinata merce (*Good*[G]) in uno specifico magazzino (*Warehouse*[G]), ovvero che la quantità disponibile alla vendita di quella merce è sottostante alla soglia minima impostata dall'Admin Globale. Questo provoca, sempre tramite il Sistema, la notifica all'Admin Globale e successivamente il suggerimento di aumentare la quantità (*Stock*[G]) disponibile, usufruendo del servizio di trasferimento[G] da parte del magazzino più

vicino che dispone di un surplus della merce in questione, favorendo l'approvvigionamento di quest'ultima e l'ammortamento dei costi.

A differenza dell'Ordine e del Trasferimento[G], l'operazione di "Riassortimento" delle merci avviene **se e solo se** viene notificato, come detto in precedenza, il superamento della soglia minima e, conseguentemente, l'abbassamento della quantità globale di una determinata merce.

2.6 - Trasferimenti

Nel *Minimum Viable Product*[G], che *ALimitedGroup* ha progettato ed implementato, e nei documenti annessi viene fatto riferimento ad una serie di termini, uno tra questi è: il **Trasferimento**[G].

Un "Trasferimento" viene inteso come un qualsiasi scambio di merci, bilateralmente, tra due magazzini; questo scambio deve avere le seguenti informazioni al suo interno:

- Un **Identificativo Unico** che rappresenta questo scambio;
- La **Quantità** di merce che viene scambiata da parte del mittente;
- L'**Indirizzo del Mittente** dello scambio di merci;
- L'**Indirizzo del Destinatario** dello scambio di merce.

A differenza dell'Ordine e del "Riassortimento", nella fase di "Trasferimento" la quantità necessaria per lo scambio è già presente nel magazzino (*warehouse*[G]) e quindi l'"Ordine", antecedente a questa fase, è stato confermato ed è in fase di elaborazione in questa fase di "Trasferimento" .

3 - Test

La realizzazione del Sistema è proceduta di pari passo con la realizzazione e l'esecuzione di test[G]. Qui sarà brevemente spiegato come eseguire tali test[G].

Per informazioni approfondite si consiglia la lettura del **Piano di Qualifica[G] v.2.0.0**.

3.1 - Esecuzione dei *test* di unità ed integrazione

Per eseguire i test di unità ed integrazione è sufficiente aprire un *prompt*/*shell* dei comandi nella radice dell'installazione ed eseguire questi due comandi:

```
go generate ./... go test -race -covermode atomic ./...
```

L'output sarà simile al seguente:

```
? github.com/alimitedgroup/MVP/common [no test files]
        github.com/alimitedgroup/MVP/common/dto coverage: 0.0% of statements
? github.com/alimitedgroup/MVP/common/dto/request [no test files]
? github.com/alimitedgroup/MVP/common/dto/response [no test files]
? github.com/alimitedgroup/MVP/common/lib [no test files]
        github.com/alimitedgroup/MVP/common/lib/broker coverage: 0.0% of statements
        github.com/alimitedgroup/MVP/common/lib/observability coverage: 0.0% of
statements
? github.com/alimitedgroup/MVP/common/stream [no test files]
        github.com/alimitedgroup/MVP/srv/api_gateway coverage: 0.0% of statements
ok github.com/alimitedgroup/MVP/srv/api_gateway/adapterin

[...]

ok github.com/alimitedgroup/MVP/srv/warehouse/adapter/sender 1.219s coverage: 92.6%
of statements
? github.com/alimitedgroup/MVP/srv/warehouse/adapter/stream [no test files]
ok github.com/alimitedgroup/MVP/srv/warehouse/business 1.158s coverage: 77.4% of
statements
? github.com/alimitedgroup/MVP/srv/warehouse/business/model [no test files]
? github.com/alimitedgroup/MVP/srv/warehouse/business/port [no test files]
        github.com/alimitedgroup/MVP/srv/warehouse/config coverage: 0.0% of
statements
```

Dove il valore iniziale di ogni riga indica:

- `?`: nella cartella non sono presenti test[G]
- `ok`: i test[G] nella cartella che segue sono stati superati.

Un output non simile a questo indica il fallimento di uno dei test[G] presenti.

3.2 - Esecuzione dei *test* di accettazione

I test[G] di accettazione sfruttano `curl`, `nats`[G] e `jq` per poter essere eseguiti e gli script utilizzati possono essere trovati all'interno della cartella `tests`.

Alcuni test[G] richiedono una verifica[G] manuale sulla *dashboard*[G] presente su Grafana:[G] questa viene automaticamente configurata all'avvio, ma, per qualsiasi informazioni, dalla configurazione al primo accesso, è disponibile nel presente manuale.

I test[G] sono disponibili all'interno della cartella `tests` nella radice dell'installazione: è sufficiente aprire un prompt[G] dei comandi (o *shell*) in questa cartella ed eseguire gli script.

4 - Installazione

4.1 - Requisiti

4.1.1 - Requisiti Hardware

In merito agli applicativi sviluppati da *ALimitedGroup*, questi sono relativamente recenti e possono essere eseguiti senza particolari esigenze anche su macchine con comparto tecnico non particolarmente elevato.

A scopo esemplificativo, il *Software* funzionava correttamente su una macchina con le seguenti caratteristiche:

- *CPU* : 3.400GHz;
- *GPU* : Intel UHD Graphics 620
- *RAM*: 8GB di memoria;

Le caratteristiche *Hardware* minime sono dunque più legate all'utilizzo della macchina in questione come server NATS[G] e/o con dei microservizi, i quali è caldamente consigliato siano avviati con Docker[G] .

A tale scopo si rimanda la lettura della documentazione ufficiale di Docker e di NATS

4.1.2 - Requisiti Software

In merito al Sistema Operativo, le componenti del Sistema non fanno distinzione in quanto il Sistema si appoggia completamente a **Docker**[G] per garantire la sua esecuzione e le immagini utilizzate provengono direttamente dai *repository*[G] di Docker[G] o vengono compilate mediante il compilatore di Go[G] scaricato anch'esso dai *repository*[G] di Docker[G].

Per assicurare dunque il corretto funzionamento delle componenti del Sistema è dunque necessario sia installato nella macchina il *Software* Docker[G].

Per facilitare l'avvio del Sistema è possibile utilizzare l'applicativo **Just**, reperibile all'indirizzo https://github.com/casey/just: si tratta di una Utility in grado di eseguire, mediante un semplice comando, tutti i comandi a loro volta necessari per l'avvio del Sistema. Il suo utilizzo resta tuttavia completamente facoltativo in quanto è pur sempre possibile inserire manualmente i comandi. I comandi che **Just** esegue sono reperibili nel file `JustFile`.

La maggior parte di quanto necessario per compilare e avviare il Sistema viene automaticamento eseguito da `Docker`. Per eseguire tutte le operazioni possibili è dunque necessario avere un'installazione funzionante dei seguenti applicativi:

Software	Versione
Docker Engine	28.0.1
Just	1.39.0
NATS	2.10.25

Tabella 2: Elenco dei *Software* utilizzati e loro versioni

4.2 - Preparazione del Sistema

4.2.1 - Installazione del *Software*

Anzitutto procedere al reperimento dell'ultima *release* del Software disponibile per il *download* alla pagina Github del progetto. Le release si trovano nell'apposita sezione nella colonna a destra della pagina, come mostrato nella seguente immagine:

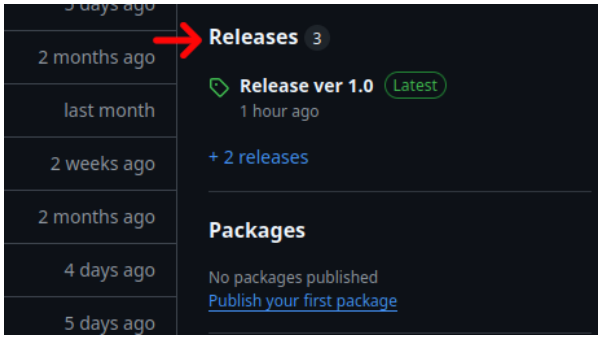

Figura 1: Selezione della sezione Release

Quindi selezionare la release di interesse (nel momento in cui questo manuale viene scritto, la 1.0):

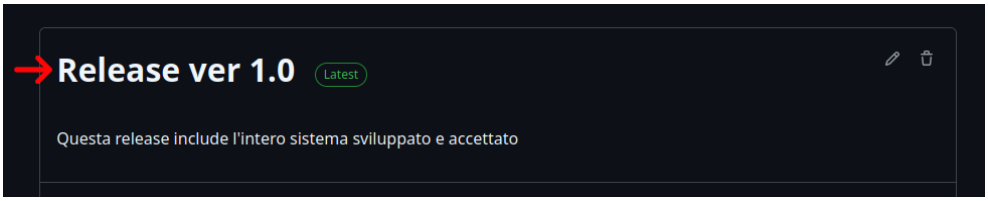

Figura 2: Selezione della Release

E procede a scaricare l'archivio compresso contenente il codice sorgente:

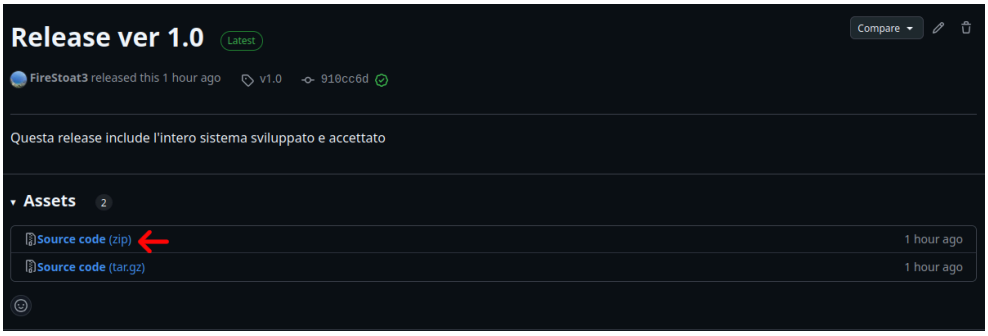

Figura 3: Download dell'archivio compresso

Scompattare l'archivio compresso in una cartella a propria scelta, quindi aprire un terminale nella posizione corrente.

Il Sistema è adesso pronto per poter essere avviato.

4.2.2 - Personalizzazione della configurazione

4.2.2.1 - Informazioni generali

Il Sistema è stato realizzato per poterne configurare a proprio piacimento i servizi in esecuzione in una stessa macchina.

Per iniziare aprire il file `compose.yml`, file `yaml` contenente i servizi che saranno avviati mediante Docker.[G] I seguenti servizi sono obbligatori per l'esecuzione di qualsiasi altro microservizio:

- **collector**
- **loki**
- **prometheus**.

Questi tre servizi offrono le funzionalità[G] telemetriche e non possono perciò essere rimossi.

Il servizio `nats`[G] avvia un'istanza di un server NATS:[G] se rimosso è bene eliminare tale servizio dal campo `depends-on` presente in alcuni altri servizi.

