Mail Parser

Introduzione

La mailing list al giorno d'oggi è un metodo di comunicazione tipicamente gestito da aziende, associazioni o organizzazioni che vogliono comunicare eventi, notizie o altro ad una vasta lista di destinatari interessati. Il meccanismo è semplice in quanto un messaggio email inviato ad un sistema server viene inoltrato automaticamente in multicast alla lista desiderata.

Il progetto si occupa della parte back-end , effettuando il parse delle email e permettendo ad un utente di inserire, cancellare e ricercare indirizzi e-mail in un database mysql.

L'obiettivo principale è quello di effettuare un'analisi sintattica di file di testo per estrarre indirizzi e-mail con formato valido.

Un indirizzo e-mail è costituito da due parti principali:

- **localpart:** numero di caratteri non superiore ai 64;
- **domainpart**: numero di caratteri non superiore ai 25;

caratteri ammissibili: a-z , A-Z, 0-9, _

(esempio: local-part@domain-part)

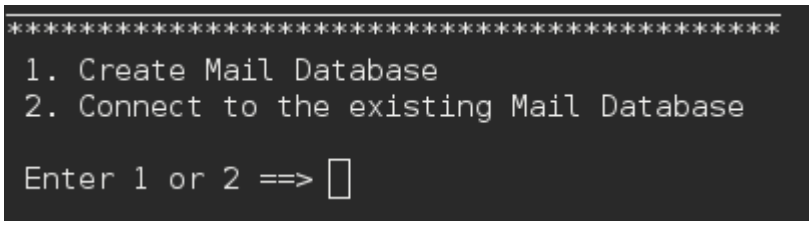
La validazione di ogni singola email avviene, quindi, mediante una funzione che controlla se il formato è conforme allo standard.

Il controllo dei caratteri ammissibili avviene tramite l'espressione regolare

“^[_a-zA-Z0-9-]+(\\.[_a-zA-Z0-9-]+)*@[a-zA-Z0-9-]+(\\.[a-zA-Z0-9-]+)*(\\.[a-zA-Z]{2,13})$”

STRUTTURA DEL PROGETTO

1. Inizializzazione del database



Come prima opzione si ha la possibilità di scegliere se

1. creare un nuovo database
2. connettersi ad un database già esistente

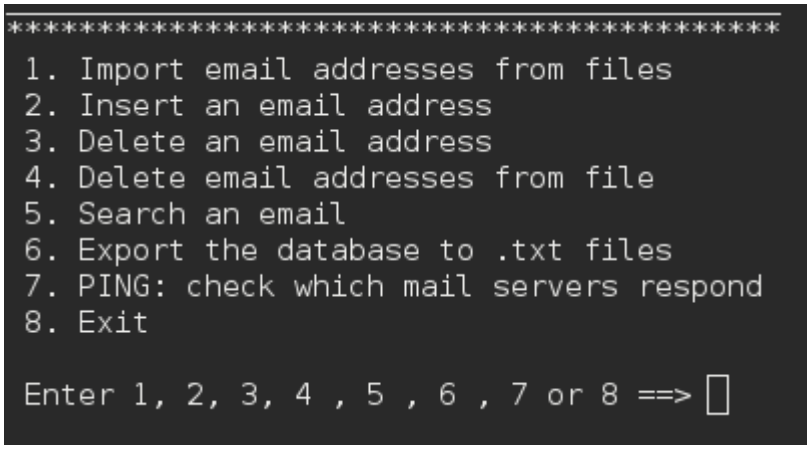
La ***creazione di un nuovo database*** viene effettuata richiamando la funzione *create_db()* che al suo interno utilizza la query *"CREATE DATABASE IF NOT EXIST".*
Vengono poi create le tabelle Address e Domain_part all'interno del database richiamando la funzione *createTable()* che al suo interno utilizza le seguenti query:

- *"CREATE TABLE Address (address_id varchar(150) NOT NULL,local_part varchar(70) NOT NULL,domain_part varchar(30) NOT NULL,PRIMARY KEY (address_id)) ENGINE=InnoDB DEFAULT CHARSET=ascii;";*
- *"CREATE TABLE Domain_part(domain_part varchar (25) NOT NULL, invalid_domain int(1),PRIMARY KEY (domain_part)) ENGINE=InnoDB DEFAULT CHARSET=ascii;"*

*La **selezione di un database*** viene effettuata richiamando la funzione *select_db()* che seleziona ed accede ad un database già esistente.

Le funzioni citate sono definite nel file *init_db.h*.

2. Servizi forniti:



Una volta collegatosi al database, vengono proposti diversi servizi :

1. ***inserimento di email tramite cartella o file***

   Viene letto il path passato da tastiera. Se quest'ultimo si riferisce ad un file *.txt / .doc / .rtf /.csv* viene richiamata la funzione *parse_file().* Se, invece, si riferisce ad una cartella viene richiamata la funzione *read_directory().*
   La funzione parse_file() legge il file passato in input e suddivide il testo in token, ovvero in stringhe delimitate dai seguenti caratteri: ",<\">:[](){} ".

