Università degli Studi di Camerino

Scuola di Scienze e Tecnologie
Corso di Laurea in Informatica (Classe L-31)

REALIZZAZIONE DI UN SERVER VOIP E RELATIVA INTERFACCIA WEB PER LA GESTIONE DEI RITIRI DI CAMPIONI IN UN CENTRO ANALISI

Laureando:
Caferri Stefano

Relatore:
Prof. Marcantoni Fausto

Correlatore:
Dott. Ing. Lupinetti Alessandro

Anno Accademico 2016/2017

Indice

Abstract

Nei centri analisi, il ritiro dei campioni da analizzare viene gestito per via telefonica:
il paziente si reca in farmacia e consegna il campione, il farmacista telefona al centro analisi per prenotare il ritiro, dal centro analisi una segretaria risponde, si appunta la prenotazione e successivamente si accorda con il corriere per procedere al ritiro presso la farmacia. Tutto ciò può causare contrattempi e problemi organizzativi a tutti i soggetti coinvolti: farmacia, centro analisi e corriere, senza contare il disagio per l'utente finale, ovvero il paziente.

Per cercare di automatizzare e semplificare questo processo, si è voluto realizzare, partendo da zero, un server VoIP che implementi un centralino con sistema IVR (Interactive Voice Response) per permettere alla farmacia di telefonare, autenticarsi ed effettuare la prenotazione per il ritiro dei campioni. La seconda parte dell'applicazione prevede la realizzazione di un'interfaccia web offerta dallo stesso server, che consenta al corriere di autenticarsi e visualizzare i ritiri da effettuare presso la sua zona di competenza. L'interfaccia si compone di altri due livelli di utenza: il manager, che coordina i corrieri di tutte le zone e l'amministratore, che gestisce ogni utenza presente sul server.

Il sistema operativo scelto per il server è Debian GNU/Linux 9 (Stretch), il software PBX utilizzato per realizzare la parte VoIP è Asterisk 13, a cui viene collegata una numerazione VoIP fornita dal provider MessageNet. La parte di interfaccia web è realizzata mediante uno stack LAMP (Linux, Apache, MySQL, Php).

1. Introduzione all'Ambiente di Sviluppo

In questo capitolo si descriveranno la tecnologia VoIP e la piattaforma utilizzata per gestire la telefonia, introducendo concetti che risulteranno essenziali nello svolgimento del progetto. Verrà poi analizzato come applicare tali tecnologie al fine di risolvere il problema preso in esame. Questo capitolo ha scopo introduttivo e non entra nello specifico della configurazione utilizzata nella realizzazione del centralino, che verrà definita in dettaglio successivamente.

1.1 La tecnologia VoIP

Il VoIP, acronimo di *voice over IP* è una tecnologia che permette di effettuare conversazioni telefoniche utilizzando il protocollo IP, sia mediante la rete internet che attraverso reti locali. Il segnale analogico della chiamata telefonica viene convertito in digitale ed il flusso dati della chiamata viene trasformato in pacchetti conformi al protocollo IP. In questo modo, si passa dalla trasmissione a commutazione di circuito della linea telefonica tradizionale (PSTN), in cui viene dedicato un circuito tra il chiamante e il chiamato, che resta attivo per l'intera durata della comunicazione, alla trasmissione a commutazione di pacchetto, in cui le risorse vengono allocate dinamicamente, instradando i pacchetti solo quando necessario, ovvero quando uno dei partecipanti sta effettivamente parlando. Mediante questa tecnologia è possibile comunicare con la rete PSTN tradizionale, oltre che ovviamente con altri apparecchi VoIP presenti sulla rete.

I vantaggi del VoIP sono molteplici, e tra i più importanti possiamo sicuramente citare l'abbattimento dei costi, soprattutto in ambito aziendale. Utilizzando una rete comune per l'intera infrastruttura di comunicazione, i costi di installazione, manutenzione e aggiornamento vengono notevolmente diminuiti. È inoltre possibile avere un unico numero telefonico al quale sono connessi più dispositivi VoIP, facilitando la comunicazione con la clientela. L'integrazione in una tecnologia esistente e consolidata come quella della rete IP fornisce ulteriori vantaggi in ambito aziendale, come ad esempio la possibilità di creare reti VoIP per effettuare conferenze e formazione del personale, fornendo la possibilità di collegare tra loro più sedi della stessa azienda, possibilità di utilizzare funzionalità già diffuse in rete quali videoconferenza, messaggistica, utilizzo di applicativi per monitorare e organizzare gli apparecchi telefonici, archiviare le chiamate, i messaggi registrati in segreteria e così via.

La problematica principale del VoIP (e della commutazione a pacchetto in generale) è qualità della comunicazione. Mentre una rete PSTN garantisce qualità costante durante tutta la chiamata, utilizzando il protocollo IP l'informazione viene compressa e trasmessa in pacchetti instradati sulla rete e successivamente decompressa e riprodotta. La qualità è dipendente dall'instradamento dei pacchetti e dalla banda a disposizione, il che in alcuni casi può

comportare problemi di eco e latenza. Un altro aspetto da tenere in considerazione sono le implicazioni relative alla sicurezza: condividendo lo stesso protocollo di trasporto, si condividono gli stessi rischi degli altri servizi della rete, come ad esempio: furto delle credenziali di accesso, intercettazione non autorizzata delle conversazioni, *denial of service*[1] È opportuno quindi assicurare un livello di sicurezza adeguato al contesto in cui si opera.
Bisogna infine notare che le implicazioni sulla sicurezza, seppur differenti, sono da tenere in considerazione anche con una linea telefonica tradizionale.

1.2 Asterisk

Asterisk è una piattaforma open source, compatibile con qualsiasi sistema operativo basato sul kernel Linux, per sviluppare applicazioni di comunicazione. Se ad esempio un webserver utilizza Apache o nginx per fornire pagine web, un IP PBX utilizza Asterisk per gestire le comunicazioni VoIP. IP PBX sta per *Internet Protocol Private Branch eXchange* e in sostanza rappresenta il sistema centrale che riceve, organizza e invia comunicazioni (telefonate, video, messaggi) in un'azienda. Come vedremo più avanti, un sistema PBX è collegato ad un provider di telefonia esterno (PSTN) che fornisce il numero di telefono e instrada il traffico verso il sistema PBX in modo analogo all'*internet service provider* (ISP) che svolge le stesse mansioni per il collegamento internet.

Il sistema PBX è incaricato di smistare il traffico VoIP e di implementare le varie funzionalità richieste dall'azienda. Tra queste ci sono la creazione di varie stazioni (o estensioni) e come queste operano tra loro e con il traffico esterno. Le stazioni non sono altro che gli apparecchi VoIP utilizzati dagli operatori dell'azienda, che possono essere persone fisiche o applicazioni che gestiscono le comunicazioni e forniscono servizi. Tra le applicazioni utilizzate vi è quella dell'*interactive voice response* (IVR), che consiste in un menù vocale che automatizza operazioni di registrazione e fornitura di servizi e che sarà una delle soluzioni utilizzate per lo sviluppo di questo progetto.
Asterisk permette agli sviluppatori di realizzare sistemi di questo tipo, attraverso la configurazione di diversi parametri, presenti nei vari file di configurazione. Agli sviluppatori è fornita una documentazione ufficiale che permette di capire le numerose funzionalità della piattaforma e come la stessa è organizzata e si interfaccia con il sistema operativo.

Sono disponibili molteplici soluzioni per utilizzare Asterisk al fine di realizzare un sistema PBX; come vedremo nel capitolo seguente, esistono distribuzioni come ad esempio FreePBX, PBX in a Flash e AsteriskNow (sviluppata da Asterisk) basate su distribuzioni Linux orientate all'uso server che vengono sviluppate da aziende del settore della comunicazione e forniscono, gratuitamente o a pagamento, un sistema completo e pronto all'uso che permette di configurare il sistema di telefonia attraverso interfaccia grafica. Un altro tipo di approccio è

1 Attacco informatico che mira a rendere indisponibile uno o più computer presenti in rete.

quello di realizzare un sistema PBX da zero, installando e configurando ogni parte del sistema operativo e di Asterisk esattamente secondo le proprie esigenze.

1.2.1 Struttura di Asterisk

Di seguito si andranno a descrivere le varie parti che compongono Asterisk e i protocolli principali utilizzati per realizzare un sistema PBX di base:

```
[directories]
astetcdir => /etc/asterisk
astmoddir => /usr/lib/asterisk/modules
astvarlibdir => /var/lib/asterisk
astdbdir => /var/lib/asterisk
astkeydir => /usr/share/asterisk
astdatadir => /usr/share/asterisk
astagidir => /srv/agi
astspooldir => /var/spool/asterisk
astrundir => /var/run/asterisk
astlogdir => /var/log/asterisk
astsbindir => /usr/sbin
```

Snippet 1: asterisk.conf

Il file *asterisk.conf* definisce la struttura dell'intera piattaforma. La sezione *directories* definisce i percorsi in cui Asterisk si aspetta di trovare gli elementi principali del sistema, in particolare:

- file di configurazione (astetcdir);
- moduli (astmoddir);
- librerie, elementi e file addizionali (astvarlibdir);
- database interno (astdbdir);
- chiavi di cifratura (astkeydir);
- suoni nei vari linguaggi definiti (astdatadir);
- classi e script AGI (*Asterisk Gateway Interface*) per far interagire Asterisk con diversi linguaggi di programmazione (astagidir);
- percorso di *spool*[2] utilizzato da diversi componenti base e moduli (astspooldir);
- processi lanciati in esecuzione (astrundir);
- file di log (astlogdir);
- file binari (astbindir).

Ognuna di queste directory può essere opportunamente modificata per riflettere la propria configurazione. Per sviluppare un sistema PBX base, sono necessari altri due componenti

2 Spostamento di un processo o risorsa in un buffer, di memoria o fisico, cioè su disco (come in questo caso) dove rimane in attesa di essere smistato verso l'applicativo o il dispositivo che lo dovrà elaborare.

fondamentali: il protocollo SIP (*Session Initiation Protocol*) e il *dialplan*, che verrranno esposti nei paragrafi successivi.

Session Initiation Protocol

```
[general]
;context=                ;  Default  context  for  incoming  calls.  Defaults  to
'default'
;allowguest=no            ; Allow or reject guest calls (default is yes)
                          ; If your Asterisk is connected to the Internet
                          ; and you have allowguest=yes
                          ; you want to check which services you offer everyone
                          ; out there, by enabling them in the default context (see
                                      ; below).
;match_auth_username=yes  ; if available, match user entry using the
                          ; 'username' field from the authentication line
                          ; instead of the From: field.
;allowoverlap=            ; Disable overlap dialing support. (Default is yes)
;allowoverlap=yes         ; Enable RFC3578 overlap dialing support.
                          ; Can use the Incomplete application to collect the
                          ; needed digits from an ambiguous dialplan match.
;allowoverlap=dtmf        ; Enable overlap dialing support using DTMF delivery
                          ; methods (inband, RFC2833, SIP INFO) in the early
                          ; media phase.  Uses the Incomplete application to
                          ; collect the needed digits.
```

Snippet 2: sip.conf

SIP è un protocollo di rete, definito dalla specifica *rfc3261*, utilizzato per stabilire, modificare e terminare sessioni multimediali (conferenze), come ad esempio chiamate VoIP.
Questo protocollo, viene configurato in Asterisk tramite il file *sip.conf*. Come tutti gli altri file di configurazione, anche questo file è diviso in sezioni e permette di impostare diverse opzioni, come le impostazioni di rete (indirizzo di rete locale, subnet mask, gateway, NAT), il protocollo di trasporto da utilizzare (TCP, UDP), i codec usati per le chiamate, la registrazione con il SIP provider (come accenato in precedenza, il provider che fornisce il numero telefonico) e con i vari dispositivi, il contesto in cui le chiamate in arrivo devono essere indirizzate, come gestire le registrazioni, ecc...
Come si può vedere dallo Snippet 2, il file contiene numerosi commenti e consente allo sviluppatore di documentarsi sul funzionamento di ogni opzione.

```
; ------------------------------------------------------------------------
; Include Section - put custom config files here
; (/etc/asterisk/sip/ is the dedicated dir)
; ------------------------------------------------------------------------
#include sip/sip_messagenet.conf
#include sip/sip_general_custom.conf
```

Snippet 3: sip.conf #include

Durante lo sviluppo del sistema PBX si è preferito, ove possibile, lasciare intatti i file di configurazione originali e per ogni configurazione includere uno o più file, utilizzando la direttiva *#include*.
In questo modo è possibile utilizzare il file principale come riferimento e modificare solo le opzioni necessarie. I file inclusi, infatti, vengono letti da Asterisk ed estendono il file originale. Se un'opzione è definita sia nel file originale che in quello incluso, la definizione presente in quest'ultimo ha la precedenza e viene utilizzata da Asterisk. Così facendo risulta molto più facile modificare ed aggiornare il sistema, come anche risolvere eventuali errori in fase di test e sviluppo. Questo approccio modulare è una pratica comune a qualsiasi ambiente di sviluppo, e non esclusiva di Asterisk. Come si vede dallo Snippet 3, si è creata la cartella denominata 'sip' che contiene il file di configurazione per la registrazione al provider SIP (*sip_messagenet.conf*) oltre al file contenente le impostazioni generiche del protocollo SIP che verranno utilizzate dal sistema PBX (*sip_general_custom.conf*).

Diaplan e Estensioni

Il dialplan rappresenta il cuore del PBX ed è definito nel file *extensions.conf*. Come gli altri file di configurazione, anch'esso è composto da sezioni. In questo caso, ogni sezione del file è detta *contesto*, e ogni contesto viene utilizzato dal protocollo SIP per effettuare chiamate. Si può ad esempio utilizzare il contesto [from-internal] per gestire le comunicazioni interne al PBX ed il contesto [from-abroad] per le gestire le chiamate provenienti dall'estero. Ogni contesto è definibile e personalizzabile in quanto composto da una serie di istruzioni, dette estensioni. Si può usare un'estensione per riprodurre un messaggio vocale di benvenuto, indirizzare una chiamata ad un account SIP registrato con il sistema PBX, che può essere a sua volta collegato al telefono VoIP di un'impiegato dell'azienda o alla segreteria telefonica. L'insieme dei contesti, formati da una serie estensioni definisce il diaplan del PBX. Anche in questo caso, oltre al file standard, si è incluso un file di configurazione ad hoc per gestire il diaplan relativo al sistema PBX da sviluppare.