I servizi che possono essere aggiunti e rimossi a piacimento sono:

- **api-gateway**: servizio che avvia un'istanza del microservizio omonimo;
- **warehouse-1**: servizio che avvia un'istanza del microservizio warehouse.[G] Dal momento che il nome del servizio è non vincolante, il nome può essere cambiato in base alle proprie esigenze (per garantire l'utilizzo offline è consigliato unire, ad un'istanza di warehouse[G], un'istanza di api[G] gateway e una di nats[G] sulla stessa macchina);
- **catalog**: servizio che avvia un'istanza del microservizio omonimo;
- **order**: servizio che avvia un'istanza del microservizio omonimo, che gestisce gli ordini e i trasferimenti;
- **authenticator**: servizio che avvia un'istanza del microservizio omonimo;
- **notification**: servizio che avvia un'istanza del microservizio omonimo;
- **Grafana[G]**: servizio per visualizzare la telemetria con una comoda *Graphic User Interface*.

Tutti i servizi appena elencati possiedono tre variabili d'ambiente configurabili in base alle proprie esigenze, presenti sotto la voce `environment`:

- **ENV_BROKER_URL**: l'indirizzo di un'istanza NATS[G] (default: `nats://nats:4222`);
- **ENV_SERVICE_ID**: l'identificativo del servizio;
- **OTLP_URL**: l'indirizzo di un'istanza di collector (default: `collector:4317`);

IMPORTANTE

Prestare attenzione alla configurazione di **ENV_SERVICE_ID**: l'impostazione di due o più istanze con stesso identificativo **IMPEDIRÀ** la distinzione dei dati telemetrici di un'istanza dalle altre. Se omesso, il Sistema genererà un identificativo causale, tuttavia questo sarà resettato al riavvio dell'istanza, determinandone un cambiamento: tale modo d'impiego è dunque **caldamente sconsigliato**.
La generazione automatica non è abilitata per i magazzini, che non entreranno in esecuzione se la variabile d'ambiente non è impostata.

Alcuni dei servizi sviluppati presentano la seguente dicitura:

```
build: { args: { SERVICE: valore } }
```

questo significa che il servizio verrà compilato dai file localmente disponibili. Il campo `valore` deve essere sostituito con il nome della cartella dove si trovano i file da compilare per quel microservizio (ad esempio, scrivere `api_gateway` per compilare e avviare un servizio che esegue Api gateway).

Nel caso un servizio richiederà ulteriori configurazioni, questo sarà ora descritto.

4.2.2.2 - Api gateway

Possiede queste ulteriori variabili d'ambiente:

- **ENV_API_PORT**: porta API (default: `8080`);
- **HTTP_HOST**: host del Sistema (default: `0.0.0.0`);
- **HTTP_PORT**: porta http (default: `8080`).

4.2.2.3 - Influxdb

Possiede queste ulteriori variabili d'ambiente:

- DOCKER_INFLUXDB_INIT_MODE: vedere la documentazione ufficiale (default: `setup`);
- DOCKER_INFLUXDB_INIT_USERNAME: vedere la documentazione ufficiale (default: `admin`);
- DOCKER_INFLUXDB_INIT_PASSWORD: vedere la documentazione ufficiale (default: `admin1234`);
- DOCKER_INFLUXDB_INIT_ORG: vedere la documentazione ufficiale (default: `my-org`);
- DOCKER_INFLUXDB_INIT_BUCKET: vedere la documentazione ufficiale (default: `stockdb`);
- DOCKER_INFLUXDB_INIT_ADMIN_TOKEN: vedere la documentazione ufficiale (default: `my-token`).

4.2.2.4 - Notifications

Possiede queste ulteriori variabili d'ambiente:

- INFLUXDB_TOKEN: stesso valore inserito al servizio influxdb (default: `my-token`);
- INFLUXDB_ORG: stesso valore inserito al servizio influxdb (default: `my-org`);
- INFLUXDB_URL: Indirizzo del servizio Influxdb (default: `http://influxdb:8086`)
- INFLUXDB_BUCKET: stesso valore inserito al servizio influxdb (default: `stockdb`);
- RULE_CHECKER_TIMER: il valore impostato determinerà ogni quanto tempo vengono controllate le soglie e le quantità globali disponibili di una merce per inviare o meno una notifica (default `5s`).

Affinché Notifications funzioni correttamente è necessario sia attiva un'istanza del microservizio catalog: qualora non sia presente il microservizio non riuscirà a completare la procedura di avvio.

Se nello stesso file `compose.yml` non è presente il microservizio catalog, eliminare la dicitura `catalog` dalle voci elencate su `depends_on`, discorso analogo se Influxdb è in esecuzione su un'altra macchina.

4.2.3 - Avviare il Sistema configurato

4.2.3.1 - Avvio

Una volta configurato il Sistema l'avvio risulta essere molto semplice.

È sufficiente eseguire il comando:

```
just up
```

o, se si è deciso di non utilizzare **Just**, è possibile limitarsi all'esecuzione del seguente comando **Docker**[G]:

```
docker compose up -d --build
```

che avvierà quanto descritto nel file `compose.yml` in *background*.

Per avviare i servizi non in background è possibile eseguire:

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

Se l'istanza era già stata avviata e non era stata resettata, al prossimo avvio i dati verranno conservati.

Notare che il primo avvio potrebbe essere notevolmente più lungo rispetto ad un riavvio.

4.2.3.2 - Spegnimento

Per arrestare i servizi avviati è possibile eseguire il comando:

```
just down
```

o, se si è deciso di non utilizzare **Just**, è possibile limitarsi all'esecuzione del seguente comando **Docker**[G]:

```
docker compose down
```

4.2.3.3 - Ripristino

A Sistema avviato, eseguire:

```
just reset
```

o, se si è deciso di non utilizzare **Just**, è possibile limitarsi all'esecuzione del seguente comando **Docker**[G]:

```
docker compose down -v --remove-orphans && docker compose up -d --build
```

I vari servizi saranno spenti, ripristinati e riavviati.

5 - Guide

5.1 - Login

Il Sistema possiede un Sistema di autenticazione mediante token: per poter accedere al Sistema è necessario avere un token valido.

Nella sua forma attuale, il Sistema permette di ottenere un token indicando il ruolo desiderato.

Per ottenere il token è sufficiente effettuare la seguente richiesta:

```
curl -Ss -X POST "http://localhost:8080/api/v1/login" -d username=valore
```

Dove, al posto di `valore` è possibile inserire uno dei seguenti valori:

- **global_admin**, per ottenere un token per il ruolo di **Admin Globale**
- **local_admin**, per ottenere un token per il ruolo di **Admin Locale**
- **client**, per ottenere un token per il ruolo di **Client**.

Per verificare la validità del token, è possibile salvare il token ricevuto in una variabile `TOKEN` eseguendo questo comando:

```
TOKEN=$(curl -Ss -X POST "http://localhost:8080/api/v1/login" -d username=valore | jq
-r '.token')
```

Quindi è sufficiente creare un *header* mediante questo comando:

```
PARAMS=(-sS -H "Authorization: Bearer $TOKEN" -H "Content-Type: application/json")
```

E, infine, eseguire la richiesta di verifica:

```
curl "${PARAMS[@]}" -X GET "http://localhost:8080/api/v1/is_logged"
```

Se il processo è andato a buon fine, la risposta conterrà il ruolo scelto.

5.2 - Aggiungere una merce

In questa sezione sarà spiegato come aggiungere una merce interagendo con il Sistema mediante l'utilizzo di `curl` e `jq`.

La merce aggiunta avrà le seguenti caratteristiche:

- **id**: hat-1;
- **nome**: hat;
- **descrizione**: blue hat.

Anzitutto è necessario ottenere un token per il ruolo di **Admin Globale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `GA_TOKEN`.

Preparare e salvare in una variabile `GA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi procedere all'aggiunta della merce eseguendo il seguente comando:

```
curl "${GA_PARAMS[@]}" -X PUT "http://localhost:8080/api/v1/goods/hat-1" \   -d
'{"name":"hat","description":"blue hat"}'
```

Operazione completata: la merce è stata aggiunta con successo.

Si noti che il comando eseguito ha manualmente assegnato l'id alla merce, tuttavia questo non è necessario: si consiglia la lettura degli *endpoint* disponibili.

Per verificare le informazioni sulle merci del Sistema si consiglia la lettura della Sezione dedicata.

5.3 - Aggiungere *stock* ad una merce

In questa Sezione sarà spiegato come aggiungere *stock*[G] ad una merce interagendo con il Sistema mediante l'utilizzo di `curl`.

Seguendo la guida saranno aggiunte **6 unità** alla merce con **id** `hat-1` al magazzino con **id** `1`.

Anzitutto è necessario ottenere un token per il ruolo di **Admin Locale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `LA_TOKEN`. Considererà inoltre creata la merce con id `hat-1`: per maggiori informazioni si consiglia la lettura della Sezione dedicata alla creazione di una merce.

Preparare e salvare in una variabile `LA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
LA_PARAMS=(-sS -H "Authorization: Bearer $LA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi procedere all'aggiunta dello *stock*[G] mediante il seguente comando:

```
curl "${LA_PARAMS[@]}" -X POST "http://localhost:8080/api/v1/goods/hat-1/warehouse/1/
stock" \   -d '{"quantity": 6}'
```

Operazione completata: lo *stock*[G] è stato aggiunto con successo.

Per verificare le informazioni sulle merci del Sistema si consiglia la lettura della Sezione dedicata.

5.4 - Rimuovere *stock* di una merce

In questa Sezione sarà spiegato come rimuovere *stock*[G] ad una merce interagendo con il Sistema mediante l'utilizzo di `curl`.

Seguendo la guida saranno rimosse **5 unità** alla merce con **id** `hat-1` dal magazzino con **id** `1`.

Anzitutto è necessario ottenere un token per il ruolo di **Admin Locale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `LA_TOKEN`. Considererà inoltre creata la merce con id `hat-1` e con *stock*[G] sufficiente per effettuarne la rimozione: per maggiori informazioni si consiglia la lettura della Sezione dedicata alla creazione di una merce e all'aggiunta di *stock*[G].

Preparare e salvare in una variabile `LA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
LA_PARAMS=(-sS -H "Authorization: Bearer $LA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi procedere alla rimozione dello *stock*[G] mediante il seguente comando:

```
curl "${LA_PARAMS[@]}" -X DELETE "http://localhost:8080/api/v1/goods/hat-1/warehouse/
1/stock" \   -d '{"quantity": 5}'
```

Operazione completata: lo *stock*[G] è stato rimosso con successo.

Per verificare le informazioni sulle merci del Sistema si consiglia la lettura della Sezione dedicata.

5.5 - Visualizzare le merci nel Sistema

Il Sistema permette l'ottenimento della lista delle merci presenti nel Sistema.

Anzitutto è necessario ottenere un token per il ruolo di **Cliente**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `C_TOKEN`.