   Ogni stringa viene poi passata come input alla funzione validate() che verifica se il suo formato è conforme al formato standard di una email.
   Se l'esito è positivo aggiunge l'email al database mediante la funzione *add_address().*

   La funzione **validate(char *address)** prevede quattro fasi:

   1. richiama la funzione match che controlla se la stringa corrisponde con l'espressione regolare sopra citata
   2. se l'esito è positivo (match ritorna 1), viene fatto un controllo sulla lunghezza della localpart e domainpart dell'email in esame
   3. verifica se il dominio, estratto dalla domain-part, appartiene all'insieme dei domini di primo livello esistenti confrontandolo, quindi, con quelli dichiarati nel file *"first_level_domain.txt"*
   4. se la verifica del dominio va a buon fine, si controlla se la lunghezza della localpart è superiore ai 25 caratteri. In questo caso l'email viene inserita anche in un file di warning, così da segnalare all'amministratore la presenza di email con localpart estramemente lunghe

   La funzione validate è definita nel file *validate.h* .

   La funzione **add_address(char *addr)** richiama al suo interno str_to_lower (definita in service.h) per convertire i caratteri dell'indirizzo da maiuscolo a minuscolo. Aggiunge poi l'indirizzo email alla tabella Address e la domain_part alla tabella Domain_part tramite le seguenti query:

   5. *"INSERT INTO Address (address_id,local_part,domain_part) VALUES ('%s','%s','%s');",addr,local_part,domain_part) "*
   6. *"INSERT INTO Domain_part (domain_part) VALUES ('%s');"*

   La funzione add_address è definita nel file *db.h.*

2. ***inserimento manuale di un'email***

   L'inserimento manuale di un' email consiste nel leggere la stringa passata da tastiera e richiamare *validate* per la sua validazione. Se l'esito è positivo la aggiunge al database tramite *add_address.*

3. **cancellazione di una email tramite inserimento manuale**

   La cancellazione di un'email inserita da tastiera consiste nel leggere la stringa di input e richiamare al suo interno la funzione *delete_address(char *address).*

   La funzione ***delete_address(char *address)*** effettua la ricerca dell'indirizzo email da cancellare tramite la funzione search_mail(char *address). Se quest'ultima ritorna 1, l'email viene cancellata dal database tramite la query *"DELETE FROM Address WHERE address_id='%s';",address)".*

   La funzione delete_address è definita in *db.h* .

4. ***cancellazione di email da file di input*** *(.txt / .doc / .rtf /.csv)*

   La cancellazione di email viene effettuata dalla funzione ***delete_addresses_from_file(char *file_path)*** che legge il file passato in input (.txt/.doc/.cvs/.rtf/) e suddivide il testo in token, ovvero in stringhe delimitate dai seguenti caratteri: ",<\">:[](){}. Per ogni stringa viene richiamata la funzione ***delete_address(char *address)*** che provvede a cancellarla dal database.

   La funzione delete_address_from_file è defnita in db.h.

5. ***ricerca di una email***

   La ricerca di un' email avviene passando in input la stringa inserita manualmente alla funzione ***search_mail(char *address)*** *che* effettua la ricerca dell'email all'interno del database tramite la query *"SELECT count(address_id) FROM Address WHERE address_id='%s';", address)"*

   La funzione search_mail è definita in *db.h* .

6. ***esportazione del database in file.txt***

   Il salvataggio di tutte le email presenti nel database avviene tramite la funzione ***export_db()*** che crea una cartella denominata db_name_valid_addresses che conterrà le email salvate.
   Per ogni carattere 0-9 e a-z , richiama la funzione ***save (char ch).*** *S*e quest'ultima ritorna 1 vuol dire che ha effettuato il salvataggio correttamente.

   La funzione ***save (char ch),*** dopo aver controllato se nel database corrente esiste almeno una entry che inizia per il carattere "ch" passato come parametro, crea una cartella "ch" in cui vengono inseriti i file contententi tutte le email che iniziano per "ch" selezionate tramite la query *"SELECT address_id FROM Address WHERE local_part like '%c%';",ch);"*

   Ogni qualvolta si arriva ad una entry la cui posizione nel database è multipla di 500, viene

creato un nuovo file di testo. Quindi, ogni file contiene al massimo 500 indirizzi email.

In seguito all'esportazione del database, viene chiamata la funzione **save_all()** che crea un ulteriore file in db_name_valid_addresses contenente tutte le email presenti nel database.

Le funzioni export_db(), save (char ch) e save_all() sono definite in *db.h.*

7. Ping

Questa opzione verifica quali domain_part tra quelle presenti nella tabella Domain_part sono raggiungibili e quali no. La funzione ***ping()*** seleziona tutte le domain_part tramite la query *"SELECT domain_part FROM Domain_part;".*
Per ognuna di esse viene effettuata la chiamata di sistema system *"ping -c 1 – w 5 domain_part"* dove -c indica il numero di pacchetti trasmessi e -w: indica il tempo di attesa in secondi.
Se quest'ultima restuisce un valore diverso da 0 e, quindi, il dominio non è raggiungibile o non esiste, viene settato il corrispondente campo invalid_domain a 1 sia nella tabella Address che nella tabella Domain_part, tramite le seguenti query:

- "UPDATE Address SET invalid_domain='1' WHERE domain_part='%s';",row[0]);
- "UPDATE Domain_part SET invalid_domain='1' WHEREdomain_part='%s';",row[0]);

Viene poi effettuato un conteggio di tutti i server dei domini che non rispondono, richiamando la funzione **count_ping_no_respons()** che utilizza la query *"SELECT count(domain_part) FROM Domain_part WHERE invalid_domain'1';".*

Le funzione ping() e count_ping_no_respons() sono definita in *ping.h* .

8. disconnessione dal database

La disconnessione del database avviene tramite la query *mysql_close(&mysql).*