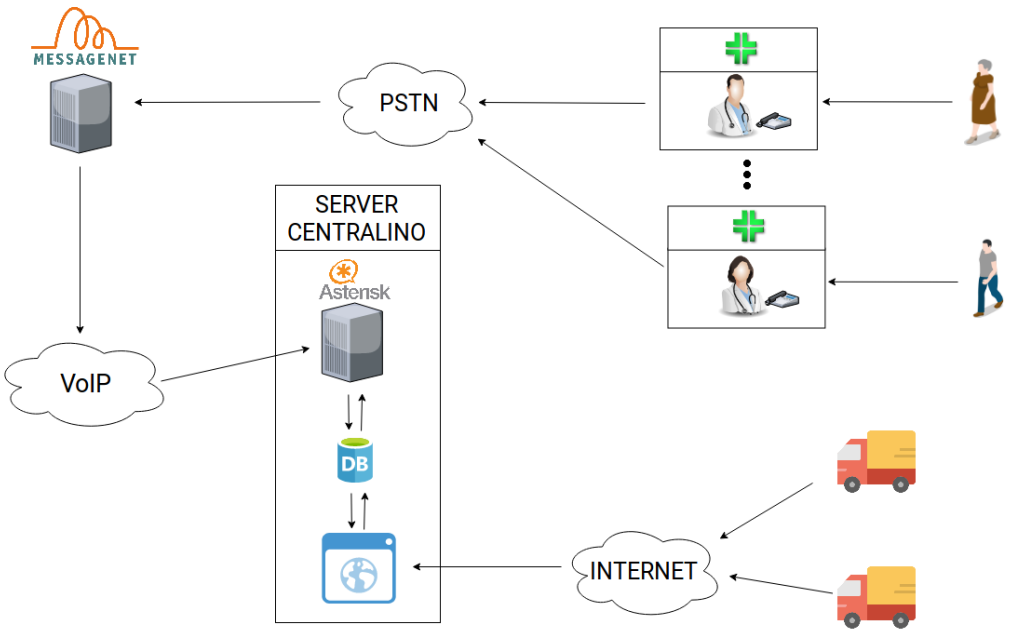
1.3 Messagenet

Figura 1: MessageNet, registrazione numerazione VoIP

Messagenet è un'azienda che fornisce servizi VoIP a privati e business, ed è il provider scelto per ricevere la numerazione telefonica da assegnare al centralino. Dato che il PBX dovrà solo ricevere chiamate, questa scelta è risultata conveniente in quanto Messagenet offre gratuitamente una numerazione italiana su cui è possibile ricevere telefonate VoIP. È sufficiente visitare l'indirizzo: https://www.messagenet.com/it/voip/free/ dal menù a destra selezionare nazione e prefesso e scegliere "Attiva ora", procedendo successivamente alla registrazione. Le credenziali di accesso a messagenet verranno inviate all'indirizzo email inserito in fase di registrazione e verranno utilizzate per collegare Asterisk alla numerazione scelta. Le chiamate in arrivo su quella numerazione saranno indirizzate da messagenet sul centralino VoIP, potendo quindi ricevere chiamate da linee telefoniche tradizionali.

1.4 Campo di Applicazione

Per concludere la parte di analisi introduttiva, resta da descrivere il contesto reale in cui si andrà a collocare il progetto e come implementare la tecnologia VoIP, utilizzando la piattaforma Asterisk per fornire le funzionalità richieste.



Schema 1: Funzionamento del centralino

Lo scopo di questo progetto è fornire ad un centro analisi un modo per semplificare e automatizzare il ritiro dei campioni, consegnati dai pazienti presso le diverse farmacie dislocate nella città. Come si può vedere dallo Schema 1, ogni paziente consegna i campioni in farmacia e ogni farmacia deve accordarsi con il centro analisi per il ritiro dei campioni, che viene gestito con diversi corrieri. La soluzione elaborata prevede due fasi: nella prima il

farmacista telefona al numero del centralino VoIP, si autentica inserendo il codice della sua farmacia e in caso di autenticazione positiva riceve la conferma di avvenuta prenotazione. L'intero processo è gestito da un menù vocale interattivo, detto *Interactive Voice Response* (IVR) che guiderà il chiamante nell'inserimento e verifica del codice. Parallelamente, ogni corriere si autenticherà alla parte web del server e riceverà un elenco dei ritiri da eseguire presso le farmacie appartenenti alla propria zona di competenza che abbiano effettuato una prenotazione per il ritiro.

La struttura medica destinataria dei campioni è il "Centro Diagnostico Cavour" di Bologna, sito in Via del Lavoro 40. Il centro è service esterno di analisi cliniche per innumerevoli strutture dislocate su Bologna e provincia. Ogni giorno convergono migliaia di provette da diversi corrieri che vengono analizzate attraverso macchinari presenti all'interno del centro. I corrieri saranno gestiti da un'entità, definita nell'interfaccia web come *manager* che potrà autenticarsi al server, visualizzare l'elenco completo dei ritiri e aggiungere, rimuovere o modificare le informazioni relative ai corrieri.

2. Software PBX

In questo breve capitolo si parlerà di FreePBX, la distribuzione che è stata inizialmente utilizzata in macchina virtuale locale (Virtualbox) per effettuare i primi test e comprendere il funzionamento di un sistema PBX.

2.1 Installazione

Tra le molteplici soluzioni diponibili, si è scelto di utilizzare FreePBX principalmente perché non è altro che una distribuzione basata su kernel Linux, CentOS, in cui vengono installati la piattaforma Asterisk, il webserver Apache (per presentare l'interfaccia grafica) oltre che diversi altri strumenti, come SSH (Secure SHell), protocollo che permette di stabilire una sessione cifrata ad un server remoto, fail2ban, un software open source che si integra con il firewall del sistema operativo e previene tentativi di accesso *bruteforce*[3] al server.

La versione utilizzata è la 10.13.66-64bit, dalla quale è possibile installare il sistema utilizzando la versione 11 o 13 di Asterisk. La iso (file di immagine che rappresenta il contenuto di un CD/DVD) è scaricabile all'indirizzo:
https://downloads.freepbxdistro.org/ISO/FreePBX-64bit-10.13.66.iso
Presumendo che VirtualBox sia già installato nel sistema, si procede a creare una nuova macchina virtuale:

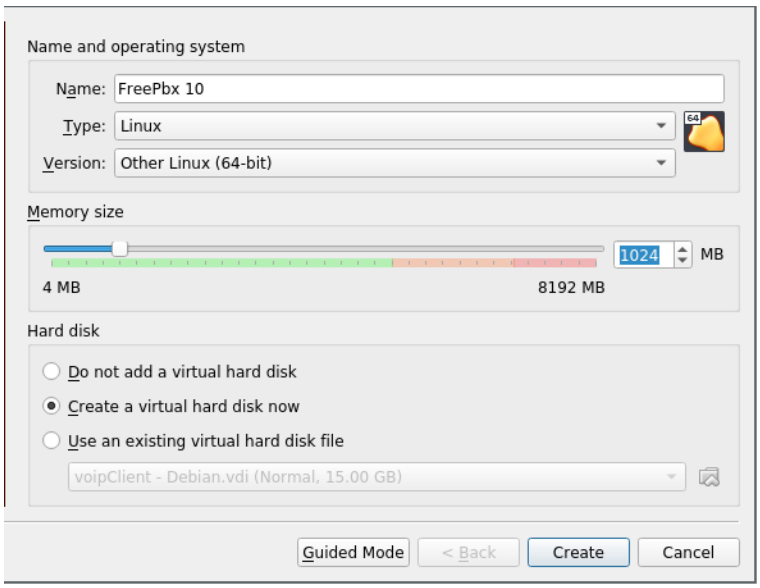


Figura 2: Virtualbox, creazione macchina virtuale.

3 Tecnica informatica che consiste nel forzare l'accesso a una risorsa provando ad autenticarsi in maniera sistematica, utilizzando ogni password presente in una lista, detta dizionario.

Si è scelto di allocare non più di 1GB di memoria RAM e nel menù seguente di assegnare 8GB di spazio sul disco da dedicare alla macchina virtuale. Queste risorse saranno più che sufficienti per effettuare le prove necessarie a capire il funzionamento del sistema PBX. Dalle impostazioni della macchina appena creata, si accede alla voce 'Storage', inserendo la iso di FreePbx precedentemente scaricata nel lettore DVD virtuale della macchina. Successivamente si può avviare la macchina e procedere con l'installazione di FreePBX.

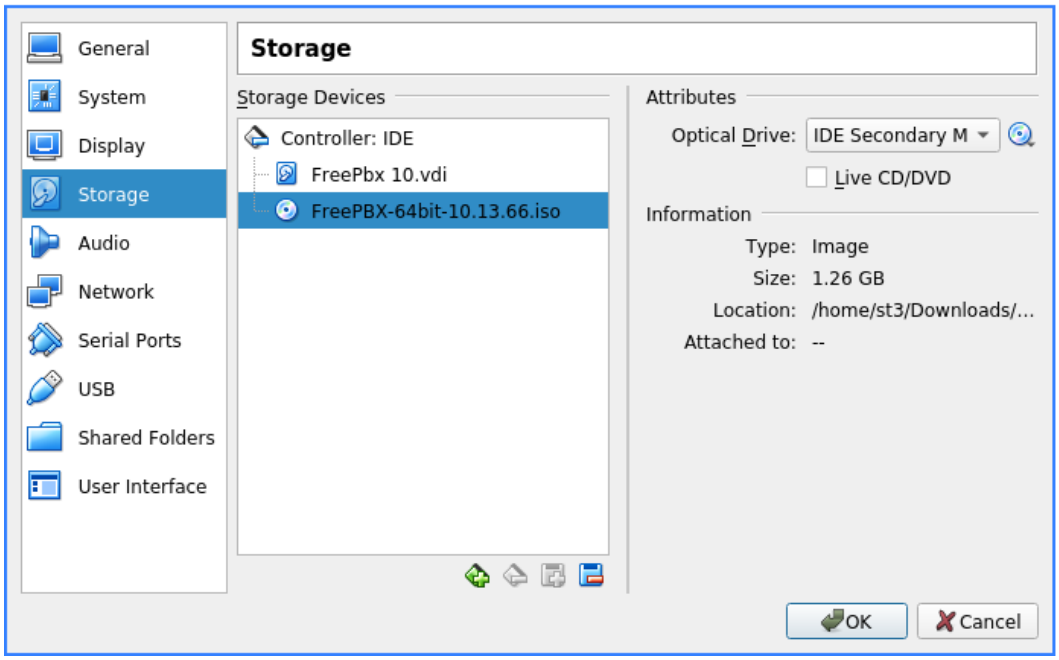


Figura 3: Virtualbox, inserimento iso nel lettore virtuale.

Una volta avviata la macchina, si seleziona l'opzione 'Full' nel menù di Asterisk 13. L'installazione è molto intuitiva e non presenta particolari difficoltà. Si seleziona il fuso orario e si imposta la password di root, dopo di che il setup procederà ad installare i pacchetti necessari e si riavvierà il sistema per completare l'installazione. Prima di avviare FreePBX è necessario tornare alle impostazioni della macchina virtuale, più precisamente alla voce 'Network' per impostare 'Bridged Adapter'. In questo modo, la macchina virtuale avrà un indirizzo IP proprio e sarà possibile, dal browser del sistema principale, accedere all'interfaccia grafica della distribuzione. A questo punto si può avviare il sistema e dopo aver effettuato l'accesso, un messaggio presenterà l'indirizzo IP della macchina e notificherà all'utente che sarà necessario attivare l'installazione, ossia registrarsi al sito di FreePBX per usufruire dell'assistenza e dei moduli commerciali. L'attivazione è facoltativa ma consigliata anche nel caso non si vogliano utilizzare moduli commerciali, principalmente perché l'utilizzo del modulo 'System Admin' è vincolato all'attivazione della macchina, ed è preferibile avere accesso al modulo, che contiene opzioni di rete avanzate.

Figura 4: FreePBX, primo accesso

Inserendo nel proprio browser l'indirizzo IP della macchina virtuale si accede all'interfaccia grafica, da cui si possono configurare tutte le opzioni del server. Dopo il setup iniziale, in cui viene chiesto di inserire nome utente, password ed indirizzo email, si procede alla configurazione del firewall Sangoma e degli indirizzi di rete. In alcuni casi si avrà l'esigenza di registrare dei telefoni al server che risiedono al di fuori della rete locale. Per fare ciò, è necessario aprire delle porte UDP per il protocollo SIP, il che è solitamente sconsigliato poiché potrebbero essere sfruttate per violare il server o causarne il crash, interrompendo il servizio. Per questo FreePBX implementa il Sangoma Firewall, che viene configurato insieme alle impostazioni di rete e che permette di segnalare le reti 'affidabili' e quelle 'non affidabili'.
La lista delle porte utilizzate da FreePBX è disponibile all'indirizzo:
https://wiki.freepbx.org/display/PPS/Ports+used+on+your+PBX
È sempre buona norma non utilizzare porte standard per i servizi di rete più utilizzati ed il protocollo SIP non fa eccezione. Per approfondire il discorso inerente alla sicurezza delle tecnologie VoIP, sono presenti diversi articoli in rete, specialmente nella documentazione di Asterisk e in quella di FreePBX.

2.2 Configurazioni Principali

Completato il setup iniziale e superate diverse pubblicità dei moduli commerciali disponibili, è possibile consultare il vastissimo menù, che permette, anche senza l'uso di alcun modulo commerciale, di configurare un sistema PBX anche molto complesso e strutturato. Vedremo di seguito come registrare delle estensioni SIP per effettuare chiamate VoIP nella rete interna, come collegare a FreePBX la numerazione VoIP fornita dal SIP provider ed infine come realizzare un menù IVR.

2.2.1 Registrazione account SIP

Dal menù ‘Settings/Asterisk SIP Settings’ è possibile configurare le opzioni del file sip.conf descritto in precedenza. Un estensione SIP viene registrata tramite ‘Applications/Extensions’ e selezionando ‘Add Extension/Add new Chan_SIP Extension’.

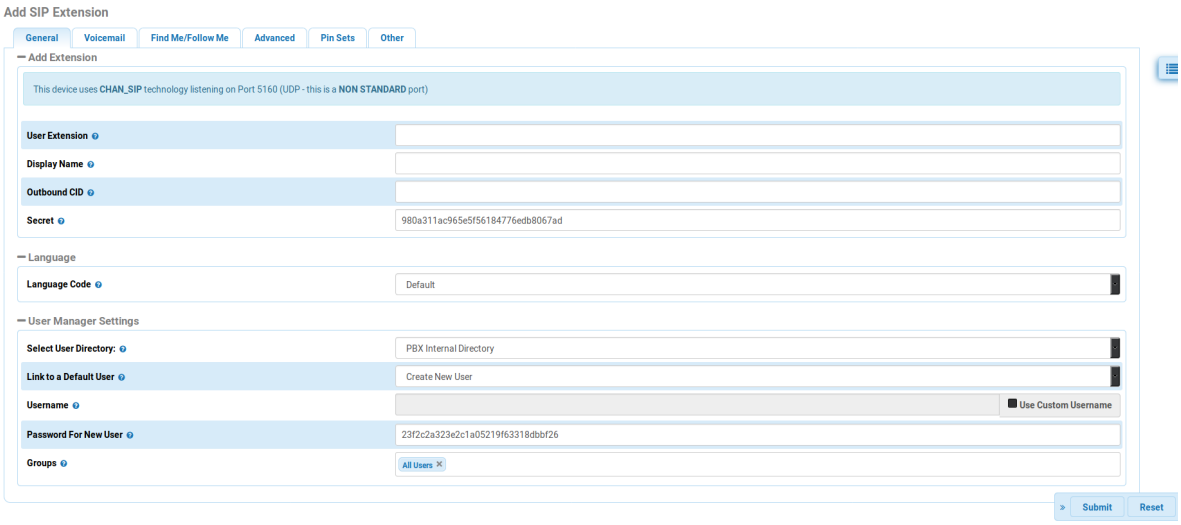


Figura 5: FreePBX, estensione Chan_SIP

Per ogni opzione è presente un punto interrogativo che ne descrive il funzionamento. Le opzioni base da configurare sono il numero dell’estensione, che sarà quello da digitare per raggiungerla, il nome dell’estensione e la password, che solitamente viene fornita da FreePBX e generata in modo casuale. Per ogni menù che si configura, bisogna confermare i cambiamenti in alto a destra cliccando ‘Apply Config’, altrimenti il sistema FreePBX non metterà in atto i cambiamenti. Eventuali errori verranno notificati preventivamente all’utente, in modo che i cambiamenti applicati non compromettano l’integrità del sistema in esecuzione.