Preparare e salvare in una variabile `C_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi ottenere una lista delle merci eseguendo il seguente comando:

```
curl "${C_PARAMS[@]}" -X GET "http://localhost:8080/api/v1/goods"
```

5.6 - Visualizzare i magazzini presenti nel Sistema

Il Sistema permette di reperire una lista di tutti i magazzini presenti nel Sistema e che hanno subito almeno un'aggiunta di *stock*[G].

Anzitutto è necessario ottenere un token per il ruolo di **Admin Globale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `GA_TOKEN`.

Preparare e salvare in una variabile `GA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi ottenere i magazzini con la seguente richiesta:

```
curl "${GA_PARAMS[@]}" -X GET "http://localhost:8080/api/v1/warehouses"
```

5.7 - Effettuare un ordine

Il Sistema permette la realizzazione di ordini: la seguente guida illustrerà come realizzarne uno, presupponendo che ci sia sufficiente quantità per poterlo portare a termine.

Anzitutto è necessario ottenere un token per il ruolo di **Client**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `C_TOKEN`.

Preparare e salvare in una variabile `C_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
C_PARAMS=(-sS -H "Authorization: Bearer $C_TOKEN" -H "Content-Type: application/
json")
```

È possibile procedere con la creazione dell'ordine mediante il seguente comando:

```
curl "${C_PARAMS[@]}" -X POST "http://localhost:8080/api/v1/orders" \   -d '{"name":
"order-1", "full_name": "John Doe", "address": "via roma 12 35012", "goods":
{"hat-1": 7}}'
```

La richiesta eseguita cercherà di portare a termine un ordine[G] dal nome **order-1** per il cliente **John Doe**, residente in **via roma 12 35012**, di **7 unità** della merce con **id `hat-1`**

È ovviamente possibile includere più merci inserendole in una lista JSON.

Per verificare l'esito dell'ordine è consigliata la lettura della Sezione relativa alla visione degli ordini.

5.8 - Visualizzare gli ordini eseguiti

Il Sistema permette di ottenere una lista degli ordini eseguiti.

Anzitutto è necessario ottenere un token per il ruolo di **Cliente**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `C_TOKEN`.

Preparare e salvare in una variabile `C_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
C_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi ottenere la lista degli ordini eseguendo il comando seguente:

```
curl "${C_PARAMS[@]}" -X GET "http://localhost:8080/api/v1/orders".
```

Si noti che gli ordini possono avere quattro stati:

- **Created**: ordine[G] creato, in attesa di iniziare le *reservation*
- **Filled**: tutte le *reservation* sono riuscite, è in corso la rimozione degli *stock*[G]
- **Completed**: ordine[G] completato;
- **Cancelled**: una o più *reservation* non sono riuscite e non è più presente merce a sufficienza per completare l'ordine.

5.9 - Effettuare un trasferimento

Il Sistema permette di realizzare trasferimenti tra magazzini. La seguente guida illustrerà come realizzarne uno, presupponendo l'esistenza di due magazzini e di stock[G] sufficiente nel primo magazzion: per maggiori informazioni visitare la pagina di installazione del presente Sistema e la Sezione dedicata all'aggiunta di *stock*[G].

Anzitutto è necessario ottenere un token per il ruolo di **Admin Globale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `GA_TOKEN`.

Preparare e salvare in una variabile `GA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

Si può quindi procedere a realizzare il trasferimento[G] mediante la seguente richiesta:

```
curl "${GA_PARAMS[@]}" -X POST "http://localhost:8080/api/v1/transfers" \   -d
'{"receiver_id": "id_magazzino_destinatario", "sender_id": "id_magazzino_mittente",
"goods": {"id_merce": qta}}'
```

Cambiando i segnaposto seguenti con i valori adeguati:

- **id_magazzino_destinatario**: identificativo del magazzino destinatario del trasferimento[G]
- **id_magazzino_mittente**: identificativo del magazzino mittente del trasferimento[G]
- **id_merce**: identificativo della merce da trasferire;
- **qta**: quantità della merce da trasferire.

È ovviamente possibile includere più merci inserendole in una lista *JSON*.

Per verificare l'esito del trasferimento[G] è consigliata la lettura della Sezione relativa alla visione dei trasferimenti.

5.10 - Visualizzare i trasferimenti eseguiti

Il Sistema permette di ottenere una lista dei trasferimenti eseguiti.

Anzitutto è necessario ottenere un token per il ruolo di **Admin Globale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `GA_TOKEN`.

Preparare e salvare in una variabile `GA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile quindi ottenere la lista dei trasferimenti eseguendo il comando seguente:

```
curl "${GA_PARAMS[@]}" -X GET "http://localhost:8080/api/v1/transfers".
```

5.11 - Impostare una soglia di allerta

Il Sistema permette di aggiungere una soglia di allerta per ricevere notifiche quando le scorte globalmente disponibile di una merce scendono sotto una certa soglia.

Anzitutto è necessario ottenere un token per il ruolo di **Admin Globale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `GA_TOKEN`.

Preparare e salvare in una variabile `GA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile procedere alla creazione di una soglia di allerta mediante il seguente comando:

```
curl "${GA_PARAMS[@]}" -X POST localhost:8080/api/v1/notifications/queries \   -d
'{"good_id": "id_merce", "operator": "<", "threshold": soglia}'
```

dove:

- al posto di **id_merce** è necessario inserire l'id della merce da tenere sotto osservazione;
- al posto di **soglia** è necessario inserire la soglia sotto cui notificare l'allerta.

La query avrà subito effetto: per verificare le notifiche in arrivo vedere la Sezione dedicata di questo manuale.

5.12 - Visualizzare le soglie impostate

Il Sistema permette di ottenere una lista delle soglie di allerta impostate.

Anzitutto è necessario ottenere un token per il ruolo di **Admin Globale**: per maggiori informazioni vedere la Sezione dedicata al login.

La guida considererà ottenuto questo token e salvato in una variabile `GA_TOKEN`.

Preparare e salvare in una variabile `GA_PARAMS` l'header da inviare assieme alla richiesta con il seguente comando:

```
GA_PARAMS=(-sS -H "Authorization: Bearer $GA_TOKEN" -H "Content-Type: application/
json")
```

È possibile procedere all'ottenimento della lista mediante la seguente richiesta:

```
curl "${GA_PARAMS[@]}" -X GET localhost:8080/api/v1/notifications/queries
```

5.13 - Ricevere le notifiche

Il Sistema al momento non possiede un *endpoint* per poter recuperare le notifiche, ma è possibile utilizzare `nats`[G] per ascoltare i canali in cui vengono pubblicati.

Per farlo è sufficiente installare `nats`[G] (per istruzioni si consiglia la lettura del paragrafo dedicato nel Repository del progetto) ed eseguire il comando:

```
nats sub "stock.alert.>"
```

Ci si troverà quindi in ascolto di tutte le notifiche inviate dal Sistema.

6 - Telemetria

Questa guida spiegherà come configurare Grafana[G] per avere sotto osservazione tutte le misurazioni di natura telemetrica.

La guida suppone che l'accesso a Grafana[G] sia effettuato da una macchina che possiede in esecuzione un'istanza dell'omonimo servizio: vedere la Sezione dedicata all'installazione del Sistema.

L'avvio con i *file* sorgente non alterati dovrebbe completare automaticamente la configurazione ma questa guida illustrerà nel dettaglio tutti i passaggi.

6.1 - Primo accesso

Avviato il container con il servizio `Grafana` aprire un browser e collegarsi all'indirizzo `localhost:3000`. Apparirà la seguente schermata di login:

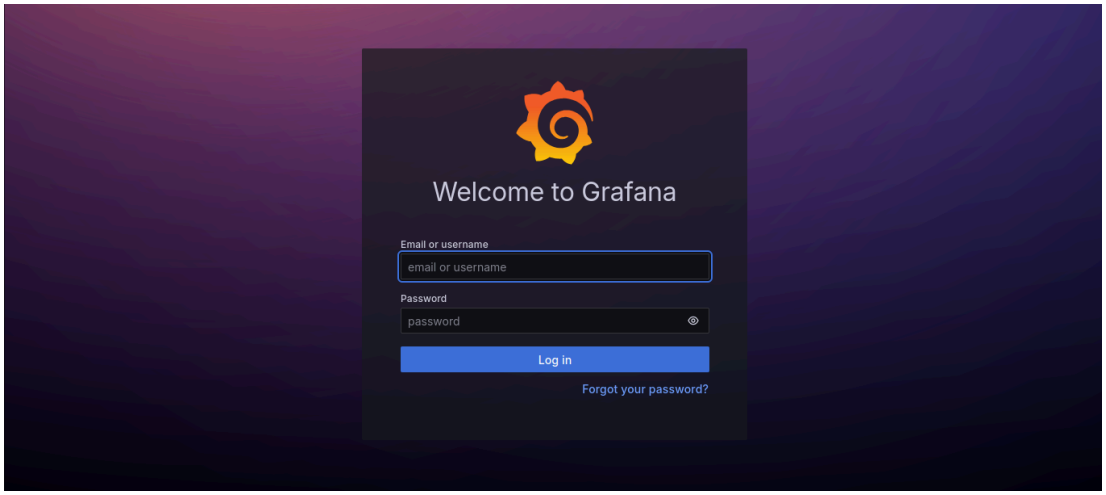

Figura 4: Pagina di login di Grafana

Inserire i dati di login di default:

- **Username**: `admin`
- **Password**: `admin`.

Si presenterà la seguente schermata: sebbene sia consigliato cambiare la password di default, al momento premeremo "Skip" per procedere con la guida.

Figura 5: Pagina di richiesta cambio password di Grafana

Login effettuato.

6.2 - Aggiungere connessione a Prometheus

Selezionare, nella colonna di sinistra, la voce **Connections**, quindi **Data sources**. Apparirà la seguente schermata:

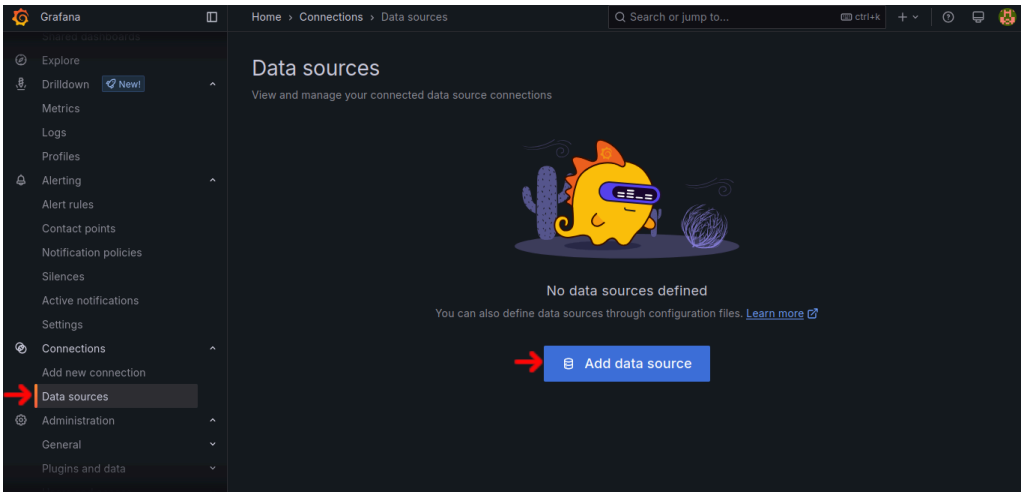

Figura 6: Pagina Data Sources di Grafana

Come indicato dalla figura 6, cliccare il bottone **Add data source**.

Nella schermata che appare, cercare **Prometheus** e selezionarlo, come mostrato in figura.

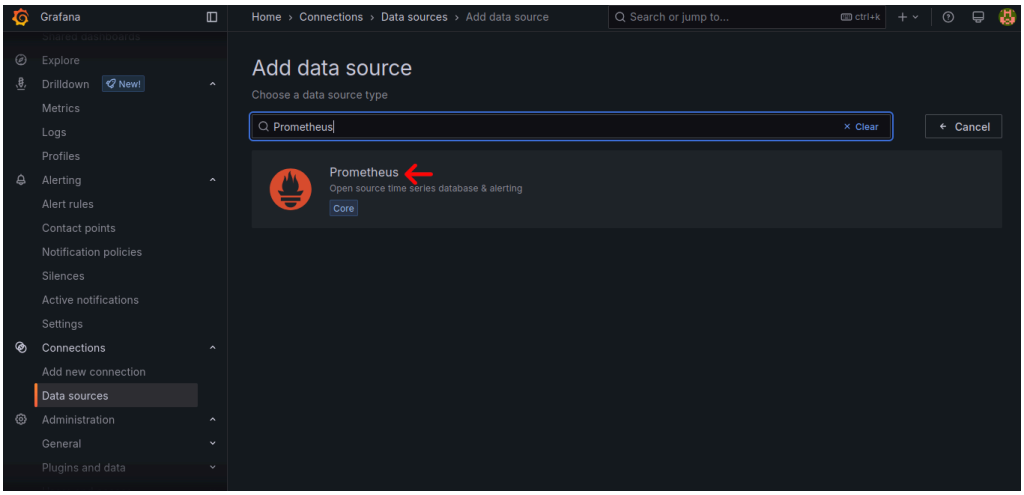

Figura 7: Pagina Data Sources di Grafana

Inserire l'indirizzo di Prometheus: se configurato localmente e con il file `compose.yml`, l'indirizzo è `http://prometheus:9090`.

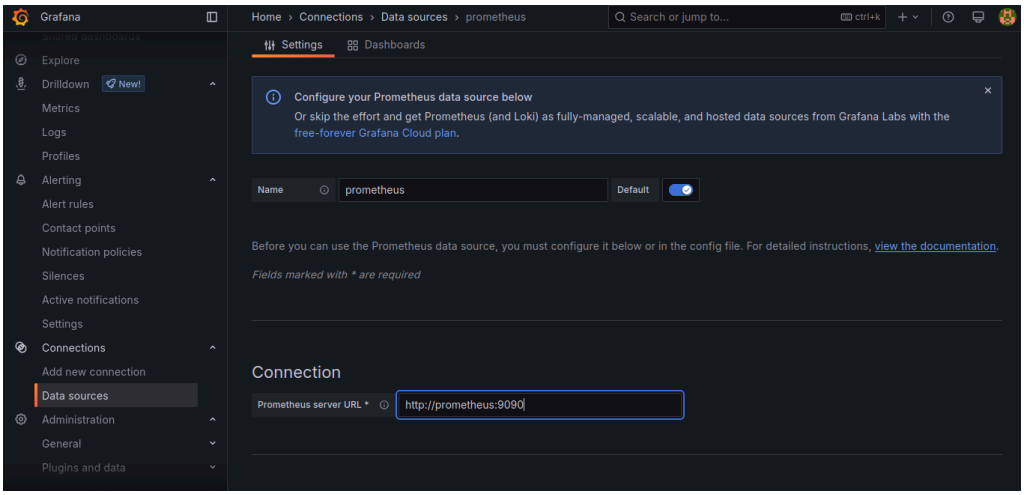

Figura 8: Inserimento indirizzo di Prometheus

Salvare quindi la configurazione.

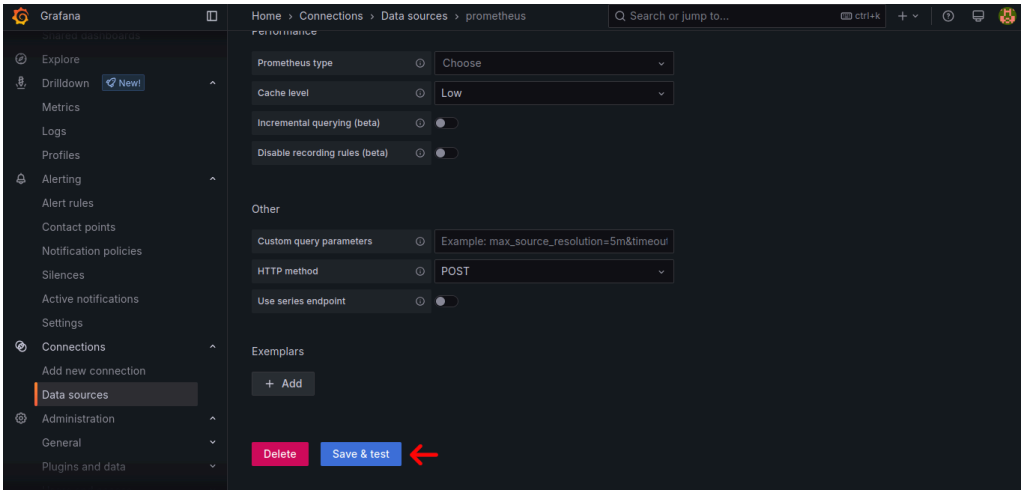

Figura 9: Salvataggio connessione a Prometheus

6.3 - Aggiungere connessione a Loki

Anzitutto, recarsi nuovamente in **Connections** > **Data sources**. Dopo l'aggiunta di **Prometheus**, la schermata sarà simile alla seguente:

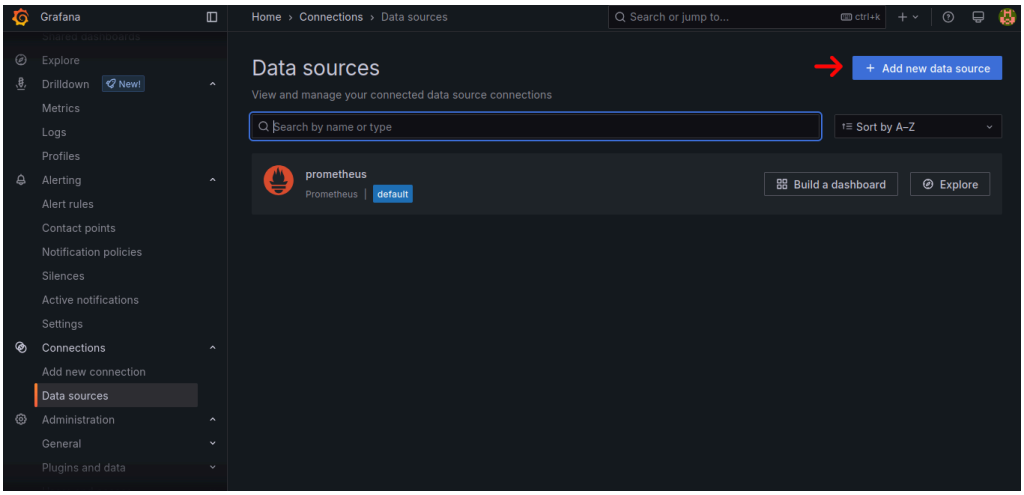

Figura 10: Aggiunta di una nuova sorgente dati

Cliccare, come mostrato in Figura 10, **Add new data source**.

A questo punto, procedere a cercare e selezionare **Loki**, come mostrato nella figura seguente:

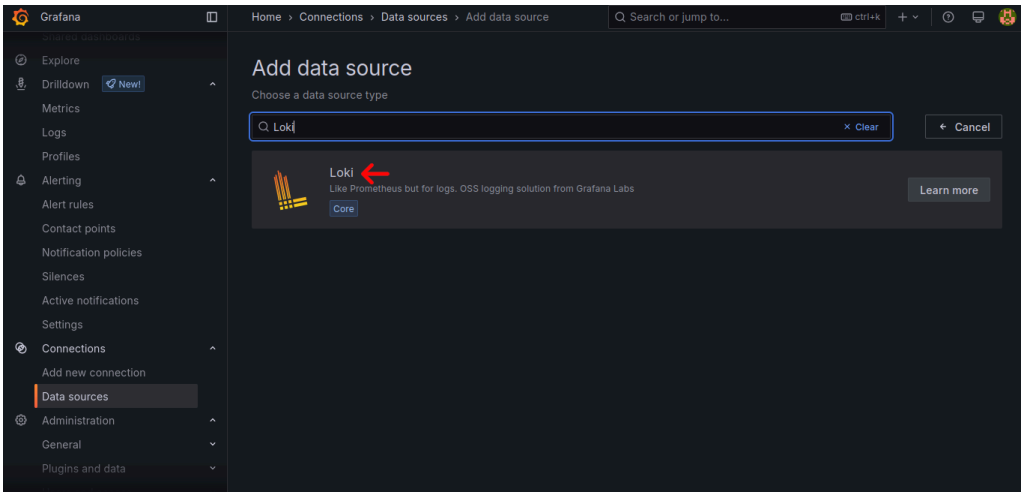

Figura 11: Selezione della sorgente dati Loki

Inserire l'indirizzo di Loki: se configurato localmente e con il file `compose.yml`, l'indirizzo è `http://loki:3100`.

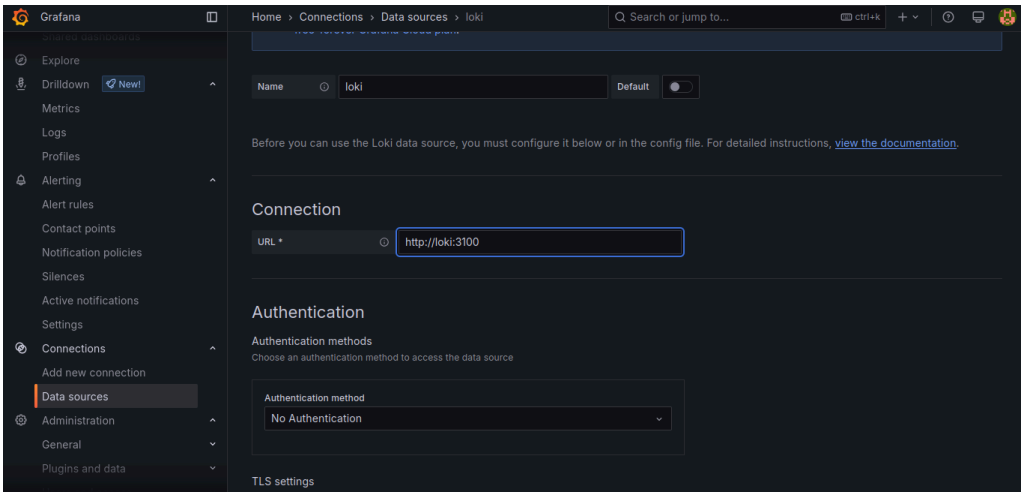

Figura 12: Inserimento indirizzo di Loki

Confermare quindi l'aggiunta come mostrato nella figura che segue:

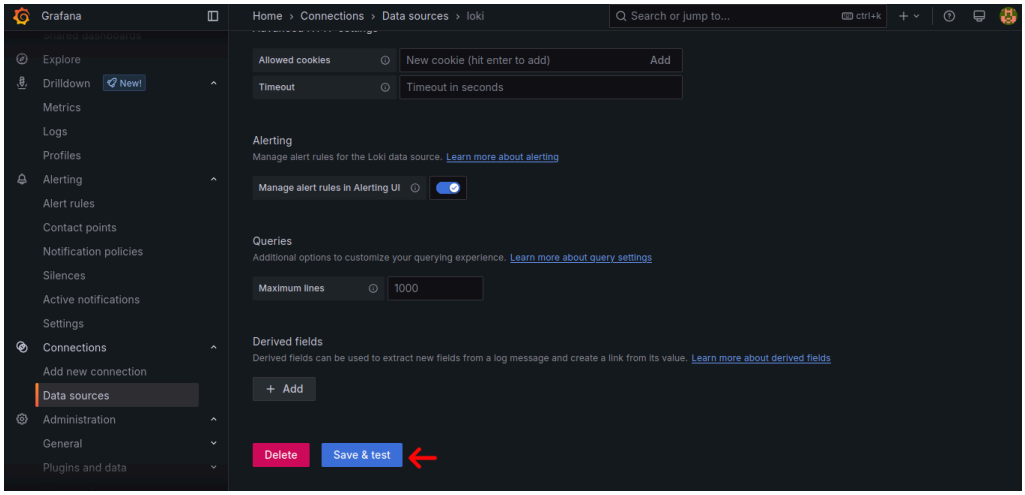

Figura 13: Salvataggio connessione a Loki

6.4 - Importare una dashboard

I file forniti includono una *dashboard*[G] configurata per avere a disposizione tutti i microservizi in esecuzione nella stessa macchina.

Tale *dashboard*[G] è automaticamente importata all'avvio del Sistema. Qualora non lo fosse, è possibile seguire questi passaggi.

Anzitutto recarsi nella sezione **Dashboards**[G] di Grafana[G], selezionando l'apposita opzione nella barra laterale sinistra, come mostrato in figura:

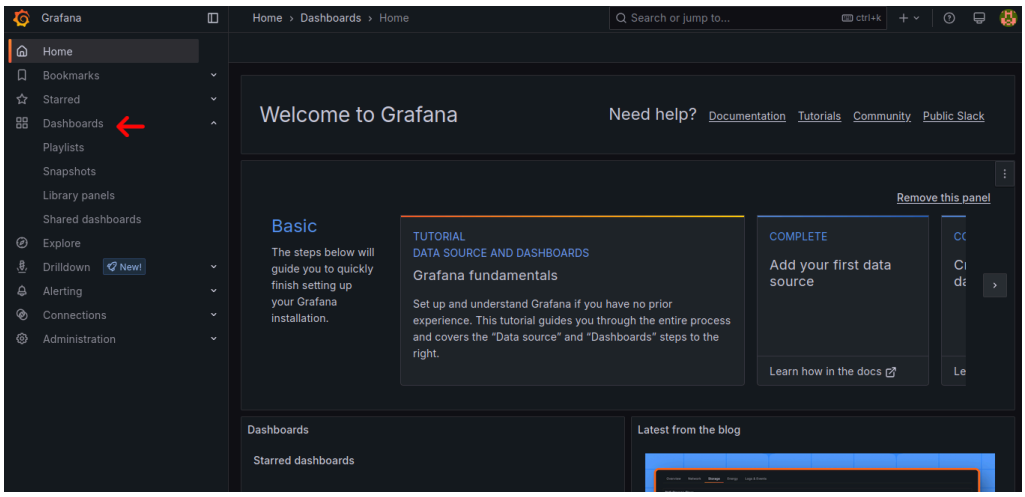

Figura 14: Selezione del menu delle dashboard

Procedere dunque ad importare la *dashboard* caricando l'apposito file *JSON* che è possibile trovare in `/containers/dashboards/json` della cartella di installazione. La pagina di importazione è simile alla figura che segue:

