Manuale Utente

2024-02-14 — v0.0.2

overture.unipd@gmail.com

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

Registro delle modifiche

Versione	Data	Autori	Verificatori	Dettaglio
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 4
1.1) Scopo del prodotto 4
1.2) Scopo del documento 4
1.3) Glossario 4
1.4) Riferimenti 4
1.4.1) Riferimenti normativi 4
1.4.2) Riferimenti informativi 4
2) Avvio del server di posta elettronica 5
2.1) Requisiti tecnici per avviare il server 5
2.1.1) Requisiti opzionali 5
2.2) Download del repository relativo al codice sorgente del server 5
2.3) Spiegazione dei principali file presenti nei sorgenti 5
2.3.1) File: Compose.yml 5
2.3.2) File: Caddyfile 5
2.3.3) File: build.gradle.kts 5
2.3.4) File: startup.jsh 6
2.3.5) File: .env 6
2.3.6) File: .justfile 6
2.4) Duck DNS 6
2.4.1) Guida a Duck DNS 6
2.4.2) Aggiornamento server per Duck DNS 7
2.5) Istruzioni per l'avvio 7
2.6) Istruzioni per lo spegnimento 8

1) Introduzione

1.1) Scopo del prodotto

Lo scopo principale del prodotto é 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 le 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 avviare un client di posta elettronica, una breve guida per mostrare il funzionamento di un client e di come riesce a fruire di tutti i requisiti che il server implementa e 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 ambiguitá o incomprensioni riguardanti la terminologia usata nel documento, é 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 v1.0.0`:
  https://overture-unipd.github.io/docs/rtb/interni/norme_di_progetto_v1.0.0.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 v1.0.0`:
  https://overture-unipd.github.io/docs/rtb/interni/glossario_v1.0.0.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

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 commandi neccessari all'avvio presenti nel file .justile come spiegato nella sezione **File: .justfile**.

2.2) Download del repository relativo al codice sorgente del server

Per scaricare i sorgenti relativi al codice sorgente del server basta clonare la repository dedicata al link: 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'istanziazione 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 farà 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:

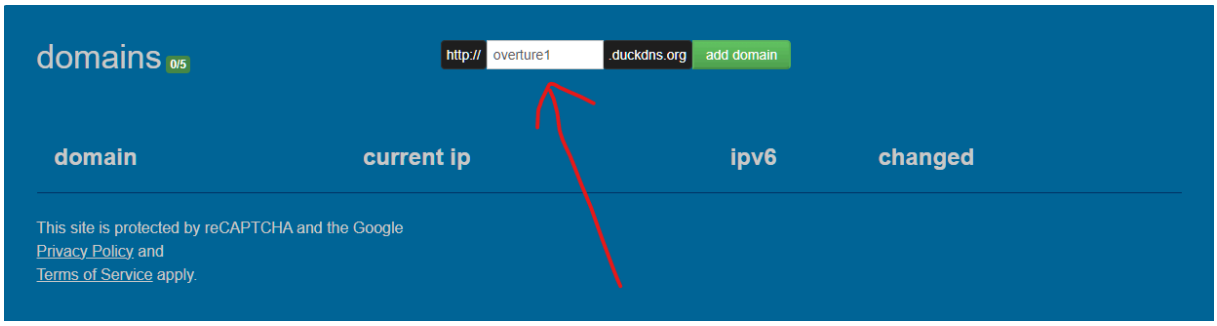

Figura 1: Maschera per l'inserimento del nome del sottodominio da associare all'Indirizzo IP privato del server in uso.

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 -versione`).

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 dockerBuilImage`: 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.