Si possono inizialmente creare due estensioni, che dovranno essere collegate a dei dispositivi VoIP, i quali si registreranno al server tramite gli account creati. I dispositivi VoIP vengono distinti in due tipologie: *hardphone* e *softphone*. Con *hardphone* si intende un dispositivo VoIP fisico (telefono VoIP), mentre il *softphone* può essere un qualsiasi dipositivo (computer, smartphone, tablet, ecc…) su cui è installato un software *softphone*. In entrambi i casi la scelta è piuttosto ampia. Per un’installazione in ambito aziendale sarà necessario acquistare degli *hardphone* ed in questo caso utilizzare FreePBX può risultare molto comodo dato che include i firmware dei modelli più diffusi in commercio e la configurazione risulta molto facile ed intuitiva. Per effettuare dei test senza avere un *hardphone* a disposizione si può utilizzare un’applicazione di *softphone* come ‘CSipSimple’ disponibile nel Google Play Store ed

installabile su qualsiasi dispositivo Android. Il processo di configurazione è comune a tutti i softphone e prevede l'inserimento dell'indirizzo IP del server PBX, specificando la porta utilizzata dal protocollo SIP, quindi ad esempio: 192.168.1.20:5060, specificando il numero dell'estensione e la password definite in FreePBX. Fatto ciò il *softphone* risulterà registrato al server FreePBX. Aggiungendo due estensioni a due *softphone* distinti e componendo il numero dell'estensione si potranno effettuare chiamate in locale per testare il funzionamento del server.

2.2.2 Trunk SIP per il collegamento al provider

Vediamo brevemente come collegare la numerazione fornita da MessageNet a FreePBX. La voce da configurare è raggiungibile dal menù 'Connectivity/Trunks'. Un *trunk* rappresenta un collegamento tra il nostro server PBX ed un provider di telefonia VoIP. Per aggiungerlo, come fatto in precedenza con le estensioni, si seleziona la voce 'Add Trunk/Add SIP (chan_sip) Trunk'. Si imposta il nome del Trunk, che nel nostro caso può essere 'Incoming-MessageNet' sia nel menu 'General' che in 'Outgoing' e successivamente si configurano le impostazioni SIP relative al trunk nella voce 'sip Settings'; FreePBX modificherà il file sip.conf di Asterisk aggiornandolo con le impostazioni inserite. Ogni provider fornisce la documentazione necessaria per la registrazione SIP, per MessageNet questa è disponibile all'indirizzo: https://messagenet.com/it/voip/sw/asterisk.html

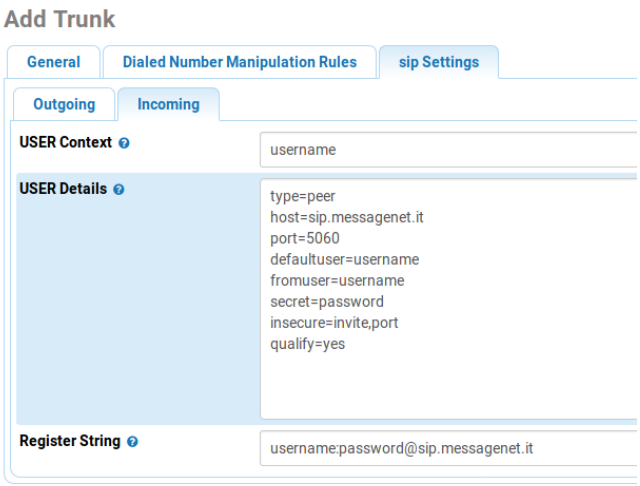


Figura 6: FreePBX, Trunk SIP/sip Settings

I parametri possono essere impostati come in Figura 6, dove 'username' e 'password' sono le credenziali inviate da MessageNet in fase di registrazione e 'port' è la porta utilizzata dal protocollo SIP nel server FreePBX. A questo punto il server FreePBX è configurato per ricevere le chiamate provenienti dalla numerazione MessageNet. Per testare la connettività, basta creare un percorso in ingresso (Inbound Route) ed assegnarlo ad una delle estensioni precedentemente create: per farlo si va su 'Connectivity/Inbound Routes/Add Inboud Route'. È sufficiente assegnare una descrizione al

percorso, che può essere ‘FromMessageNetTo201’ dove ‘201’ è il numero dell’estensione assegnata e alla voce ‘Set Destination’ selezionare ‘Extensions’ e successivamente l’estensione desiderata. Ora è possibile telefonare alla numerazione MessageNet e la chiamata verrà indirizzata all’estensione che, se registrata con un telefono (softphone o hardphone), permetterà di rispondere e gestire la chiamata.

2.2.3 Dialplan con menù IVR

Per implementare le funzionalità richieste per il PBX, bisognerà collegare la numerazione MessageNet ad un menù vocale interattivo, che andrà a sostituire il softphone degli esempi precedenti, e si occuperà di:

- leggere il codice inserito in input dall’utente;
- servendosi dell’*Asterisk Gateway Interface* (AGI), di eseguire uno script che verificherà il codice nel database del server, ed effettuerà la prenotazione nel caso il codice sia esatto o comunicherà all’utente di reinserire il codice, nel caso in cui sia errato.

Questo paragrafo descrive come collegare un server FreePBX (ed annessa numerazione VoIP) ad un dialplan Asterisk. La realizzazione e il funzionamento del dialplan, oltre che l’implementazione AGI, verranno trattate nel capitolo di configurazione di Asterisk.

Il diaplan vero e proprio andrà definito nel file disponibile al percorso:
`/etc/asterisk/extensions_custom.conf`
Questo file viene utilizzato da FreePBX per estendere `extensions.conf` tramite la direttiva #include, come descritto nel capitolo precedente. Una volta definito il dialplan, dovrà essere aggiunta la destinazione personalizzata tramite il menù ‘Admin/Custom Destinations’.

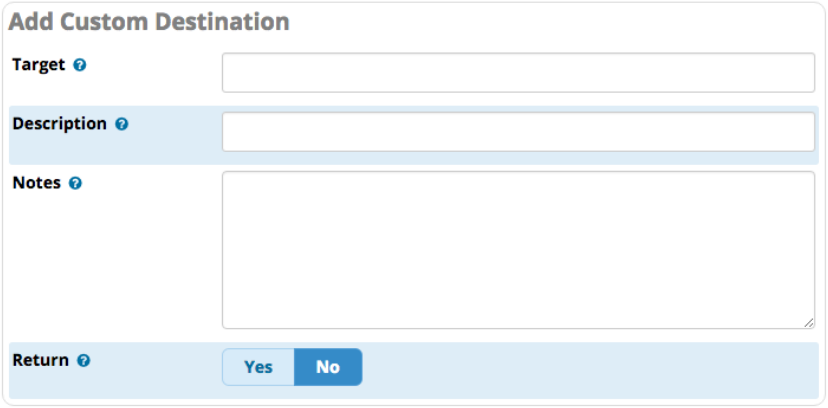


Figura 7: FreePBX, Custom Destination

In ‘Target’ va specificato il contesto del dialplan definito in `extensions_custom.conf` che deve rispettare il formato ‘contesto,estensione,priorità’ e può essere ad esempio ‘`from-incoming-messagenet, n, 1`’ ma deve coincidere con l’inizio del dialplan che si vuole utilizzare. Con ‘n’ nella sintassi del dialplan si indica ‘next’ come priorità, cioè si procede all’istruzione successiva. Come fatto in precedenza per assegnare un percorso in ingresso ad una estensione, stavolta dal menù ‘`Connectivity/Inbound Routes`’ invece di un’estensione si selezionerà ‘`Custom Destinations`’ e la destinazione appena creata. Per implementare un IVR sono necessari dei file vocali da includere nel dialplan. In FreePBX si può creare un linguaggio personalizzato tramite il menù ‘`Admin/Sound Languages/Custom Languages/Add New Custom Language`’ che permette di caricare i file audio che verranno converiti nel formato desiderato e posizionati nel percorso dei file audio definito in `asterisk.conf`.

2.3 Considerazioni su FreePBX

FreePBX risulta una soluzione molto utile per muovere i primi passi nella configurazione di un server PBX basato su Asterisk e dato che, eccezion fatta per i moduli commerciali, il resto della distribuzione è open source ed include un’installazione funzionante di Asterisk, è possibile esaminare le cartelle e le configurazioni utilizzate, costituendo materiale prezioso per lo sviluppo di un server PBX personale.
Nel caso di installazioni di sistemi PBX molto estesi, configurare ‘a mano’ tutti i file richiesti da Asterisk può risultare un’impresa complessa e sicuramente FreePBX rappresenta una soluzione molto pratica, testata e sicuramente funzionante. Per l’implementazione richiesta, tuttavia, si è preferito procedere sviluppando il server PBX da zero, dato che:

- nessun modulo commerciale verrebbe utilizzato;
- il sistema PBX da sviluppare è piuttosto semplice e non richiede la configurazione di svariati telefoni VoIP o altre funzionalità che possono essere semplificate utilizzando l’interfaccia di FreePBX;
- risulta comunque necessario configurare il dialplan e l’interfaccia AGI al di fuori del PBX e direttamente in Asterisk;
- resta comunque possibile espandere il server ed inserire nuove funzionalità senza complicarsi troppo la vita.

3. Installazione del Server

In questo capitolo e nei successivi si entrerà nel merito dell'implementazione scelta per il server PBX, in particolare questo capitolo tratterà dell'installazione dello stack LAMP su cui si baserà il server.
LAMP è acronimo di Linux, Apache, MySQL, Php e si procederà in quest'ordine a descrivere l'installazione e configurazione di ogni componente. Nei capitoli successivi si passerà a trattare la configurazione di Asterisk e dell'interfaccia web.

3.0 Ubicazione del Server

La macchina su cui verrà installato il server PBX è un server HP, localizzato in un datacenter a Bologna, nella quale è installato il software di emulazione VMWare e che già contiene diversi server virtuali necessari alla gestione del Centro Medico Cavour. L'installazione del sistema operativo verrà svolta, come visto in precedenza per FreePBX in VirtualBox, scaricando l'immagine iso del sistema, in questo caso Debian 9 Stable, ed inserendola nel lettore DVD della macchina virtuale VMWare. A questo punto è possibile, dopo aver creato un account nel server VMWare, accedere attraverso il client vSphere inserendo l'indirizzo del server, nome utente e password e procedere con l'installazione da remoto.

Figura 8: vSphere, schermata di accesso

3.1 Installazione Debian

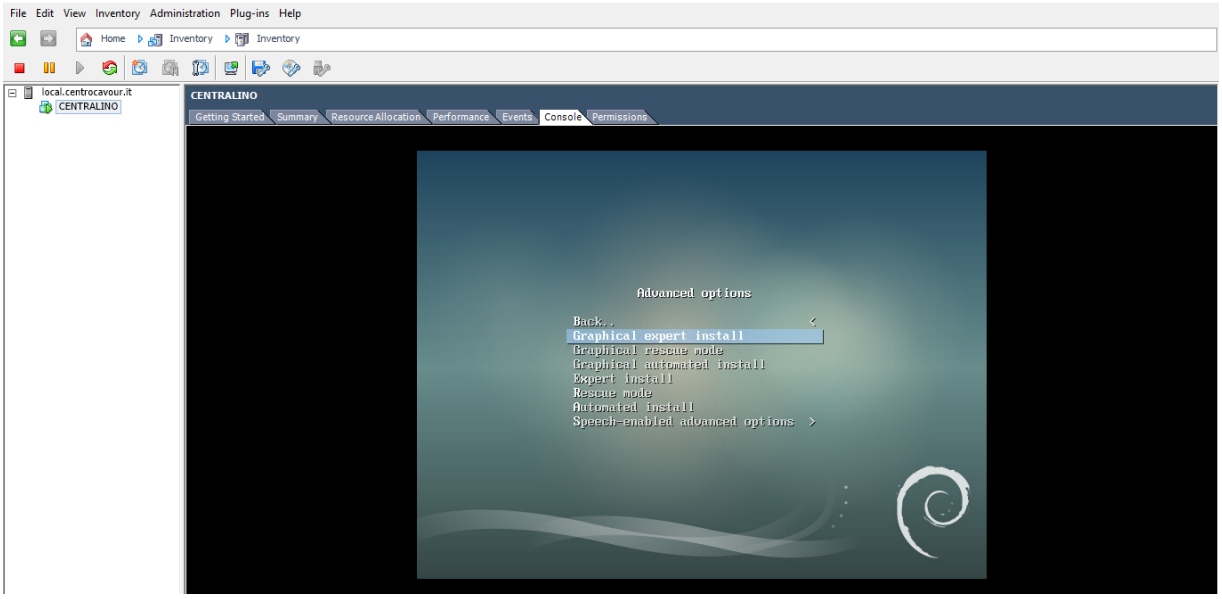


Figura 9: vSphere, installazione Debian

È stato scelto Debian poiché rappresenta nella versione '`stable`' uno dei sistemi operativi più utilizzati in ambito server. Essendo un sistema stabile, esso permette un'installazione essenziale pur avendo a disposizione innumerevoli programmi (pacchetti) da poter installare, senza doversi preoccupare di problemi di compatibilità o di assenza di documentazione. La iso scelta è la 'netinst' di Debian 9 Stable ('Stretch'), scaricabile da questo indirizzo: https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-9.4.0-amd64-netinst.iso Si è scelta l'opzione '`Graphical Expert Install`' che permette il più alto grado di configurazione e si differenza da '`Expert Install`' soltanto per un'interfaccia più leggibile, le opzioni sono identiche.

3.1.1 Lingua, Layout Tastiera e Rete

Dalla lista delle lingue si è scelto l'italiano e nel menù seguente come locale[4] :
'`it_IT.UTF-8`'
I locale sono utilizzati per definire le impostazioni di una determinata lingua, oltre che la codifica dei caratteri. Se ad esempio la lingua del locale selezionato contiene dei caratteri non ASCII, questi verranno inclusi. Allo stesso modo si seleziona il layout di tastiera italiano nel menù seguente e si procede a configurare la rete. Successivamente verrà chiesto all'utente se

4 Insieme di parametri che definiscono lingua, regione ed eventuali caratteri speciali visualizzabili dall'utente.

sarà necessario caricare componenti aggiuntivi relativi all'installazione; si potrà tranquillamente andare avanti senza selezionare componenti aggiuntivi.