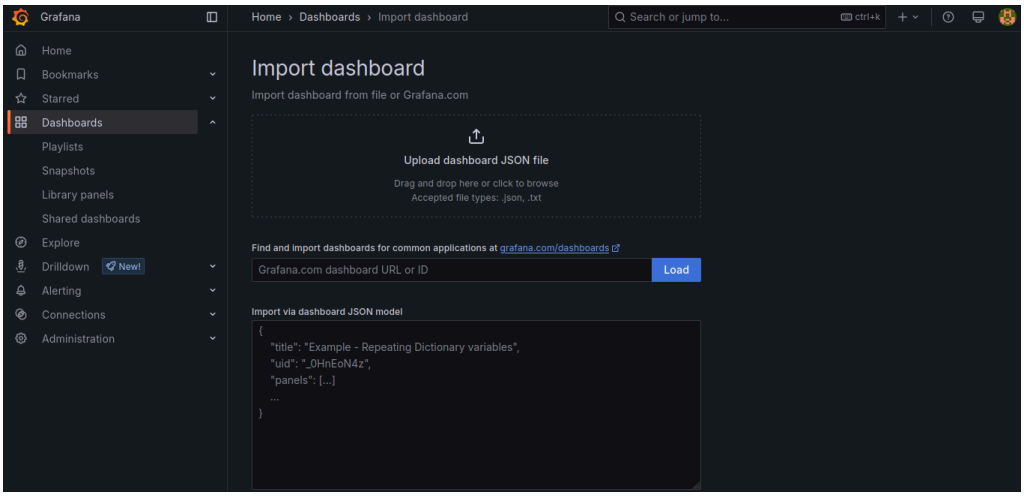

Figura 15: Importare una dashboard

6.5 - Modificare una dashboard

Selezionare anzitutto la *dashboard* appena importata come mostrato in figura:

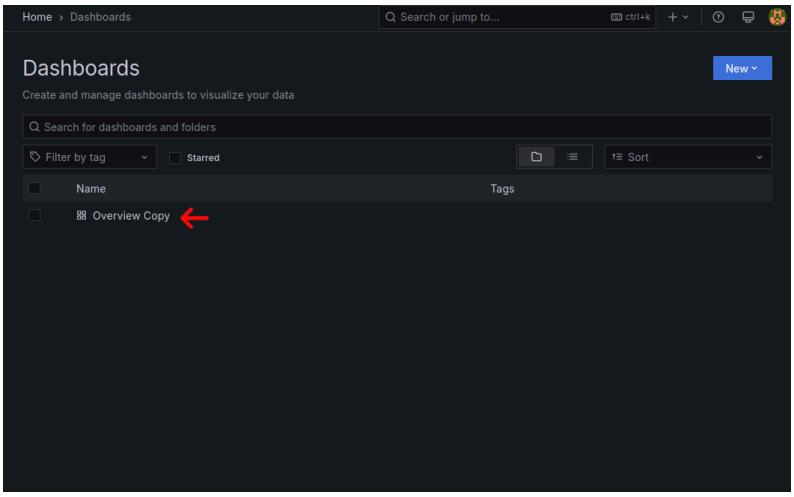

Figura 16: Selezionare una dashboard

A questo punto, è possibile aggiungere i log e le misurazioni del microservizio.

Attivare anzitutto la modifica della *dashboard* corrente premendo il tasto **Edit**, come mostrato nella figura che segue:

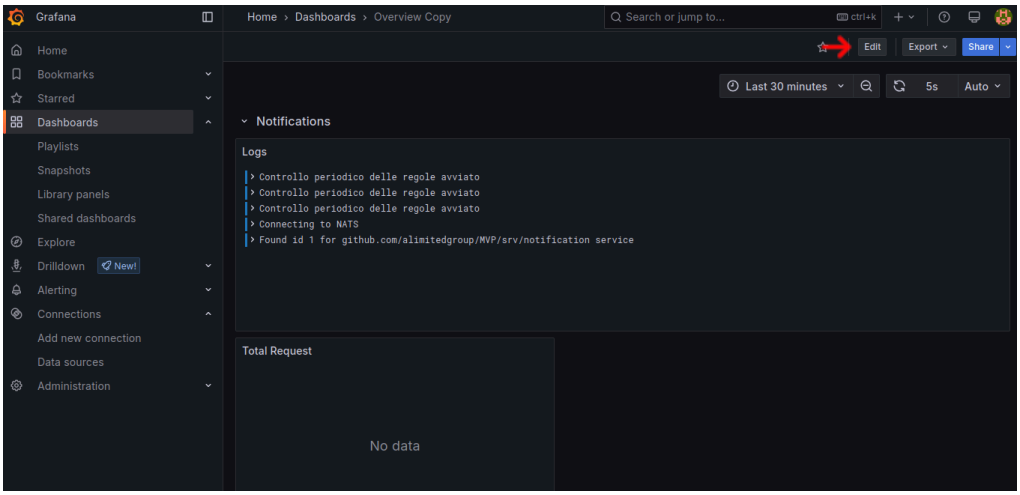

Figura 17: Selezionare del tasto di modifica

Premere quindi il tasto **Add** e quindi **Visualization**

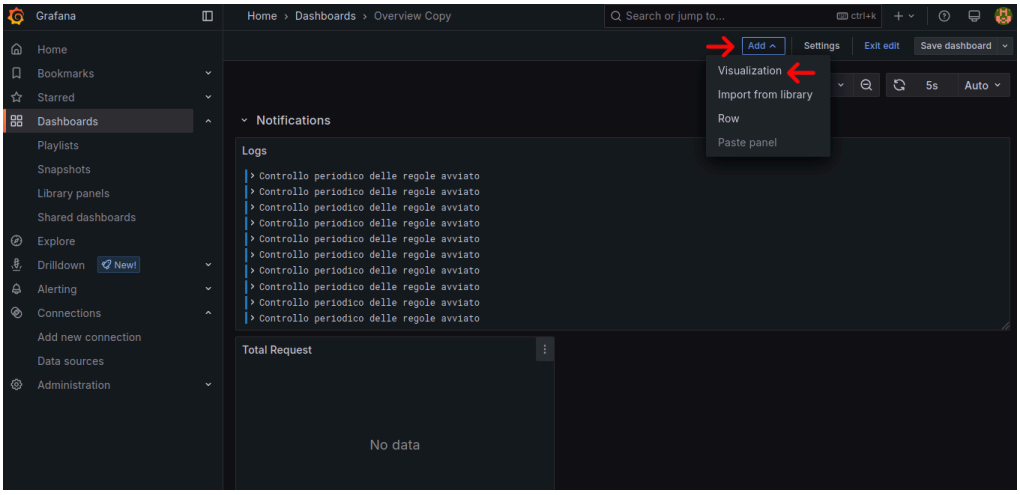

Figura 18: Selezione del tasto di aggiunta

A questo punto il processo differisce in base all'aggiunta di log e misurazioni, per cui si consiglia di seguire i paragrafi successivi.

6.5.1 - Aggiungere una finestra di log

Dalla finestra che si apre, selezionare, dal menu a tendina vicino la voce **Data source**, la voce **Loki**.

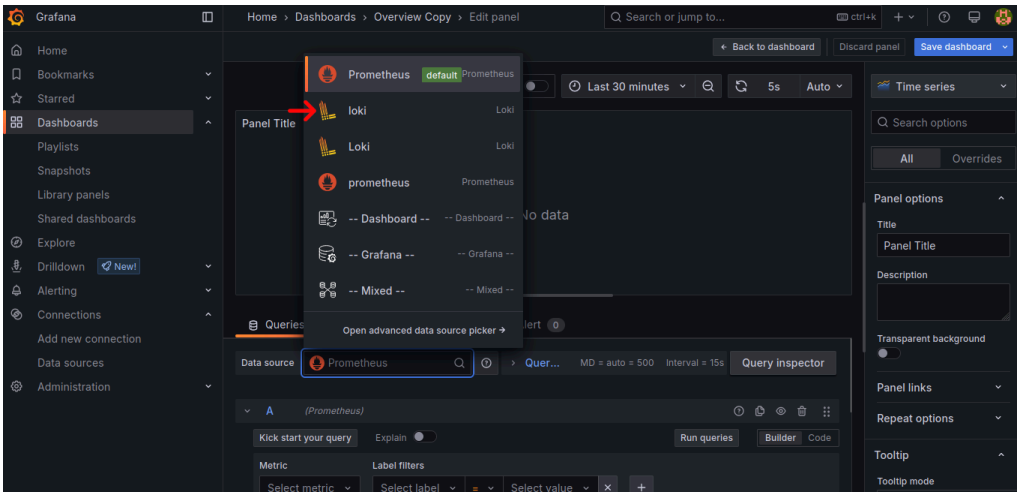

Figura 19: Selezionare di Loki come sorgente dati

Selezionare ora il menu a tendina indicato dalla figura che segue:

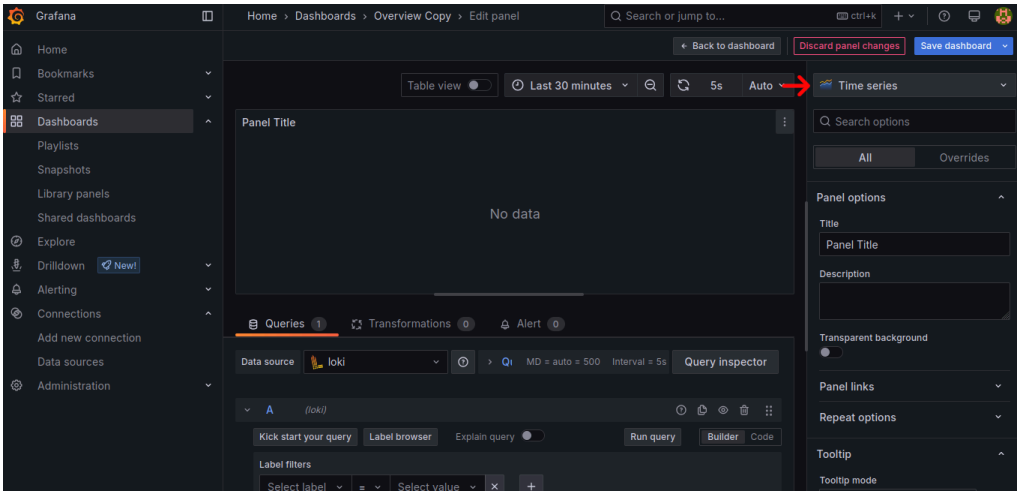

Figura 20: Selezione del tipo di dati

Cercare e selezionare **Logs**.

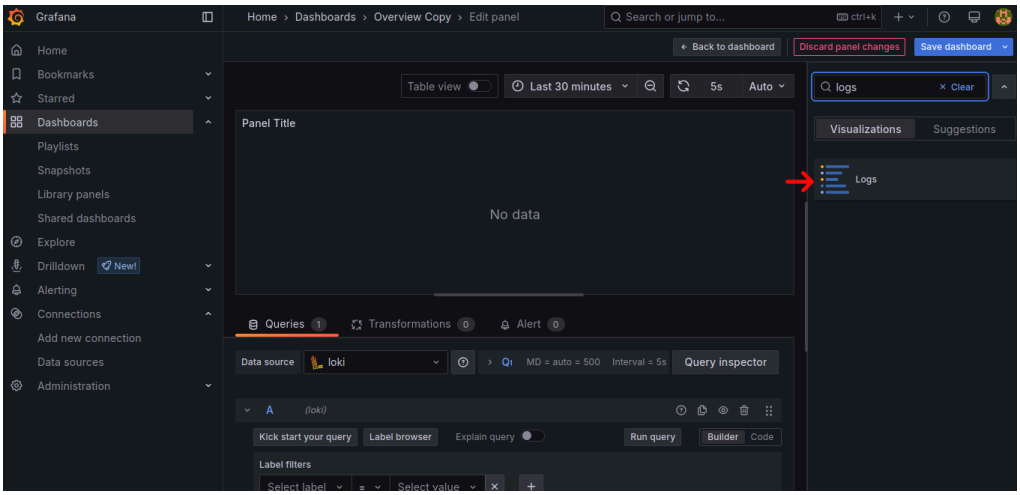

Figura 21: Selezione del tipo di dati

Dalla Sezione **Label filters**, selezionare la voce **Select label** e, dal menu a tendina, la voce **service_name**.

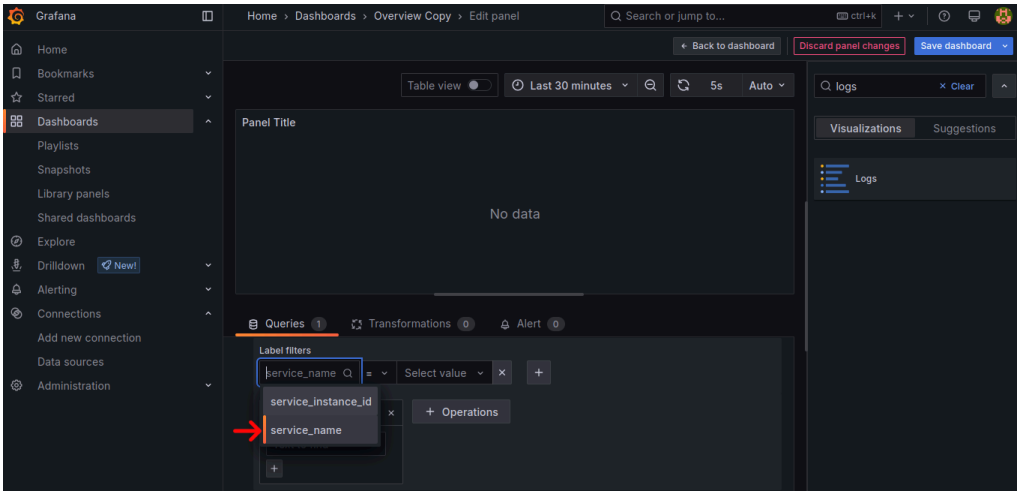

Figura 22: Selezione modalità individuazione servizio per log

Premere ora su **Select value**, quindi selezionare il servizio di interesse. Per una rapida individuazione, i nomi sono nella forma che segue:

`github.com/alimitedgroup/MVP/srv/nome_servizio#servizio`

dove

- **nome_servizio** è il nome del servizio;
- **#servizio** è l'id assegnato al servizio (vedere la Sezione relativa alla configurazione del Sistema).

Un esempio è riportato nella figura che segue:

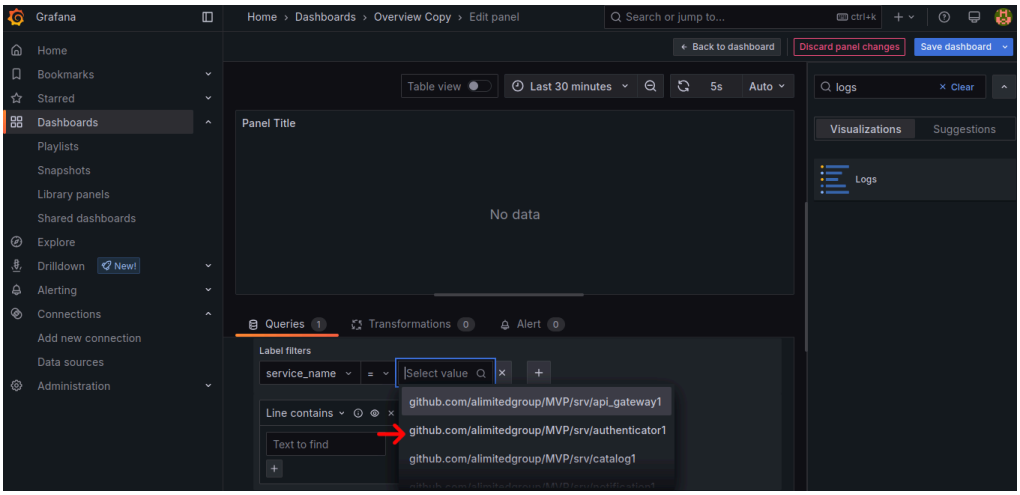

Figura 23: Selezione del servizio di cui visualizzare i log

Attenzione: il servizio potrebbe comparire nella solo nel momento in cui viene fornito il primo output.

Premere quindi **Run query** e **Save dashboard** per sincronizzare i dati e salvare le modifiche.

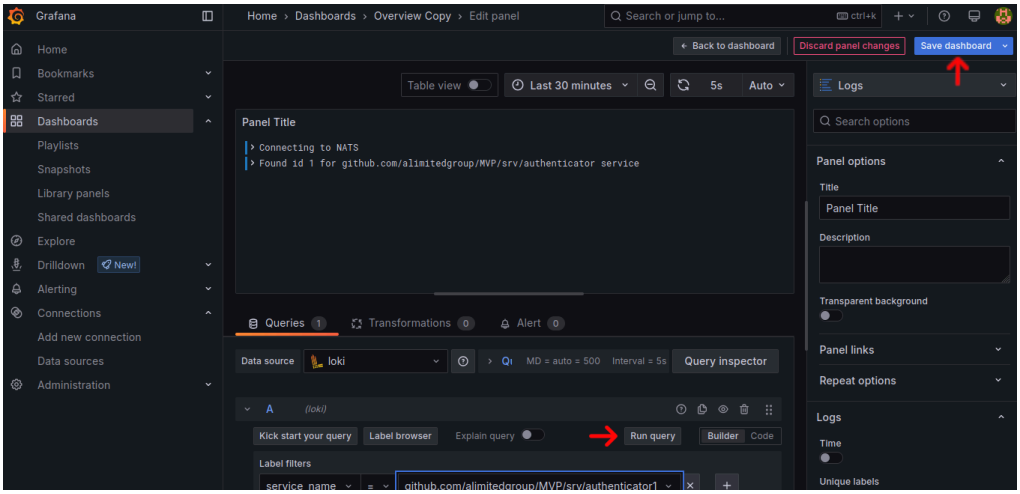

Figura 24: Salvataggio dashboard dopo aggiunta dei log

Il pannello è stato aggiunto: premere **Back to dashboard** per visualizzarlo.

6.5.2 - Aggiungere una misurazione

Dopo l'apertura del menu di aggiunta visualizzazione, il campo **Data source** presente nella parte inferiore dovrebbe essere già popolato con **Prometheus**: se così non fosse selezionarlo similmente a quanto fatto con Loki.

Premere quindi sul menu **Select metric**, quindi su **Metrics explorer**.

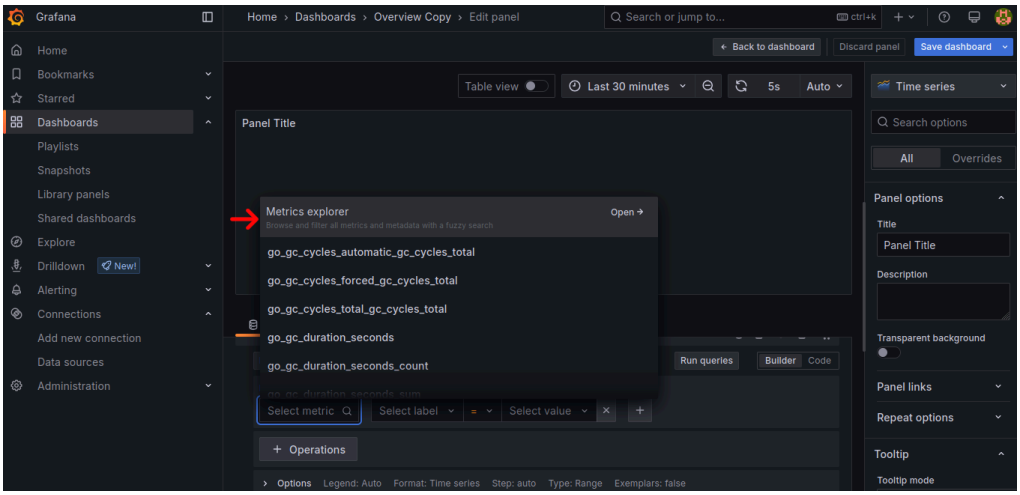

Figura 25: Apertura del menu di selezione metriche

Cercare, nella finestra successiva, la metrica di interesse, quindi premere sul bottone mostrato in figura:

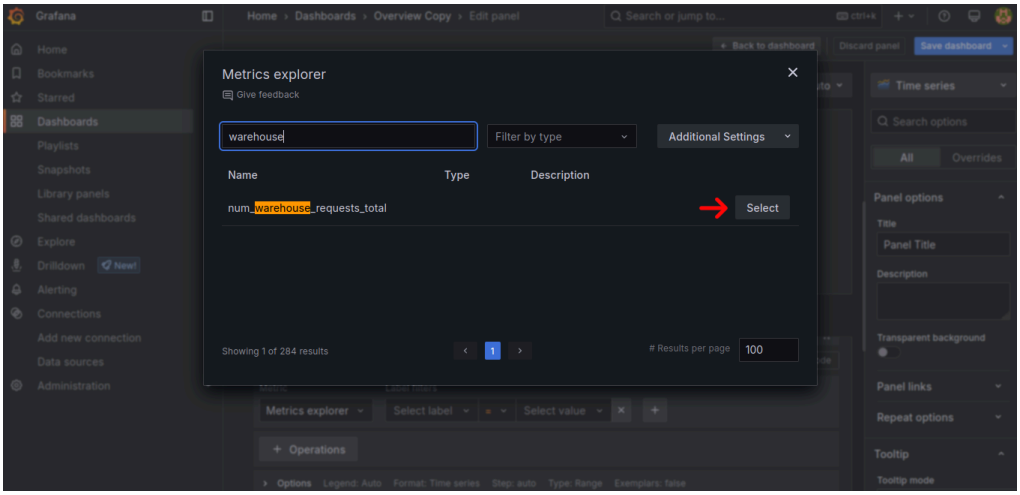

Figura 26: Selezione della metrica

Attenzione: la metrica potrebbe non apparire se non è mai stata fatta una misurazione a riguardo.

Premere quindi su **Select label** e, nel menu che si apre, premere **Job**.

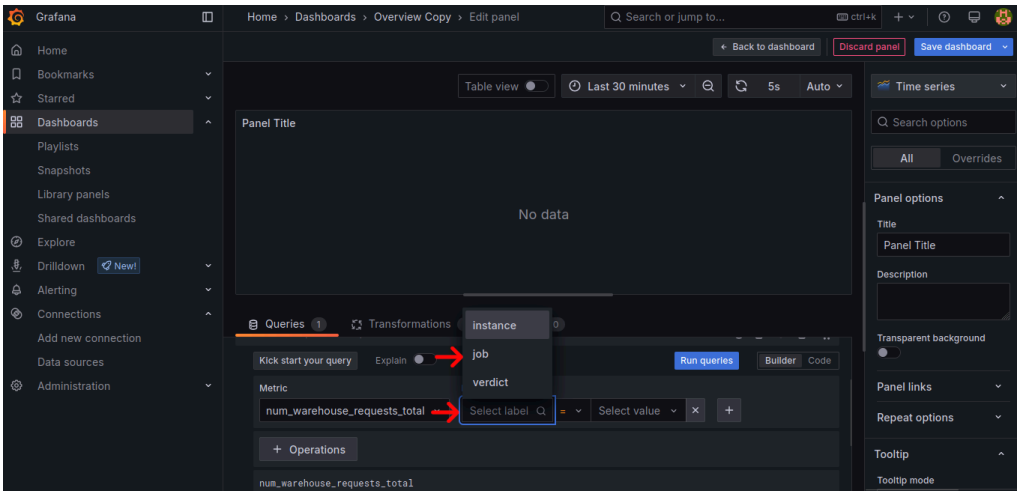

Figura 27: Selezione modalità individuazione servizio per metriche

Premere ora su **Select value** e premere sul servizio di interesse. Per una rapida individuazione, i nomi sono nella forma che segue:

`github.com/alimitedgroup/MVP/srv/nome_servizio#servizio`

dove

- **nome_servizio** è il nome del servizio;
- **#servizio** è l'id assegnato al servizio (vedere la Sezione relativa alla configurazione del Sistema).

Un esempio è riportato nella figura che segue:

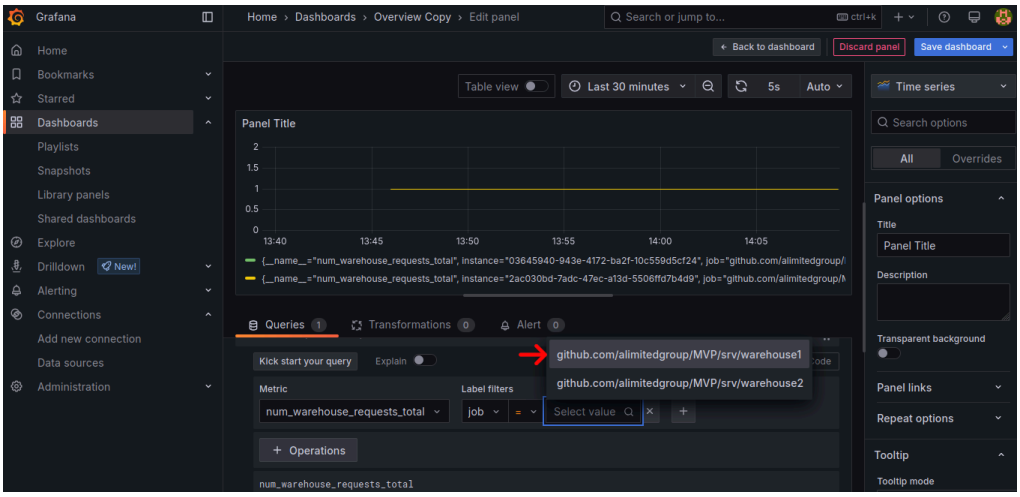

Figura 28: Selezione del servizio di cui visualizzare una metrica

Attenzione: il servizio potrebbe comparire nella solo nel momento in cui viene fornita una prima misurazione.

Premere quindi **Run query** e **Save dashboard** per sincronizzare i dati e salvare le modifiche.

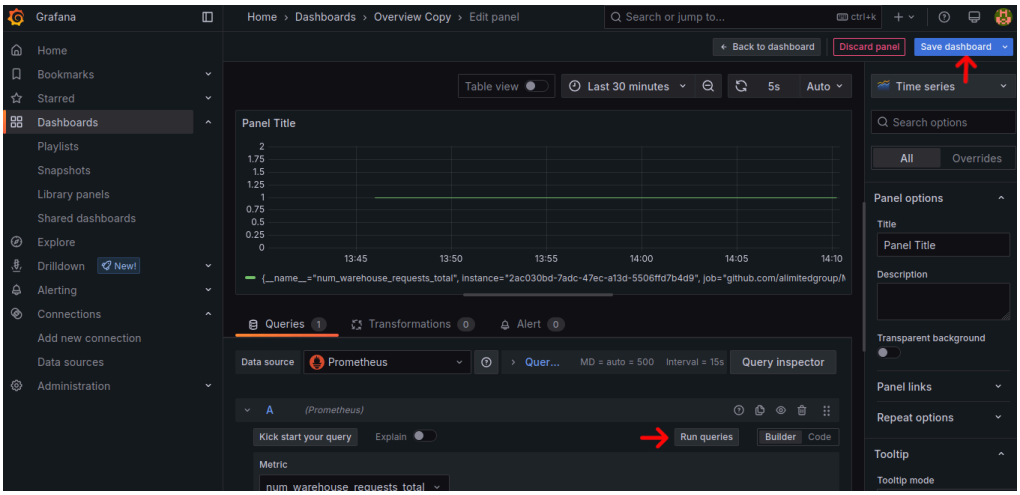

Figura 29: Salvataggio dashboard dopo aggiunta dei log

Il pannello è stato aggiunto: premere **Back to dashboard** per visualizzarlo.

7 - Riferimento API

7.1 - GET `/api/v1/ping`

Questa route consente di verificare se il servizio `api_gateway` è correttamente in esecuzione.

7.1.1 - Richiesta autenticazione

No.

7.1.2 - Parametri

Nessuno.

7.1.3 - Corpo della richiesta

Nessuno.

7.1.4 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`message`	`string`	Valore fisso, sempre uguale a «pong».	`pong`

Tabella 3: Risposta di GET `/api/v1/ping`

7.2 - POST `/api/v1/login`

Questa route consente di autenticarsi presso il servizio `api_gateway`, in modo da poter accedere alle route che richiedono autenticazione.

7.2.1 - Richiesta autenticazione

No.

7.2.2 - Corpo della richiesta

Codifica: `application/x-www-form-urlencoded`

Nome	Tipo	Descrizione	Esempio
`username`	`string`	L'username con cui ci si vuole autenticare	`admin`

Tabella 4: Corpo della richiesta di POST `/api/v1/login`

7.2.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio

`token`	`string`	Un token JWT che autentica l'utente.	`eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.`
			`eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZ`
			`SI6IkpvaG4gRG9lIiwiYWRtaW4iOn`
			`RydWUsImlhdCI6MTUxNjIzOTAyMn0.`
			
			`KMUFsIDTnFmyG3nMiGM6H9FNFUROf3wh7SmqJp-QV30`

Tabella 5: Risposta di POST `/api/v1/login`

7.3 - GET `/api/v1/is_logged`

Questa route consente di verificare lo stato di autenticazione. Inoltre, consente di ottenere il proprio ruolo attuale, qualora non sia desiderabile ottenerlo estraendolo dal token JWT.

7.3.1 - Richiesta autenticazione

Sì. Per ulteriori dettagli, fare riferimento a "autenticazione e autorizzazione", all'interno dei concetti.

7.3.2 - Corpo della richiesta

Nessuno.

7.3.3 - Risposta: 200

Il token fornito è valido

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`role`	`string`	Il ruolo dell'utente associato al token fornito. Il ruolo può essere uno tra le tre stringhe `client`, `local_admin` e `global_admin`.	`global_admin`

Tabella 6: Risposta di GET `/api/v1/is_logged`

7.4 - GET `/api/v1/goods`

Questa route consente di recuperare la lista di `good`[G] (merci)

7.4.1 - Richiesta autenticazione

Sì.

7.4.2 - Corpo della richiesta

Nessuno.

7.4.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`goods`	`GoodAnd Amount[]`	Una lista di merci, ognuna con la quantità disponibile globalmente e per singoli magazzini.	`"goods": [{"name": "hat", "description": "a hat", "id": "e9281371-b9fe-41b4-968a-e510a98cf849", "amount": 20, "amounts": {"warehouse-1": 10, "warehouse-2": 10}}]`

Tabella 7: Risposta di GET `/api/v1/goods`

7.5 - GET `/api/v1/warehouses`

Questa route consente di recuperare la lista di `warehouse`[G] (magazzini)

7.5.1 - Richiesta autenticazione
Sì.

7.5.2 - Corpo della richiesta
Nessuno.

7.5.3 - Risposta: 200
L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`ids`	`string[]`	Una lista di ID dei magazzini.	`"ids": ["f7465619-4e3c-4a3e-8f0d-0379666bb47a", "2ecd185a-f7a6-4834-b29c-92657485d284"]`

Tabella 8: Risposta di GET `/api/v1/warehouses`

7.6 - GET `/api/v1/orders`
Questa route consente di recuperare la lista di `orders` (ordini)

7.6.1 - Richiesta autenticazione
Si.

7.6.2 - Corpo della richiesta
Nessuno.

7.6.3 - Risposta: 200
L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`orders`	`Order[]`	Una lista di ordini, con le relative informazioni.	`[{"order_id": "0353a051-f1a4-40fd-9189-aedbd28cb0a7", "status": "Filled", "name": "Order 1", "full_name": "Mario Rossi", "address": "Via Roma 1, Milano", "goods": {"002bcb63-32b3-4087-ab5b-2cbb881f8824": 2, "67209dcc-2103-49af-aeb1-4c81a8fb3ddd": 3}, "reservations": ["74092245-36b4-473a-bacc-b1bbe245a113", "f6410fea-7518-45ff-b39d-622a488f6304"]}]`

Tabella 9: Risposta di GET `/api/v1/orders`

7.7 - GET `/api/v1/transfers`

Questa route consente di recuperare la lista di `transfers` (trasferimenti)

7.7.1 - Richiesta autenticazione

Si.

7.7.2 - Corpo della richiesta

Nessuno.

7.7.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`transfers`	`Transfer[]`	Una lista di trasferimenti, con le relative informazioni.	`[{"transfer_id": "9ae7b261-3ade-49be-962c-b9c036aee947", "status": "Filled", "sender_id": "warehouse_001", "receiver_id": "warehouse_002", "goods": {"e8e0e7e9-3df6-44d9-86d9-341eedaf3e74": 2, "6c0344b8-262b-4ff6-8922-62d898d76f07": 3}}]`

Tabella 10: Risposta di GET `/api/v1/transfers`

7.8 - POST `/api/v1/goods`

Questa route consente di creare una merce.

7.8.1 - Richiesta autenticazione

Si.

7.8.2 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`name`	`string`	Nome della merce	`"Abito blu"`
`description`	`string`	Descrizione della merce	`"Abito blu di alta qualità"`

Tabella 11: Corpo della richiesta di POST `/api/v1/goods`

7.8.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`good_id`	`string`	Identificativo della merce creata	`876f754f-e7da-4f07-b6de-258bbfb384bc`

Tabella 12: Risposta di POST `/api/v1/goods`

7.9 - POST `/api/v1/orders`

Questa route consente di creare un ordine[G] di merci.

7.9.1 - Richiesta autenticazione

Si.

7.9.2 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`name`	`string`	Nome dell'ordine	`"ordine1"`
`full_name`	`string`	Nome completo del cliente	`"Mario Rossi"`
`address`	`string`	Indirizzo del cliente	`"Via Roma 10, Milano"`
`goods`	`map[string]int64`	Mappa delle merci con quantità associate	`{"b55a6a92-6c3e-4934-9392-d6a40d940c81": 5, "7f2a6642-a6bf-4c79-acad-43f1038761b5": 3}`

Tabella 13: Corpo della richiesta di POST `/api/v1/orders`

7.9.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`order_id`	`string`	Identificativo dell'ordine creato	`c7bea6a8-8f44-4146-b6ab-c84ba0d14774`

Tabella 14: Risposta di POST `/api/v1/orders`

7.10 - POST `/api/v1/transfers`

Questa route consente di creare un trasferimento[G] di merci da un magazzino all'altro.

7.10.1 - Richiesta autenticazione

Si.

7.10.2 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`sender_id`	`string`	Identificativo del magazzino mittente	`"warehouse-1"`
`receiver_id`	`string`	Identificativo del magazzino destinatario	`"warehouse-2"`
`goods`	`map[string]int64`	Mappa delle merci con quantità associate	`{"e6064c0c-4a0c-4f37-b810-7025510a6ecf": 10, "14794ce1-c418-42be-a564-d66e23833db8": 20}`

Tabella 15: Corpo della richiesta di POST `/api/v1/transfer`

7.10.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`transfer_id`	`string`	Identificativo del trasferimento creato	`a877f9f4-5f98-4ede-8284-8ea1f74195d5`

Tabella 16: Risposta di POST `/api/v1/transfers`