Essendo in un ambiente virtualizzato, l'infrastruttura di rete è stata propriamente configurata sia a livello fisico del server che nel controller VMWare. La macchina virtuale in uso è composta da una scheda di rete ethernet virtuale, che non richiede particolari configurazioni in fase di installazione. La configurazione automatica della rete dovrebbe rilevare la scheda di rete e stabilire il collegamento internet per scaricare i pacchetti necessari all'installazione del sistema.

3.1.2 Hostname, Utenti e Fuso Orario

L'hostname è il nome che identifica il computer nella rete che, per questo server, sarà 'centralino'. L'installazione chiederà all'utente del server se abilitare le 'shadow password'. Se abilitata, questa opzione installerà la 'shadow suite' e gli hash[5] delle password degli utenti saranno memorizzati nel file `/etc/shadow` visibile soltanto dall'utente root. Con l'opzione successiva, sarà possibile disabilitare il login all'utente root. A questo punto si crea l'utente del server, denominato '`voip`', impostando successivamente la password di accesso. Questo utente, tramite il comando `sudo`, sarà abilitato ad eseguire comandi che richiedono privilegi di root. Ad esempio per installare il pacchetto '`openssh-server`' basterà digitare: `sudo apt-get install openssh-server`. Successivamente verrà richiesto di impostare il fuso orario e il server NTP (Network Time Protocol) da utilizzare.

3.1.3 Partizionamento

In un sistema Linux, una partizione è una divisione a livello logico del disco rigido.
Il punto di mount rappresenta la posizione principale a partire dalla quale un determinato filesystem[6] metterà a disposizione del sistema tutte le risorse (file e cartelle) che esso contiene. Root (indicato con '/') è il punto di mount principale, che contiene l'intero sistema, è buona norma creare più partizioni per rendere il sistema più modulare, facilitarne l'aggiornamento e la manutenzione. La tabella delle partizioni è memorizzata nel file `/etc/fstab` che viene usato dal sistema operativo in fase di avvio per assegnare ad ogni partizione il proprio punto di mount. Non avendo esigenze particolari per quanto riguarda la configurazione dei dischi (memorizzazione di file più grandi di 16TB, configurazioni RAID, ecc…) si può tranquillamente scegliere il filesystem ext4 per le varie partizioni che andranno a costituire il server. Il partizionamento che verrà utilizzato non è presente tra le scelte preconfigurate, quindi andrà selezionata l'opzione per il partizionamento manuale, scegliendo di creare una

5 Funzione non invertibile che mappa una stringa di lunghezza arbitraria ad un'altra stringa. È una misura di sicurezza comune utilizzata per memorizzare password su un server.
6 Metodo con il quale i file vengono memorizzati e organizzati nel disco.

tabella delle partizioni di tipo `msdos` (compatibile con BIOS classico) usando il disco virtuale da 34GB che si ha a disposizione. La tabella delle partizioni sarà così composta:

Punto di Mount	Filesystem	Dimensione	Flag Avviabile
/boot	Ext4	256MB	Sì
Swap	Swap	4GB	
	LVM	30.1GB	

Tabella 1: Partizionamento

La partizione di boot è contrassegnata dal flag avviabile, essendo quella in cui risiederà il bootloader. L'area di swap rappresenta una partizione del disco che, in caso di bisogno, verrà usata per trasferire porzioni della memoria RAM non richieste immediatamente dal sistema.

Il resto del disco è stato riservato come LVM, acronimo di Logical Volume Manager. Il Logical Volume Manager è uno strumento che consente di organizzare lo spazio su disco in gruppi di volumi, ognuno dei quali composto da uno o più volumi logici. Il vantaggio di questo approccio è che all'interno di un LVM è possibile ridimensionare e riorganizzare a proprio piacimento lo spazio a disposizione all'interno dei vari volumi logici. In questo modo, se una partizione in futuro richiederà dello spazio aggiuntivo, sarà possibile allocargliene senza compromettere l'integrità del sistema. L'opzione 'configurare il Logical Volume Manager' permette di creare gruppi di volumi e, per ognuno di essi, volumi logici. Dopo aver creato il gruppo 'lvm', si creano i seguenti volumi logici:

Nome	Filesystem	Punto di Mount	Dimensione
root	Ext4	/	8GB
home	Ext4	/home	4GB
var	Ext4	/var	10GB
srv	Ext4	/srv	4GB

Tabella 2: Composizione LVM

Nella partizione `/var` risiederanno i file di log dei servizi installati (Apache, MariaDB, Asterisk), la cartella `/var/lib/asterisk`, contenente il database interno della piattaforma e i file audio utilizzati, la cartella `/var/lib/mysql`, contenente il database del server. Per questo è la partizione a cui si riserverà maggior spazio. In `/srv`, invece, verranno allocati tutti i file utilizzati per l'interfaccia web del sito, oltre ai file necessari per l'implementazione della Asterisk Gateway Interface. In `/home` risiederanno i file relativi ad ogni utente del server. Non avendo necessità particolari in questo senso, si riserveranno soltanto 4GB di spazio a questa partizione. Il resto del sistema è incluso nella partizione di root. In ultimo, verranno messi da

parte 4GB di spazio non partizionato in modo da poterli allocare ai vari volumi logici, qualora ce ne fosse bisogno.

3.1.4 Installazione Sistema Base

La fase successiva dell'installazione consisterà nel selezionare il tipo di kernel da installare. Si dovrà selezionare la scelta standard '`linux-image-amd64`', che installerà l'ultima versione del kernel linux rilasciata per Debian stable. Nel menù seguente si sceglierà di installare solo driver 'mirati' per il sistema in uso, evitando così di installare pacchetti non necessari. Avendo scelto una iso minimale, composta soltanto dal programma d'installazione, si dovrà selezionare il server da cui reperire i pacchetti da installare (detto 'mirror di rete'). Selezionare il server più vicino è solitamente la scelta migliore. Se non usato, si può ignorare la parte relativa al proxy di rete e successivamente scegliere se utilizzare 'software non libero', cioè proprietario. Non escludendo una necessità futura di software proprietario e, per non dover riconfigurare il gestore dei pacchetti, si può selezionare questa opzione. I repository[7] sorgenti servono per ricevere il codice sorgente di un determinato pacchetto. Per risparmiare spazio sul disco e tempo richiesto per gli aggiornamenti si può disabilitare questa opzione.

Solitamente in un server si punta a mantenere il sistema il più stabile e immutato possibile, quindi si sceglierà di ricevere solo gli aggiornamenti di sicurezza. Nella lista del software da installare, si selezionerà soltanto 'Utilità di sistema base', non avendo necessità di interfaccia grafica o server di stampa. Ulteriori strumenti verranno installati e configurati separatamente dopo l'installazione del sistema operativo.

3.1.5 Bootloader

Il bootloader è un software che viene eseguito prima del sistema operativo, e viene utilizzato per eseguire il kernel, iniziando così il processo di avvio del sistema. Basta selezionare il disco su cui installare GRUB (il bootloader utilizzato da debian) ed il programma di installazione si occuperà del resto. Terminato questo passaggio l'installazione è conclusa. Sarà quindi possibile rimuovere la iso dal lettore virtuale di VMWare e riavviare la macchina virtuale.

3.1.6 Login e Post-Installazione

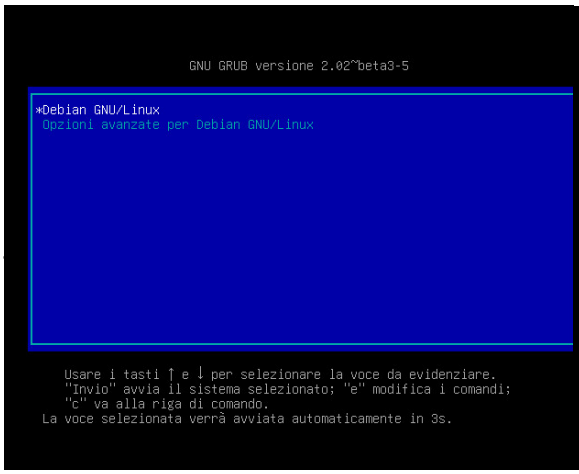


Figura 10: Debian, menù di GRUB

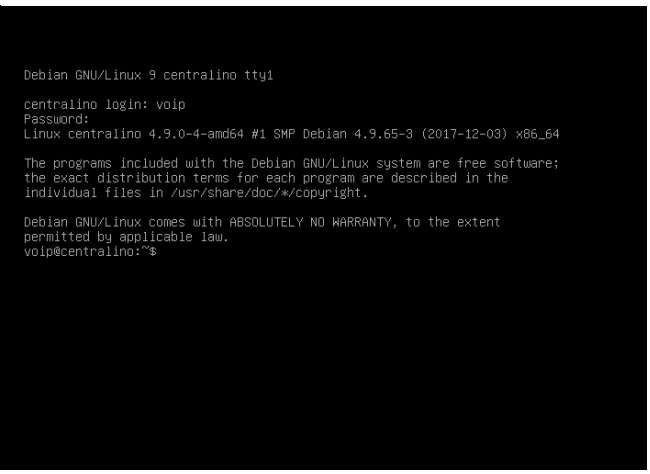


Figura 11: Debian, primo login

Riavviato il server, si potrà effettuare il primo accesso ed procedere alle configurazioni preliminari. Innanzitutto sarà necessario aggiornare i repository dei pacchetti con il comando:

```
sudo apt-get update
```

Successivamente si installeranno le utilità di rete contenute nel pacchetto net-tools e lo strumento `curl`, utile per comunicare con altri server da linea di comando:

```
sudo apt-get install net-tools curl
```

Utilizzando curl sarà possibile, ad esempio, visualizzare l'indirizzo IP esterno del server:

```
curl ipinfo.io/ip
```

Il che equivarrebbe a visitare il sito ipinfo.io/ip sul proprio browser.

Configurazione Rete

Per il corretto funzionamento di tutte le componenti del server, Asterisk in primis, sarà necessario configurare la scheda di rete in modo da ricevere un indirizzo IP statico. Per fare ciò bisognerà individuare il nome dell'interfaccia di rete con il comando:

```
ip link show
```

Le interfacce ethernet solitamente iniziano per 'e', mentre quelle wifi per 'w'. Essendo in ambiente virtualizzato, è presente una scheda di rete, in questo caso nominata '`ens192`'.

A questo punto si dovrà modificare il file /etc/network/interfaces utilizzando l'editor 'nano':

```
sudo nano /etc/network/interfaces
```

aggiungendo il seguente blocco, sostituendo gli indirizzi con la propria configurazione di rete:

```
allow-hotplug ens192
                  iface ens192 inet static
                  address 192.168.10.213
                  netmask 255.255.255.0
                  gateway 192.168.10.250
```

Snippet 4: Configurazione indirizzi di rete

L'hotplug è un evento del kernel linux che occorre ad esempio quando la macchina si avvia e viene rilevata l'interfaccia di rete, come anche quando si collega/scollega il cavo di rete.

Nanorc e Bashrc

Nella home dell'utente (`/home/voip`) sono presenti i file `.nanorc` e `.bashrc` che vengono rispettivamente letti dall'editor nano e dalla shell utilizzata dal sistema, in questo caso bash, e nel quale possono essere specificate opzioni e funzioni (nel caso del file `.bashrc`) da utilizzare per semplificare l'amministrazione del server. Per il file `.nanorc`, ad esempio si possono abilitare le seguenti opzioni:

```
set autoindent
set backup
set backupdir "/var/file_backups"
set boldtext
set casesensitive
include "/usr/share/nano/*.nanorc"
```

Snippet 5: Configurazione editor nano

In questo modo i file modificati da nano verranno automaticamente copiati nella cartella specificata (che andrà creata preventivamente), permettendo di ripristinare configurazioni precedenti in caso di errore. Tra le altre opzioni, le più importanti sono l'indentazione automatica del testo e l'inclusione dei file presenti in `/usr/share/nano` con estensione `.nanorc`, che rappresentano i file contenenti l'evidenziatore di sintassi per numerosi linguaggi di programmazione. Per evidenziare la sintassi dei file di configurazione di Asterisk è disponibile un file `.nanorc` all'indirizzo:
https://www.voip-info.org/wiki/view/Nano+syntax+highlighting
Si può copiare il contenuto in un file denominato 'asterisk.nanorc', inviarlo al server e copiarlo in `/usr/share/nano`. Nel file `.bashrc` possono essere definiti funzioni e alias per ridefinire ed espandere determinati comandi.

Ad esempio, si definiscono i seguenti alias per i comandi di base:

```
# aliases
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
alias ls='ls -l --color=auto'
alias sudo='sudo '
alias rasterisk='asterisk -rx'
```

Snippet 6: alias .bashrc

In questo modo, ogni volta che si cancellerà, sposterà o copierà un file l'opzione -i verrà invocata, chiedendo conferma dell'operazione. È necessario effettuare l'alias del comando `sudo` con '`sudo` ' (notare lo spazio) per permettere alla shell di riconoscere degli ulteriori alias dopo il comando `sudo`. L'alias `rasterisk` invoca direttamente la shell di Asterisk, permettendo di eseguirne i comandi senza accedere al prompt di Asterisk, come accade in FreePBX. Essendo questo file uno script bash/sh a tutti gli effetti, si possono definire delle funzioni, come si farebbe in un generico script bash.
Si può definire una funzione per cambiare i permessi di una cartella e di tutte le sotto cartelle a partire da una posizione di origine:

```
# change permissions to $1 for folder $2 and subfolders
function chmod_dir(){
      sudo find $2 -type d -exec chmod $1 {} \;
}
```

Snippet 7: funzione chmod_dir in .bashrc

Ad esempio, per impostare i permessi a 744 (lettura,scrittura,esecuzione per proprietario, lettura per gruppo e tutti gli altri utenti) a tutte le cartelle a partire da `/srv/centralino`, basterà eseguire nel terminale: `chmod_dir 744 /srv/centralino`

È possibile creare un'altra funzione, `chmod_files` che farà la stessa cosa per i file a partire da una directory. Basterà sostituire `-type d` con `-type f`.
Se si deciderà di abilitare il login all'account root, basterà digitare: `su` ed impostare la password. Per disabilitarlo nuovamente si potrà eseguire il comando:
```
sudo passwd -l root
```

3.2 SSH

La secure shell, come detto in precedenza, è un protocollo che permette di collegarsi ad un server remoto attraverso una connessione cifrata. È un metodo piuttosto sicuro e conveniente per amministrare server remoti. Si procederà con l'installazione e configurazione della secure shell, in modo da proseguire con il resto della configurazione mediante una connessione ssh.