7.11 - PUT `/api/v1/goods/:good_id`

Questa route consente di aggiornare una merce.

7.11.1 - Richiesta autenticazione

Si.

7.11.2 - Variabili PATH

Nome	Tipo	Descrizione	Esempio
`good_id`	`string`	L'identificativo della merce	`c8225f15-ba60-47f6-8f84-ee72ca3d3d73`

Tabella 17: Variabili PATH di PUT `/api/v1/goods/:good_id`

7.11.3 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`name`	`string`	Nome della merce	`"Abito blu"`
`description`	`string`	Descrizione della merce	`"Abito blu di alta qualità"`

Tabella 18: Corpo della richiesta di PUT `/api/v1/goods/:good_id`

7.11.4 - Risposta: 200

L'operazione è stata eseguita con successo.

Corpo della risposta vuoto.

7.12 - POST `/api/v1/goods/:good_id/warehouse/:warehouse_id/stock`

Questa route consente di aggiungere dello stock[G] ad una merce del magazzino.

7.12.1 - Richiesta autenticazione

Si.

7.12.2 - Variabili PATH

Nome	Tipo	Descrizione	Esempio
`good_id`	`string`	L'identificativo della merce	`7be87589-6835-4437-ba6a-96d6b9a383c1`
`warehouse_id`	`string`	L'identificativo del magazzino	`warehouse-1`

Tabella 19: Variabili PATH di POST `/api/v1/:good_id/warehouse/:warehouse_id/stock`

7.12.3 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`quantity`	`int`	La quantità di stock da aggiungere	`10`

Tabella 20: Corpo della richiesta di POST `/api/v1/:good_id/warehouse/:warehouse_id/stock`

7.12.4 - Risposta: 200

L'operazione è stata eseguita con successo.

Corpo della risposta vuoto.

7.13 - DELETE `/api/v1/goods/:good_id/warehouse/:warehouse_id/stock`

Questa route consente di rimuovere dello stock[G] ad una merce del magazzino.

7.13.1 - Richiesta autenticazione

Si.

7.13.2 - Variabili PATH

Nome	Tipo	Descrizione	Esempio
`good_id`	`string`	L'identificativo della merce	`7be87589-6835-4437-ba6a-96d6b9a383c1`
`warehouse_id`	`string`	L'identificativo del magazzino	`warehouse-1`

Tabella 21: Variabili PATH di DELETE `/api/v1/goods/:good_id/warehouse/:warehouse_id/stock`

7.13.3 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`quantity`	`int`	La quantità di stock da rimuovere	`10`

Tabella 22: Corpo della richiesta di DELETE `/api/v1/goods/:good_id/warehouse/:warehouse_id/stock`

7.13.4 - Risposta: 200

L'operazione è stata eseguita con successo.

Corpo della risposta vuoto.

7.14 - POST `/api/v1/notifications/queries`

Questa route consente di creare una query per le notifiche.

7.14.1 - Richiesta autenticazione

Si.

7.14.2 - Corpo della richiesta

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`good_id`	`string`	Nome della merce	`"Identificativo della merce"`
`operator`	`string`	Operatore logico da utilizzare nella query	`"<"`
`threshold`	`int`	Valore numerico della query	`"10"`

Tabella 23: Corpo della richiesta di POST `/api/v1/notifications/queries`

7.14.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`good_id`	`string`	Identificativo della query creata	`876f754f-e7da-4f07-b6de-258bbfb384bc`

Tabella 24: Risposta di POST `/api/v1/notifications/queries`

7.15 - GET `/api/v1/notifications/queries`

Questa route consente di recuperare la lista di query delle notifiche.

7.15.1 - Richiesta autenticazione

Sì.

7.15.2 - Corpo della richiesta

Nessuno.

7.15.3 - Risposta: 200

L'operazione è stata eseguita con successo.

Codifica: `application/json`

Nome	Tipo	Descrizione	Esempio
`queries`	`Query[]`	Una lista di query delle notifiche, ognuna con il tipo di soglia monitorata.	`"queries": [{"query_id":"39dfe1b4-35b2-467b-b86f-a915ac362cba","good_id": "hat-1","operator": "<","threshold": 5}]`

Tabella 25: Risposta di GET `/api/v1/notifications/queries`