3.2.1 Installazione

```
sudo apt-get install ssh
```
Con questo comando verranno installati tutti i pacchetti necessari, tra cui il client e server openssh, l'utility sftp che può essere usata per trasferire file mediante il protocollo ftp in maniera sicura, oltre che delle utility raccomandate dal gestore di pacchetti, tra cui `ufw`, 'Uncomplicated FireWall', un front-end che consente di configurare in maniera più semplice il firewall di sistema, chiamato `iptables`.
Successivamente si abilita il servizio ssh, in modo che vada in esecuzione ad ogni avvio della macchina. Questo non manderà in esecuzione il servizio subito dopo averlo installato e, per non riavviare la macchina, si avvierà manualmente, attraverso il seguente comando:
```
sudo systemctl enable ssh && sudo systemctl start ssh
```

È possibile verificare lo stato del servizio con:
```
systemctl status ssh
```
(Il comando precedente non richiede permessi di root)
Il protocollo ssh di norma utilizza la porta 22 lato server. È possibile specificare una porta differente nel file di configurazione oppure, essendo in ambiente virtuale, effettuare il redirezionamento su un'altra porta al di fuori della macchina virtuale. In questo modo, nel server la porta indicata resta la 22, ma il client per connettersi dovrà specificare la porta differente impostata.

3.2.2 Configurazione

Client

Il client è la macchina utilizzata per accedere al server e si presumerà sia anch'essa un sistema basato su kernel Linux. Sarà necessario una coppia di chiavi di 4096 byte utilizzando l'agoritmo RSA:

```
ssh-keygen -t rsa -b 4096 -f ~/.ssh/centralino.rsa
```

Questo creerà i file `centralino.rsa` e `centralino.rsa.pub` nella cartella `.ssh` presente nella home directory dell'utente corrente. Successivamente bisognerà creare un file di configurazione che sostanzialmente permetterà di connettersi al server utilizzando un alias, in modo tale da non dover specificare tutti i parametri ad ogni connessione.

```
nano ~/.ssh/config
```

```
Host centralino
     Hostname ip_esterno_del_server
     User voip
     PubKeyAuthentication yes
     IdentityFile ~/.ssh/centralino.rsa
     Port porta_ssh_del_server
```

Snippet 8: File di configurazione locale per ssh

Si copierà la chiave pubblica nel server con:

```
ssh-copy-id -i ~/.ssh/centralino.rsa.pub -p porta_del_server voip@ip_esterno_del_server
```

Verrà chiesta la password dell'utente voip. A questo punto sarà possibile autenticarsi al server digitando:

```
ssh centralino
```

Se nella fase di creazione della chiave è stata impostata la passphrase (pratica consigliata), bisognerà, ad ogni accesso, digitarla. Per evitare di digitare più volte nella stessa sessione la passphrase si può utilizzare l'utility `ssh-agent` che mantiene in memoria RAM la passphrase per la durata della sessione. Basterà avviare `ssh-agent` con:

```
eval "$(ssh-agent -s)"
```

e aggiungere la chiave digitando:

```
ssh-add ~/.ssh/centralino.rsa
```

Per terminare la sessione e 'dimenticare' la passphrase si potrà eseguire:

```
ssh-agent -k
```

Server

Nel server è necessario modificare il file di configurazione di ssh, per migliorare la sicurezza del protocollo:

```
sudo nano /etc/ssh/sshd_config

PermitRootLogin no
StrictMode yes
PubKeyAuthentication yes
PasswordAuthentication no
AllowGroups ssh
```
Snippet 9: file sshd_config

Con queste opzioni viene disabilitato il login all'account `root` tramite ssh, abilitata l'autenticazione usando la chiave pubblica caricata in precedenza, disabilitata l'autenticazione tramite password e viene permesso di autenticarsi al server solo attraverso utenti che appartengano al gruppo `ssh`. Di seguito viene creato il gruppo ssh e viene aggiunto l'utente `voip` al gruppo `ssh` con:
`sudo groupadd ssh && sudo gpasswd -a voip ssh`
Infine si riavvia il servizio ssh con il comando: `sudo systemctl restart ssh`
Per aggiungere un nuovo utente al server, copiando la chiave pubblica con `ssh-copy-id`, è necessario abilitare temporaneamente l'accesso mediante password. In alternativa si può copiare la nuova chiave 'manualmente' con il comando:
`cat ~/.ssh/nuova_chiave.pub | ssh centralino "cat >> ~/.ssh/authorized_keys"`
il quale legge il contenuto della chiave da copiare e la copia in fondo al file delle chiavi autorizzate nel server remoto. Può essere 'tradotto' in:
`Prendi il contenuto della nuova chiave | connettiti al server centralino "aggiungi la nuova chiave in fondo al file delle chiavi autorizzate"`

Come ultima raccomandazione è bene impostare i permessi, sia nel client che nel server, per la cartella .ssh e i file che la compongo come segue:
`sudo chmod 700 ~/.ssh && sudo chmod 600 ~/.ssh/*`
In questo modo solo i rispettivi proprietari avranno i permessi di lettura e scrittura per le chiavi (è necessario impostare il permesso di esecuzione nella cartella per permettere l'accesso alla stessa).

3.3 Apache + Php

Si procede con l'installazione del web server Apache e del il supporto al linguaggio Php, che è stato scelto per lo sviluppo dell'interfaccia web, principalmente per la sua semplicità di installazione e sviluppo e anche per la possibilità di utilizzarlo con l'Asterisk Gateway Interface, per permettere la comunicazione del server PBX con il database che verrà creato per gestire l'interfaccia web. Il comando per installare il server Apache e Php in Debian è il seguente (Apache ed i moduli per integrare Php verranno installati automaticamente come dipendenze): `sudo apt-get install php7.0 php7.0-mysql`

3.4 MariaDB

MariaDB è un Database Management System che permette di creare e gestire database MySQL.
È possibile installarlo tramite il comando:
`sudo apt-get install mariadb`
Per impostare la password di root che verrà usata per accedere a MariaDB verrò eseguito:
`sudo mysql_secure_installation`

3.5 Uncomplicated FireWall

`ufw` è un programma che semplifica la creazione e la gestione di regole per il firewall IPTables. Di seguito si vedrà come installarlo e configurarne le regole base.
Installazione:
`sudo apt-get install ufw`
Regola predefinita per bloccare tutte le connessioni in ingresso:
`sudo ufw default deny incoming`
Regola predefinita per consentire tutte le connessioni in uscita:
`sudo ufw default allow outgoing`
Regola per permettere ssh:
`sudo ufw allow ssh`
Regola per permettere apache:
`sudo ufw allow www`
Per abilitare il firewall (resterà attivo anche dopo un eventuale riavvio del sistema):
`sudo ufw enable`
Per visualizzare in dettaglio le regole:
`sudo ufw status verbose`
Per cancellare una regola si usa il comando:
`sudo ufw delete nome_regola`
In alternativa, si può visualizzare una lista numerata delle regole:
`sudo ufw status numbered`
e, ad esempio, per cancellare la regola `[1]`:
`sudo ufw delete 1`

Ulteriori informazioni si possono trovare all'indirizzo:
https://wiki.archlinux.org/index.php/Uncomplicated_Firewall
In alternativa con il comando: `man ufw`

4. Installazione di Asterisk

In questo capitolo verranno esplicate le procedure di installazione di Asterisk sul server, configurandone tutti gli aspetti, al fine di rendere il server Debian un server PBX con le funzionalità richieste.

4.1 Installazione, asterisk.conf

Per installare Asterisk e i pacchetti necessari al suo funzionamento dovrà essere eseguito il seguente comando:

```
sudo apt-get install asterisk dahdi libpri1.4
```

Si aggiunge l'utente 'voip' al gruppo 'asterisk', in modo da poter modificare i file di configurazione senza utilizzare il comando 'sudo':

```
sudo gpasswd -a voip asterisk
```

Il file asterisk.conf verrà modificato nel modo seguente:

```
nano /etc/asterisk.conf
```

```
[directories]
astetcdir => /etc/asterisk
astmoddir => /usr/lib/asterisk/modules
astvarlibdir => /var/lib/asterisk
astdbdir => /var/lib/asterisk
astkeydir => /usr/share/asterisk
astdatadir => /usr/share/asterisk
astagidir => /srv/agi
astspooldir => /var/spool/asterisk
astrundir => /var/run/asterisk
astlogdir => /var/log/asterisk
astsbindir => /usr/sbin

[options]
languageprefix = yes
execincludes = yes
```

Snippet 10: asterisk.conf nel server centralino

Sono stati definiti percorsi personalizzati per alcune funzionalità di Asterisk e le due opzioni nell'apposita sezione permettono di usare la direttiva `#include` nei file di configurazione e di aggiungere linguaggi audio personalizzati.

4.2 Gestione del Servizio e Introduzione alla Console:

Dopo l'installazione, il servizio di Asterisk dovrebbe già essere configurato per l'esecuzione all'avvio del sistema. È possibile verificarlo con il comando:
`systemctl list-unit-files | grep asterisk`

Si verifica che sia in esecuzione con:
`systemctl status asterisk`

Ed eventualmente sarà possibile avviare il servizio digitando:
`sudo systemctl start asterisk`

In seguito ad ogni modifica alla configurazione di Asterisk è bene riavviare la piattaforma, utilizzando:
`sudo systemctl restart asterisk`

In alternativa si può usare la console di Asterisk, per il:

- riavvio immediato dell'intera piattaforma:
  `rasterisk 'core restart now'`
- riavvio '`graceful`', Asterisk non interromperà le chiamate in corso ma impedirà la ricezione di nuove chiamate. Quando tutte le chiamate in corso saranno terminate, riavvierà la piattaforma: `rasterisk 'core restart gracefully'`

Se si vuole evitare di riavviare l'intera piattaforma e avendo modificato la configurazione di uno o più moduli, ad esempio '`chan_sip`' (implementazione del protocollo SIP utilizzata in questa installazione) e '`res_agi`' (modulo utilizzato per implementare l'Asterisk Gateway Interface), è possibile riavviare i singoli moduli attraverso il comando: `sudo rasterisk 'module reload chan_sip res_agi'`

Per una lista di tutti i moduli installati:
`sudo rasterisk 'module show'`

Per cercare un modulo particolare tramite parola chiave:
`sudo rasterisk 'module show like parola_chiave'`

Questa è solo una piccola parte dei comandi che la console di Asterisk è in grado di interpretare. Altri comandi verranno utilizzati nel corso della configurazione; è possibile consultare la documentazione di Asterisk per una panoramica più ampia di tutti i comandi disponibili.

4.3 Protocollo SIP

In questa sezione si andrà a modificare il file di configurazione `/etc/asterisk/sip.conf`, includendone i file che definiranno la configurazione sip generale e del provider MessageNet:
`nano /etc/asterisk/sip.conf`

```
#include sip/sip_messagenet.conf
#include sip/sip_general_custom.conf

[general]
...
```

Snippet 11: Asterisk, direttiva include in sip.conf

In seguito verrà creata la cartella con il comando:
`mkdir /etc/asterisk/sip`

Il file `sip_general_custom.conf`:
`nano /etc/asterisk/sip/sip_general_custom.conf`

```
; Custom General Settings File for the Asterisk PBX Settings
; -----------------------------------------------------------
; context
context=from-incoming-messagenet

; codecs
disallow=all
allow=alaw
allow=ulaw
allow=gsm
allow=ilbc

; sip
registertimeout=20 ; retry registration calls every x seconds (default 20)
registerattemps=10 ; number of times to try for a sip registration - 0=forever
(default 0)

; rtp
rtptimeout=30 ; hang call if no RTP is received in 30 seconds
rtpholdtimeout=300 ; hang call if RTP stream stops for 300 seconds

; security
allowguest=no ; disable anonymous sip calls to the asterisk server
alwaysauthreject=yes ; reject bad passwords attemps on valid usernames, answering as
if the username was wrong too

; network
bindaddr=0.0.0.0 ; let asterisk listen on any interface
localnet=192.168.10.250/24 ; local network address
```

```
bindport=5060  ; default sip port
externip=ip_esterno ; external ip of the server
```
Snippet 12: Asterisk, sip_general_custom.conf

I commenti inseriti (preceduti da ';') illustrano ogni impostazione. Il file imposta il contesto delle chiamate in arrivo a '`from-incoming-messagenet`' che verrà poi definito nel diaplan. Si definiscono i codec da utilizzare, le impostazioni relative alla registrazione sip, le impostazioni consigliate per la sicurezza: in particolare verranno disabilitate le chiamate anonime al server PBX e non verrà specificato, in caso di autenticazione errata, quale dei due campi (username e password) sia sbagliato; in questo modo un possibile attacco bruteforce non potrà escludere che anche il nome utente (il nome dell'estensione SIP) sia errato, riducendo le possibilità di riuscita dell'attacco. Inoltre, verranno configurate le impostazioni di rete: l'indirizzo del gateway di rete, la porta da dedicare al protocollo SIP (come già detto, nel caso in cui fosse necessario utilizzarla all'esterno della rete locale è consigliabile non utilizzare le standard 5060, 5061, 5160, 5161) e l'indirizzo esterno del server PBX. Il protocollo RTP (Real Time Transport Protocol) è utilizzato in un sistema PBX per lo scambio dei pacchetti multimediali (audio). Nel file di configurazione sip vengono impostati i limiti di tempo per la ricezione di pacchetti RTP, oltre i quali il protocollo SIP terminerà la chiamata.

Configurazione delle impostazioni per il provider MessageNet:
`nano /etc/asterisk/sip/sip_messagenet.conf`

```
; MessageNet voip service trunk configuration file
;-------------------------------------------------------

; registers the messagenet number
[general]
register => user:password@sip.messagenet.it

; defines how to communicate with sip.messagenet.it
[MessageNet]
type=peer
host=sip.messagenet.it
defaultuser=username
fromuser=username
secret=password
insecure=invite,port
qualify=yes
context=from-incoming-messagenet ; redirect incoming calls to the specified
context
```
Snippet 13: Asterisk, sip_messagenet.conf

Nel contesto `[general]` viene definita la procedura per effettuare la registrazione al provider e nel contesto `[MessageNet]` vengono definite le impostazioni necessarie per stabilire la comunicazione con il provider.

4.4 Dialplan

A questo punto il server PBX è pronto a ricevere chiamate sulla numerazione MessageNet, bisognerà quindi implementare un dialplan che gestisca le chiamate in arrivo.
Per prima cosa, verrà modificato il file `/etc/asterisk/extensions.conf`, includendo il file che definirà il dialplan:

```
nano /etc/asterisk/extensions.conf

#include dialplans/dialplan_centralino.conf

[general]
...
```

Snippet 14: Asterisk, extensions.conf

Successivamente verrà creata la cartella:

```
sudo mkdir /etc/asterisk/dialplans
```

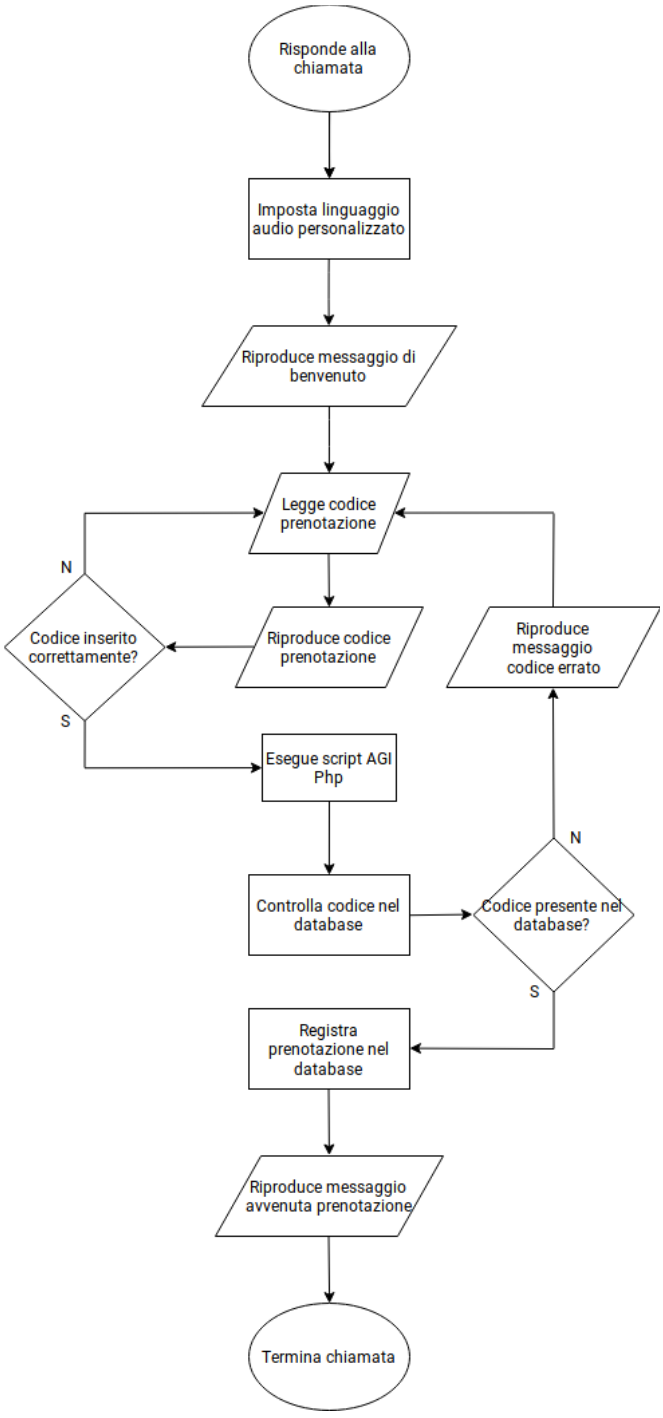
ed il file del diaplan:

```
nano /etc/asterisk/dialplans/dialplan_centralino.conf

; Custom Dialplan for our server application
; -------------------------------------------------------------
[from-incoming-messagenet]
; answer incoming call
exten => s,1,Answer
exten => s,n,Wait(2)
;    set    chan    language    to    custom    'mc'    sound    language    (files    in
/var/lib/asterisk/sounds/mc)
exten => s,n,Set(CHANNEL(language)=mc)
; play welcome message
exten => s,n,Playback(custom/welcome-general)
exten => s,n,Wait(2)
; jump to reservation context
exten => s,n,Goto(reservation,s,input-code)
; play goodbye message
exten => s,n(goodbye-general),Wait(1)
exten => s,n,Playback(custom/goodbye-general)
; hangup the call
exten => s,n,Hangup
; reset to default chan language
exten => s,n,Set(CHANNEL(language)=)

[reservation]
```

```
; read the code (3 tries, 10 second wait each try)
exten => s,1(input-code),Read(userCode,custom/input-code,,,3,10)
; read the code     back to      the user
exten => s,n,Playback(custom/output-code)
exten => s,n,SayDigits(${userCode})
; let the user confirm the code, or correct if wrong (1 try, 5 second wait)
exten => s,n(confirm-code),Read(confirmCode,custom/confirm-code,,,1,5)
; jump back to input-code if wrong, otherwise go to check-code
exten => s,n,GotoIf($[ "${confirmCode}" = "1" ]?check-code:input-code)
; call php script to check the code
exten => s,n(check-code),AGI(scripts/agi_reservation.php,${userCode})
; jump to success if right code, otherwise check the error type
exten => s,n,GotoIf($[ "${result}" = "1" ]?success-reservation:error-check)
; if query error go to error-general, otherwise go to error-code
exten => s,n(error-check),GoToIF($[ "${result}" = "2" ]?error-general:error-code)
; play error-code message and jump back to input code
exten => s,n(error-code),Playback(custom/error-code)
exten => s,n,Goto(reservation,s,input-code)
; play error-general message and jump to goodbye
exten => s,n(error-general),Playback(custom/error-general)
exten => s,n,Goto(from-incoming-messagenet,s,goodbye-general)
; play success-reservation message and jump to goodbye
exten => s,n(success-reservation),Playback(custom/success-reservation)
exten => s,n,Goto(from-incoming-messagenet,s,goodbye-general)
```

Snippet 15: Asterisk, dialplan_centralino.conf

Questo file è ciò che definisce il comportamento del server PBX richiesto. Il suo funzionamento, già documentato dai commenti, può essere descritto tramite il diagramma di flusso della pagina seguente.



Schema 2: Asterisk, Diagramma di flusso del dialplan

La sintassi del dialplan è la seguente:
`exten => nome,priorità,comando`

Con ‘`s`’ si indica una generica estensione, utile quando non si deve passare la chiamata ad una specifica estensione sip collegata ad un telefono VoIP, come nel caso del centralino da implementare. La priorità ‘`n`’ sta per ‘next’ e punta all’istruzione successiva. È possibile assegnare delle etichette alle priorità, in modo da rendere il diaplan più leggibile e attraverso il comando: `Goto(contesto,nome,etichetta)`
È possibile eseguire direttamente l’istruzione con una determinata etichetta, evitando di procedere in sequenza e potendo implementare dei salti condizionali, come avviene in alcuni linguaggi di programmazione (ad esempio nel linguaggio Assembly).

I comandi utilizzati nel dialplan sono i seguenti:
`Answer`
Risponde alla chiamata

`Wait(secondi)`
Resta in attesa per i secondi specificati

`Hangup`
Termina la chiamata

`Set(CHANNEL(language)=lingua)`
Imposta la lingua alla sigla specificata es. ‘`it`’

`Playback(tipo_audio/nome_messaggio)`
Riproduce un file audio del linguaggio selezionato.
Ad esempio, `custom/welcome_message` per il linguaggio ‘`it`’ si troverà in:
`/var/lib/asterisk/sounds/it/custom/welcome_message.wav`

`Read(variabile,tipo_audio/`
`nome_messaggio,max_cifre,,tentativi,secondi_attesa)`
Riceve l’input dall’utente, memorizzandolo nella variabile specificata, riproducendo il messaggio audio. Il campo ‘`max_cifre`’ indica la lunghezza massima in cifre del codice da leggere, ‘`tentativi`’ indica il numero di tentativi massimi e ‘`secondi_attesa`’ i secondi da attendere per ricevere l’input. Si noti che è presente un campo non impostato, delimitato da ‘`,,`’; il campo può assumere tre valori: `s, n, i` che stanno per ‘`skip`’, ‘`noanswer`’, `indication`’. ‘`Skip`’ ritorna al dialplan se la linea non è attiva, ‘`Noanswer`’ tenta di leggere il codice anche se la linea non è attiva, ‘`indication`’ sostituisce il messaggio di input con un tono telefonico.

`SayDigits(${variabile})`
Legge i numeri che compongono la variabile indicata. Il linguaggio selezionato deve contenere una cartella '`digits`' contente i file audio per ogni cifra numerica.

`GotoIF($[ "${variabile}" = "valore" ]?etichetta1:etichetta2)`
Se '`variabile`' è uguale a '`valore`' il dialplan si sposta su '`etichetta1`', altrimenti va ad '`etichetta2`'.

`AGI(/percorso/script.php,${variabile})`
Tramite l'Asterisk Gateway Interface, invoca lo script indicato, passando '`variabile`' come argomento.

4.5 Suoni

Per definire i suoni si può decidere di registrare dei messaggi audio personali o si possono utilizzare degli strumenti gratuiti per il Text-To-Speech, disponibili anche online, come: `http://fromtexttospeech.com` che è stato utilizzato per realizzare i file audio del server PBX. I file dovranno essere converiti in formato wav e copiati nella cartella '`sounds`' che dovrà trovarsi al percorso specificato in `asterisk.conf`. In questo caso è stato definito il linguaggio '`mc`' (medcenter) e i file sono organizzati in questo modo:

```
/var/lib/asterisk/sounds/
├── custom
└── mc
    ├── custom
    │   ├── confirm-code.wav
    │   ├── error-code.wav
    │   ├── error-general.wav
    │   ├── goodbye-general.wav
    │   ├── input-code.wav
    │   ├── output-code.wav
    │   ├── success-reservation.wav
    │   ├── welcome-general.alaw
    │   └── welcome-general.wav
    └── digits
        ├── 0.wav
        ├── 1.wav
        ├── 2.wav
        ├── 3.wav
        ├── 4.wav
        ├── 5.wav
        ├── 6.wav
        ├── 7.wav
        ├── 8.wav
        └── 9.wav

4 directories, 19 files
```

Figura 12: Asterisk, vista ad albero di /var/lib/asterisk/sounds

L'output di Figura 12 è riproducibile installando l'utility '`tree`' (`sudo apt-get install tree`), che prende come argomento un percorso e stampa la vista ad albero a partire dal percorso indicato, mostrando tutti i file e le cartelle che lo compongono.

4.6 AGI

```
/srv/agi/
├── class
│   ├── centralino-Reservation.php
│   ├── phpagi-asmanager.php
│   └── phpagi.php
└── scripts
    └── agi_reservation.php

2 directories, 4 files
```

Figura 13: Asterisk, vista ad albero /srv/agi

L'Asterisk Gateway Interface è uno strumento che permette di eseguire uno script in un linguaggio supportato (in questo caso Php). Sarà possibile passare delle variabili allo script che a sua volta potrà crearne di nuove, che saranno poi disponibili al dialplan.

Nella cartella '`class`' sono definite le classi per l'implementazione AGI del linguaggio php. Un'implementazione funzionante, open source e abbastanza aggiornata, è quella sviluppata dalla compagnia di telecomunicazioni 'MCN Telecom' e resa disponibile all'indirizzo: https://github.com/welltime/phpagi
I file necessari sono `phpagi.php` e `phpagi-asmanager.php`, dal server si possono reperire con:

```
wget -P /srv/agi/ \
https://raw.githubusercontent.com/welltime/phpagi/master/phpagi.php        \
https://raw.githubusercontent.com/welltime/phpagi/master/phpagi-
asmanager.php
```

Il file `centralino-Reservation.php` è una classe che è stata implementata per relazionarsi con la classe Database, di cui si parlerà nel capitolo relativo all'interfaccia web. Questa classe viene utilizzata nello script `agi_reservation.php` per effettuare una prenotazione, cioè una query `INSERT` nel database. Di seguito il costruttore della classe e i prototipi dei metodi che essa utilizza:

```
/**
 *
 * Reservation constructor
 *
 * $code int is the only parameter needed
 *
 */
public function __construct(){
```

```
        $args = func_get_args();
        $this -> code = $args[0];
        $this -> db = new Database();
    }
```

Snippet 16: Php, costrutture della classe “Reservation”

```
    /**
     *
     * Check Code method - checks if the code is valid in the database
     * returns the pharmacyId if code matches
     *
     * @return pharmacyId | false bool
     *
     */
    public function checkCode(){
    }

    /**
     *
     * NewReservation Method - inserts the reservation in the database
     *
     * @param id int - the pharmacy id
     * @return true | false bool
     *
     */
     public function newReservation($id){
    }
```

Snippet 17: Php, protitipi dei metodi della classe Reservation

Entrambi i metodi si collegano al database; il primo, attraverso una query `SELECT` controlla che il codice inserito sia presente nella tabella della farmacia, mentre il secondo effettua la prenotazione per mezzo di una query `INSERT`.

Lo script `agi_reservation.php`, eseguito dal dialplan è riportato di seguito:

```
#!/usr/bin/env php
<?php

/**
*     this script gets the code from the user
*     and checks it against the database
*
*     if the code is correct, creates a new reservation and returns "true"
*     if the code is not found in the database returns "false"
*     if there was an error executing the insert query returns "error"
*
*     @return "true" | "false" | "error" using AGI set_variable() function
```

```
*/

// required for asterisk communication
require_once "/srv/agi/class/phpagi.php";

// required to make a reservation
require_once "/srv/agi/class/centralino-Reservation.php";

// get code from user input
$code = $argv[1];
// this variable will be passed back to the asterisk dialplan
$exit_code = 0;

// create new reservation object passing the code from the user
$reservation = new Reservation($code);
// if the code is present in the database make the reservation
$id = $reservation -> checkCode();
if ($id){
      // create a new reservation in the Pickup table
      if($reservation -> newReservation($id) )
            // if the insert was successfull, return "true"
            $exit_code = 1;
      else
            // if there was a problem with the query, return "error"
            $exit_code = 2;
}

// create AGI object and return the exit code to the calling dialplan
$agi = new AGI();
$agi -> set_variable("result", $exit_code);
?>
```

Snippet 18: Php, script agi_reservation.php

Si noti l'intestazione dello script: `#!/usr/bin/env php`
Questo è necessario perché Asterisk, attraverso l'Asterisk Gateway Interface, interpreta qualsiasi script come uno script bash/sh (compatibile cioè con la shell del sistema operativo), è quindi necessario puntare all'interprete desiderato, in questo caso Php. Lo script riceve il codice dal dialplan, lo controlla nel database istanziando un oggetto di tipo `Reservation` e crea la prenotazione, attribuendo alla variabile di ritorno 1 in caso di successo e 2 in caso di fallimento. Prima di terminare lo script, si crea un oggetto di tipo `AGI`, attraverso il quale verrà chiamato il metodo '`set_variabile`', che a sua volta creerà la variabile '`result`', attribuendogli il valore di ritorno dello script. Il dialplan riceverà la variabile '`result`' con il risultato dell'operazione.

4.7 Moduli e Permessi

4.7.1 Moduli inutilizzati

È possibile evitare di caricare moduli di Asterisk non necessari al funzionamento del server PBX, modificando il file `/etc/asterisk/modules.conf` come segue:
`nano /etc/asterisk/modules.conf`

```
; modules.conf - the Asterisk module loading configuration

[modules]
; autoload
autoload=yes

; preload
preload = pbx_config.so
preload = func_db.so
preload = res_odbc.so
preload = res_config_odbc.so

; load
load = format_wav.so
load = format_pcm.so

; noload
noload = pbx_gtkconsole.so
noload = pbx_kdeconsole.so
noload = app_intercom.so
noload = chan_modem.so
noload = chan_modem_aopen.so
noload = chan_modem_bestdata.so
noload = chan_modem_i4l.so
noload = chan_capi.so
noload = chan_alsa.so
noload = chan_oss.so
noload = cdr_sqlite.so
noload = app_directory_odbc.so
noload = res_config_pgsql.so
noload = cdr_radius.so
noload = cel_radius.so
noload = res_phoneprov.so
noload = res_config_ldap.so
noload = cel_custom.so
noload = cdr_pgsql.so
noload = cel_pgsql.so
noload = cel_tds.so
noload = chan_pjsip.so
noload = res_pjsip_phoneprov_provider.so
```

Snippet 19: Asterisk, modules.conf

In questo file vengono definiti i moduli da caricare prima dell'avvio di Asterik (‘preload‘), da caricare in automatico (‘autoload=yes‘ carica in automatico i moduli standard e quelli definiti con ‘load’) e da non caricare (‘noload‘).

4.7.2 Permessi di file e cartelle

Asterisk è presente nelle directory definite in asterisk.conf nel caso di questa installazione:

```
/etc/asterisk
/usr/lib/asterisk/modules
/var/lib/asterisk
/var/lib/asterisk
/usr/share/asterisk
/usr/share/asterisk
/srv/agi
/var/spool/asterisk
/var/run/asterisk
/var/log/asterisk
```

Ognuna di queste directory, ogni file e sottodirectory, deve avere come proprietario l'utente asterisk ed appartenere al gruppo asterisk. È possibile implementare uno script o una funzione in .bashrc che svolga questo compito. Il comando da utilizzare, per ogni cartella, è il seguente:

```
sudo chown -R asterisk:asterik /percorso/directory
```

I permessi che si andranno ad impostare per i file sono di lettura e scrittura per il proprietario ed il gruppo, nessun permesso per gli altri utenti:

```
chmod_files 660 /percorso/directory
```

Per le cartelle, i permessi saranno di lettura, scrittura ed esecuzione per il proprietario, lettura ed esecuzione per il gruppo, nessun permesso per gli altri utenti:

```
chmod_dir 750 /percorso/directory
```

4.8 File di Log e Debug

Le impostazioni relative ai file di log vengono configurate nel file:

```
/etc/asterisk/logger.conf
```

È possibile includere una configurazione personalizzata usando la direttiva:

```
#include logger_custom.conf
```

Successivamente si modificherà il file /etc/asterisk/logger_custom.conf in questo modo:

```
nano /etc/asterisk/logger_custom.conf
```

```
; Custom logger configuration file
;-----------------------------------
```

```
;logging settings
[general]
dateformat=%F %T
appendhostname=no
rotatestrategy=rotate

;logfiles settings
[logfiles]
full => info,debug,error,notice,verbose,warning,security
console => debug,error,notice,verbose,warning
centralino => notice,warning,error
```
Snippet 20: Asterisk, logger_custom.conf

Questo file imposta il formato di data e ora allo standard ISO 8601, non viene incluso l'hostname e viene impostata la rotazione dei file di log, che verrà gestita in automatico da Asterisk.
Si definiscono poi tre tipologie di log:

- full: tutti i livelli di log sono abilitati;
- console: i log che verranno visualizzati mentre si è collegati alla console di Asterisk;
- centralino: un log di base per il server PBX, più facile da consultare del log full;

Per rendere effettivi i cambiamenti, dovrà essere riavviata l'intera piattaforma Asterisk oppure il solo modulo responsabile per il logging.

Ciò sarà possibile attraverso la console di Asterisk con il comando:
`sudo rasterisk 'module reload logger'`

È possibile impostare il debug per uno specifico modulo attraverso la console, ad esempio:
`sudo rasterisk 'sip set debug on'`
Permetterà ai log di livello 'debug' di visualizzare più informazioni sul protocollo SIP.

Sono disponibili ulteriori informazioni nella documentazione Asterisk, a questo indirizzo:
https://wiki.asterisk.org/wiki/display/AST/Collecting+Debug+Information

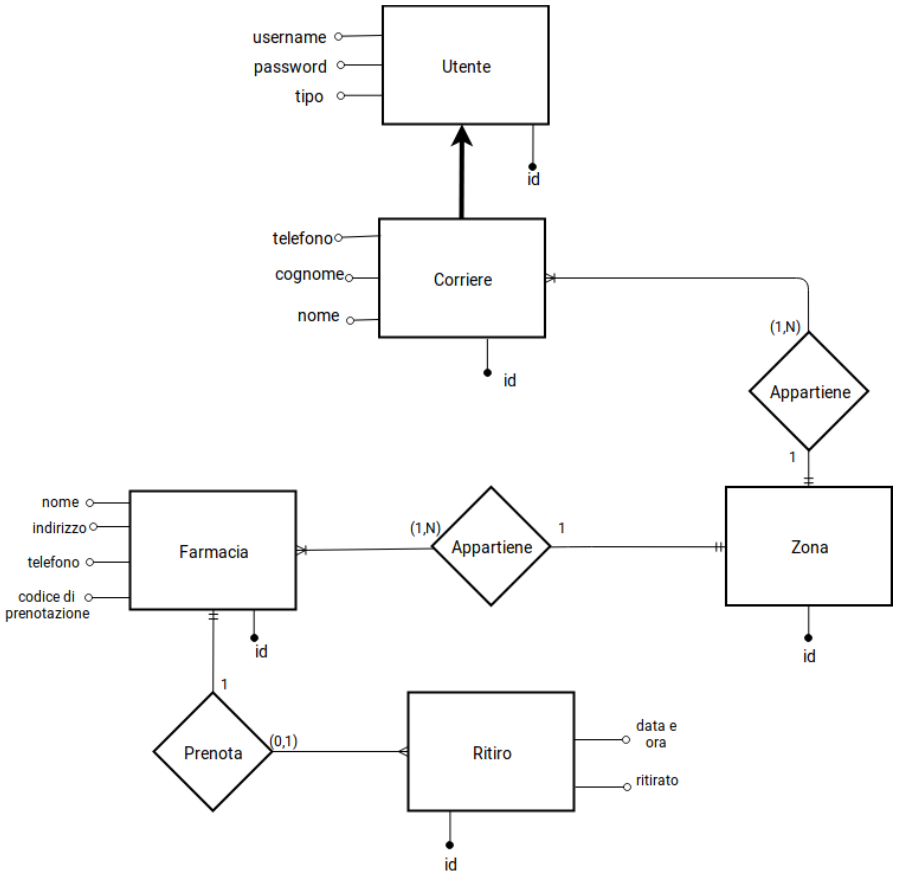
5. Interfaccia Web

In questo capitolo verrà trattata la parte relativa all'interfaccia web del server, a cominciare dal Database e dalle query SQL utilizzate. In seguito verranno analizzate la parte relativa alle classi Php di back-end, il front-end del sito, in particolar modo la parte relativa al codice Javascript per controllare gli input dell'interfaccia e la parte di CSS per organizzare la vista delle pagine web. In ultimo, verrà descritto come configurare Apache per gestire l'interfaccia web sviluppata e verranno indicati i permessi da impostare su file e cartelle relativi all'interfaccia web.

5.1 Database SQL

Di seguito verrà illustrata la struttura del database attraverso il diagramma Entità/Relazione, successivamente verranno elencate tutte le query utilizzate e verrà descritto come importare il database in MariaDB.

5.1.1 Diagramma E/R



Schema 3: SQL, Diagramma E/R

5.1.2 Query Utilizzate

Le query verranno divise per tipologia e se necessario verranno descritte brevemente.

```
CREATE TABLE `Pharmacy` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `reservationCode` int(11) NOT NULL,
  `name` varchar(25) NOT NULL,
  `address` varchar(50) NOT NULL,
  `phoneNumber` varchar(11) NOT NULL,
  `zoneId` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `reservationCode` (`reservationCode`),
  KEY `zoneId` (`zoneId`),
   CONSTRAINT `Pharmacy_ibfk_1` FOREIGN KEY (`zoneId`) REFERENCES `Zone`
(`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
```

Crea la tabella '`Pharmacy`'.

```
CREATE TABLE `Pickup` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
   `timeSchedule` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
  `pharmacyId` int(11) NOT NULL,
  `shipped` tinyint(1) NOT NULL DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `pharmacyId` (`pharmacyId`),
    CONSTRAINT `Pickup_ibfk_1` FOREIGN KEY (`pharmacyId`) REFERENCES
`Pharmacy` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
```

Crea la tabella '`Pickup`'. Cancellando un record nella tabella '`Pharmacy`' si cancelleranno anche i record di questa tabella contenenti la chiave esterna di '`Pharmacy`'.

```
CREATE TABLE `PonyExpress` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `userId` int(11) NOT NULL,
  `name` varchar(25) DEFAULT NULL,
  `surname` varchar(25) DEFAULT NULL,
  `phoneNumber` varchar(11) DEFAULT NULL,
  `zoneId` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `userId` (`userId`),
  KEY `zoneId` (`zoneId`),
   CONSTRAINT `PonyExpress_ibfk_1` FOREIGN KEY (`userId`) REFERENCES `User`
(`id`) ON DELETE CASCADE,
```

```
  CONSTRAINT `PonyExpress_ibfk_2` FOREIGN KEY (`zoneId`) REFERENCES `Zone`
(`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Crea la tabella '`PonyExpress`'. Cancellando un record nella tabella '`User`' si cancelleranno anche i record di questa tabella contenenti la chiave esterna di '`User`'.

```
CREATE TABLE `Session` (
  `id` varchar(32) NOT NULL,
  `access` int(11) DEFAULT NULL,
  `data` text,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Crea la tabella '`Session`'.

```
CREATE TABLE `User` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(20) NOT NULL,
  `password` varchar(32) NOT NULL,
  `type` varchar(7) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`),
  UNIQUE KEY `id` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=latin1;
```

Crea la tabella '`User`'.

```
CREATE TABLE `Zone` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;
```

Crea la tabella '`Zone`'.

```
DELETE FROM Pickup WHERE id = :id
```

Cancella un record dalla tabella '`Pickup`' se '`id`' è uguale all'id ricevuto (`:id`).

```
SELECT * FROM User WHERE type = 'manager'
```

Seleziona tutti gli utenti di tipo '`manager`'.

```
SELECT PonyExpress.id, PonyExpress.userId, User.username, PonyExpress.name,
PonyExpress.surname, PonyExpress.phoneNumber, PonyExpress.zoneId
FROM PonyExpress, User
WHERE User.id = PonyExpress.userId
```

Seleziona i dati di un '`PonyExpress`' dalle tabelle '`User`' e '`PonyExpress`'.

```
SELECT * FROM Pharmacy
```
Seleziona tutti i dati dalla tabella '`Pharmacy`'.

```
SELECT Pickup.id as id, Pharmacy.name as name, Pharmacy.address as address,
Pharmacy.phoneNumber as number, DATE_FORMAT(Pickup.timeSchedule, '%d-%m-%Y
alle %H:%i:%s') AS timeSchedule, Pharmacy.zoneId as zoneId
FROM Pharmacy, Pickup
WHERE Pickup.pharmacyId = Pharmacy.id
ORDER BY Pharmacy.name
```
Seleziona tutte le informazioni relative al ritiro, la data in formato '`GG-MM-AAAA alle hh:mm:ss`', disposti in ordine alfabetico per nome della farmacia.

```
SELECT Pickup.id as id, Pharmacy.name as name, Pharmacy.address as address,
Pharmacy.phoneNumber as number,DATE_FORMAT(Pickup.timeSchedule, '%d-%m-%Y
alle %H:%i:%s') AS timeSchedule
FROM Pharmacy, Pickup
WHERE Pickup.pharmacyId = Pharmacy.id AND Pharmacy.zoneId = :zoneId
ORDER BY Pharmacy.name
```
Come la query precedente, ma il risultato viene filtrato secondo lo '`:zoneId`' ricevuto.

```
SELECT id FROM Zone
```
Seleziona il campo '`id`' nella tabella '`Zone`'.

```
SELECT data FROM Session WHERE id = :id
```
Seleziona '`data`' dalla tabella '`Session`' dove il campo '`id`' è uguale all'id ricevuto.

```
REPLACE INTO Session (id, data, access) VALUES(:id, :data, :access)
```
Aggiorna la tabella '`Session`' in ogni campo per il quale la chiave primaria coincide con quella impostata (`id = :id`). Il vecchio record viene cancellato e viene inserito il nuovo.

```
DELETE FROM Session WHERE id = :id
```
Cancella un record da '`Session`' se l'id coincide con quello ricevuto.

```
DELETE FROM Session WHERE access < :old
```
Cancella un record da '`Session`' se l'ultimo accesso è meno recente del valore ricevuto.

```
INSERT INTO PonyExpress (userId, name, surname, phoneNumber, zoneId)
VALUES (:userId, :name, :surname, :phoneNumber, :zoneId)
```
Inserisce un record nella tabella '`PonyExpress`'

```
SELECT zoneId FROM PonyExpress WHERE userId = :userId
```
Seleziona il campo '`zoneID`' se il valore '`userId`' coincide con quello ricevuto.

```
UPDATE PonyExpress SET name = :name WHERE id = :id
```
Sostituisce il campo ‘name’ con il nome ricevuto se l’id è uguale all’id ricevuto.
(Questo tipo di query è usato per ogni singolo campo della tabella)

```
SELECT reservationCode FROM Pharmacy WHERE reservationCode = :code
```
Seleziona il campo ‘reservationCode’ se esso è uguale a quello fornito.

```
INSERT INTO Pharmacy (reservationCode, name, address, phoneNumber, zoneId)
VALUES (:reservationCode, :name, :address, :phoneNumber, :zoneId)
```
Inserisce un record nella tabella ‘Pharmacy’.

```
DELETE FROM Pharmacy WHERE id = :id
```
Cancella un record dalla tabella ‘Pharmacy’ per un determinato ‘id’.

```
SELECT id FROM Pharmacy WHERE reservationCode = :code
```
Seleziona il campo ‘id’ per un determinato ‘reservationCode’.

```
SELECT reservationCode
FROM Pharmacy
WHERE reservationCode = :reservationCode AND id != :id
```
Questa query viene usata per verificare, in fase di inserimento del codice, che esso non sia già utilizzato da un’altra farmacia.

```
UPDATE Pharmacy SET reservationCode = :code WHERE id = :id
```
Aggiorna il campo ‘reservationCode’ per un determinato ‘id’.
(Questo tipo di query è usato per ogni campo della tabella ‘Pharmacy’)

```
SELECT username FROM User WHERE username = :username
```
Seleziona il campo ‘username’ per un determinato ‘username’.

```
INSERT INTO User (username, password, type)
VALUES (:username, :password, :type)
```
Inserisce un record nella tabella ‘User’.

```
SELECT username FROM User WHERE username = :username AND password
= :password
```
Seleziona ‘username’ per una determinata coppia di ‘username’ e ‘password’.
Questa query è usata per verificare, in fase di inserimento di un nuovo utente, che lo ‘username’ selezionato non sia già utilizzato da un altro utente.

```
DELETE FROM User WHERE id = :id AND NOT (type = 'admin')
```
Cancella un record dalla tabella ‘`User`’ per un determinato ‘`id`’, purché l’utente relativo al record non sia di tipo ‘`admin`’.

```
SELECT type FROM User WHERE username = :username
```
Seleziona il campo ‘`type`’ per un determinato ‘`username`’.
(Questo tipo di query viene usato anche per il campo ‘`id`’)

```
SELECT username FROM User WHERE username = :username AND id != :id
```
Questa query è usata per verificare, in fase di modifica di un utente, che lo ‘`username`’ selezionato non sia già utilizzato da un altro utente.

```
UPDATE User SET username = :newUsername
WHERE id = :id AND NOT (type = 'admin')
```
Aggiorna il campo ‘`username`’ per un determinato ‘`id`’ purché l’utente relativo al record non sia di tipo ‘`admin`’.
(Questo tipo di query viene usato anche per il campo ‘`password`’)

```
SELECT id FROM Pharmacy WHERE reservationCode = :code
```
Seleziona il campo ‘`id`’ nella tabella ‘`Farmacia`’ per un determinato ‘`reservationCode`’.

```
INSERT INTO Pickup (timeSchedule, pharmacyId) VALUES (NOW(), :pharmacyId)
```
Inserisce un nuovo record nella tabella ‘`Pickup`’.
Snippet 21: SQL, Query utilizzate

5.1.3 Importazione in MariaDB

Per importare il database in MariaDB, si dovrà generare un file .sql contenente i comandi per creare un nuovo database, un nuovo utente identificato da una password (i dati che saranno presenti nel file di configurazione .ini) e sarà necessario assegnare a quell’utente i privilegi di accesso, modifica e cancellazione del database.
Per fare ciò, verrà usato il seguente codice SQL:

```
CREATE DATABASE IF NOT EXISTS `medcenterdb`;
CREATE USER 'sqluser'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON medcenterdb . * TO `sqluser`@`localhost`;
USE `medcenterdb`;
```
Snippet 22: SQL, inizializzazione database.

Successivamente verranno aggiunte le `CREATE TABLE` definite in precedenza, e prima di ogni tabella verrà utilizzata la direttiva SQL: `DROP TABLE IF EXISTS `nome_tabella`;`
per evitare errori nel caso in cui si voglia sostituire un database esistente.

Per importare il database in MariaDB, si dovrà copiare il file `database.sql` nel server e sarà necessario il comando:

```
sudo mysql -u root -p < database.sql
```

Il comando non farà altro che autenticarsi a MariaDB ed eseguire tutti i comandi SQL presenti nel file indicato.

5.2 Back-end Php

Di seguito verranno elencate le classi utilizzate per il back-end dell'interfaccia; per ogni classe verranno indicati il costruttore, i prototipi dei metodi e una breve descrizione del funzionamento dove necessario.

5.2.1 Database

La classe che implementa la comunicazione con il database utilizza il driver PDO (PHP Data Objects), il quale utilizza un file di configurazione (dbconfig.ini) per connettersi al database, con sintassi:

```
[database]
host = 'localhost'
user = 'sqluser'
passwd = 'password'
dbname = 'medcenterdb'
```

Snippet 23, dbconfig.ini

```
	/**
	* Database __construct() function
	*
	* Uses PDO driver to handle db interaction
	*
	*/

	public function __construct(){

		// load config file
		$conf = parse_ini_file("db_config.ini");

		// create string for the PDO obj instantiation
		$pdo_string = "mysql:host=".$conf['host'].";dbname=".$conf['dbname'];

		try {
			// create PDO instance
			$this -> db = new PDO($pdo_string, $conf['user'],
					$conf['passwd']);

		/**
		 * you must always catch the PDO Exception or
		 * it will print the backtrace with db sensitive data in it
```

```
        * (see PHP Manual for reference)
        */

       } catch (PDOException $e){

              die ($e -> getMessage() );
       }
}
```

Snippet 24: Php, Costruttore della classe Database

```
/**
* Database query function
*
* prepares the SQL query to be executed
*
* @param $query string
* @return true | false bool
*
*/
public function dbQuery($query){
}

/**
 * Database bind function
 *
 * binds the values using bindParam() PDO Method to execute the query
 *
 * @param $param, $value
 * @return true | false bool
 *
 */
 public function dbBind($param, $value){
}

/**
 * Database execute function
 *
 * executes the statement
 *
 * @return true | false bool
 *
 */
public function dbExecute(){
}
/**
 * Database result set function
 *
 * returns the values fetched from a query that returns multiple rows
 *
 * @return PDO::fetchAll array of type PDO::FETCH_ASSOC
 *
 */
 public function dbResultSet(){
 }
```

```
/**
 * Database single result set function
 *
 * returns the first value fetched from a query, this should be used when 1
 *  result is expected
 *
 * @return PDO::fetchAll array of type PDO::FETCH_ASSOC
 *
 */
 public function dbSingleResultSet(){
 }
```

Snippet 25: Php, prototipi dei metodi della classe Database

Il costruttore utilizza i dati del file di configurazione per creare un oggetto `PDO`, il quale costituirà la connessione con il database. Il metodo `dbQuery` prepara la query per essere eseguita, utilizzando il metodo `PDO::prepare($query)`. I valori delle variabili non vengono specificati in `dbQuery`, ma vengono sostituiti da delle etichette, come ad esempio:
`dbQuery("SELECT FROM Pharmacy WHERE id = :id");`

In questo modo si evita di esporre il database ad attacchi di SQL Injection[8]; il metodo `dbBind` servirà proprio a sostituire ad ogni etichetta il valore della variabile da passare alla query.
Attraverso uno switch-case, il metodo `dbBind` riconosce il tipo di dato che viene passato e lo converte in un tipo di dato `PDO`, ad esempio se il dato è di tipo '`boolean`', gli verrà assegnato il valore '`PDO::PARAM_BOOL`', secondo la sintassi del driver `PDO`. Il metodo `PDO::bindParam($param, $value, $type)` viene utilizzato per assegnare ad ogni etichetta il suo valore reale, le query verranno poi eseguite con il metodo `dbExecute`.
Per recuperare dei dati dal database, attraverso una query di tipo `SELECT`, si utilizza la funzione `dbResultSet`, che restituirà un array contente il risultato della query.
Il metodo `dbSingleResultSet` invece, restituirà il primo valore trovato dalla `SELECT`, il che risulterà utile nel caso in cui il risultato atteso della query sia costituito da un solo record.

Per eseguire una query il procedimento è il seguente. Si prepara la query:
`dbQuery("SELECT name, address, phoneNumber FROM Pharmacy WHERE id = :id");`

Successivamente, si dovrà associare all'etichetta la variabile corrispondente:
`dbBind(':id', $id);`

In fine, si potrà eseguire la query:
`dbExecute();`

8 Tipo di attacco informatico che consiste nell'inserire delle stringhe di codice SQL malevolo all'interno dei campi di input al fine di essere essere eseguite.

Per ottenere il risultato della query in un array:

```
$array = dbResultSet();
```

L'array risultante sarà multidimensionale e di lunghezza pari al numero di righe restituite dalla query, ciascuna delle quali avrà il numero dei campi specificati dalla `SELECT`.

5.2.2 Session

La classe Session viene creata con lo scopo di salvare le sessioni Php nel database. Nel costruttore verranno sovrascritti i metodi utilizzati da Php per aprire, chiudere, leggere, scrivere e distruggere le sessioni, indirizzando le operazioni al database. Verrà sovrascritto anche il metodo di 'garbage collection', in modo che utilizzi il database. Questo metodo viene utilizzato da Php per cancellare le sessioni che sono inutilizzate da una determinata quantità di tempo. Di seguito il costruttore della classe e i prototipi dei metodi:

```
public function __construct(){

		// db obj to use to handle sessions

		$this -> db = new Database();

		// override the default methods for handling sessions
		session_set_save_handler(
			array($this, "_open"),
			array($this, "_close"),
			array($this, "_read"),
			array($this, "_write"),
			array($this, "_destroy"),
			array($this, "_gc")
		);

		// needed to call write when session is closed (standard practice)
		register_shutdown_function('session_write_close');
}
```

Snippet 26: Php, costrutture della classe Session

```
	/* methods starting with '_' (underscore) are overrides of default php
	session handler class methods */

	/**
	* open
	*
	*/
	public function _open(){
	}

	/**
	 * close
	 *
```

```
 */
public function _close(){
}

/**
 * read
 *
 * @param $id - session id
 *
 * @return session data if any, or an empty string
 *
 */
public function _read($id){
}

/**
 * write
 *
 * updates an existing session with the new access time or creates a new
 *  entry
 *
 * @param $id, $data
 *
 * @return true | false bool
 *
 */
public function _write($id, $data){
}

/**
 * destroy
 *
 * deletes a session with $id from the database, called with
 * session_destroy();
 *
 * @param $id
 *
 * @return true | false bool
 *
 */
public function _destroy($id){
}

/**
 * gc
 *
 * garbage collection function - called based on the value of
 * session.gc_probability         * and  session.gc_divisor in php.ini
 *
 * return true | false bool
 *
 */
 public function _gc($max){
 }
```

```
/**
  * start method
  *
  * starts the session and sets $_SESSION variables needed by the application
  *
  * @param $username, $type string
  *
  * @return true | false bool
  *
  */
 public function start($username, $type){
 }

 /**
  * clear method
  *
  * calls session_destroy() and clears $_SESSION array for $username
  *
  * @return true | false bool
  *
  */
 public function clear(){
 }

 /**
  * get
  *
  * @param $key - session index variable to get
  * @return $_SESSION[$key] - the session variable with the $key index
  *
  */
public function get($key){
}
/**
 *
 * set - sets a session attribute.
 *
 * @param $name - session attribute name
 * @param $value - value of the attribute
 *
 * @return true | false bool
 *
 */
 public function set($name, $value){
}
```

Snippet 27: Php, prototipi metodi della classe Session

5.2.3 User

La classe User viene usata per gestire la registrazione, il login e il logout di un utente. Dato che in Php non vi è un metodo per dichiarare più modi di definire un oggetto (come in Java), nel costruttore si è implementato uno switch-case che a seconda del numero degli argomenti coi quali viene dichiarato l'oggetto, assegna agli attributi il valore corrispondente, in maniera posizionale.

Ad esempio:
`$user = new User('foo');`
l'oggetto è stato dichiarato con un solo parametro, si assume che verrà utilizzato per il logout e '`foo`' viene interpretato come l'attributo '`username`'.

Se invece si dichiara l'oggetto in questo modo:
`$user = new User(1, 'foo','bar');`
l'oggetto viene dichiarato con tre parametri e si assume che verrà utilizzato per la registrazione, quindi '`1`', '`foo`' e '`bar`' saranno i rispettivi attributi '`id`', '`username`' e '`password`'.

Questo approccio si è rivelato funzionale ma non privo di inconvenienti: non è possibile infatti definire due costruttori diversi aventi lo stesso numero di parametri. Sia per la registrazione che per la creazione di un array di tipo '`User`' sono necessari tre parametri ma, nel caso della registrazione servono 'id', '`username`' e '`password`' mentre pere la creazione dell'array '`id`', '`username`' e '`type`'.
Si è ovviato al problema utilizzando quattro parametri per la creazione dell'array, utilizzando un parametro '`segnaposto`' che non verrà utilizzato.

```
public function __construct(){

            // put the arguments of the constructor in the $args array
            $args = func_get_args();
            $n = func_num_args();

            // switch cases for each possible Object creation
            switch($n){

                  case 0:
                        // Database object to query the db
                        $this -> db = new Database();
                        // Session object to handle user sessions
                        $this -> session = new Session();
                        break;
```

```
                case 1:

                        // Deletion, logout: $username
                        $this -> username = $args[0];
                        // Database object to query the db
                        $this -> db = new Database();
                        // Session object to handle user sessions
                        $this -> session = new Session();
                        break;

                case 2:

                        /**
                         * Login: $username, $passwd
                         *
                         * to get login type use getUserType() instead
                         *
                         */

                        $this -> username = $args[0];
                        $this -> passwd = $args[1];
                        // Database object to query the db
                        $this -> db = new Database();
                        // Session object to handle user sessions
                        $this -> session = new Session();
                        break;

                case 3:

                        // Registration: $username, $passwd, $type
                        $this -> username = $args[0];
                        $this -> passwd = $args[1];
                        $this -> type = $args[2];
                        // Database object to query the db
                        $this -> db = new Database();
                        break;

                case 4:

                        /* used to retrieve all the records from the db and put it
                        in a User array */
                        $this -> id = $args[0];
                        $this -> username = $args[1];
                        // args[2] contains empty 'password' placeholder
                        $this -> type = $args[3];
                        // Database object to query the db
                        $this -> db = new Database();
                        break;

                default:
                        break;
        }
}
```

Snippet 28: Php, costruttore classe User

```
/**
 *
 * Registration method - registers a user into the db
 *
 * @param no param needed, class attributes are used
 *
 * @return true | false bool
 *
 */
public function register(){
}

/**
 *
 * Login method - logs in a user
 *
 * @param no param needed, class attributes are used
 * @return true | throws exceptions: wrong credentials, user already logged
 * in
 *
 */
public function login(){
}

/**
 *
 * Logout method - logs out the user and destroys its session
 *
 *
 * @param no param needed, class attributes are used
 * @return true | false bool
 *
 */
public function logout(){
}

/**
 * Delete User method - deletes a given $id and $type
 *
 * @param $id int - Username id
 * @param $type string - Username type
 * @return true | false bool
 *
 */
public function deleteUser($id){
}

/**
 * isLogged method - checks if a user is logged in
 *
 * private method - not used outside of this class
 *
 * @param $user string - the username to check
```

```
 * @return true | false bool - true if logged, false otherwise
 *
 */
protected function isLogged($user){
}

/**
*
* Getter methods
*
*/
public function getType(){
}

/**
 * get type directly from the db
 *
 * needed to get the type after user login
 *
 * @param no param needed, class attributes are used
 * @return type attribute from db, false bool
 *
 */
public function getTypeFromDb(){
}

/**
 * get id from the db
 *
 * needed to get the primary key (id) of the user
 *
 * @param no param needed, class attributes are used
 * @return id attribute from db, false bool
 *
 */
public function getIdFromDb(){
}

public function getId(){
}

public function getUsername(){
}

public function getPasswd(){
}

/**
*
* Setter methods
*
*/
public function setType($type){
}
```

```
public function setUsername($username){
}

public function setPasswd($passwd){
}

/**
 *
 * Update methods
 *
 * update a specific attribute in the database
 *
 */
public function updateUsername($id, $newUsername){
}

public function updatePassword($id, $newPassword){
}
```
Snippet 29: Php, prototipi metodi della classe User

5.2.4 Pharmacy

Questa classe permette di creare, eliminare, ottenere e aggiornare i dati dalla tabella 'Pharmacy' del Database. L'approccio usato per il costruttore e i metodi segue la stessa logica dalla classe 'User'.

```
/**
 * Pharmacy constructor
 *
 * can be created using:
 * $reservationCode string (e.g. to handle a Pickup)
 *
 * or using:
 *
 * $reservationCode string
 * $name            string
 * $address           string
 * $phoneNumber     string
 * $zoneId          string
 *
 * or using:
 *
 * $id                    int (primary key of the table)
 * $reservationCode string
 * $name            string
 * $address             string
 * $phoneNumber     string
 * $zoneId          string
 *
 */
```

```
public function __construct(){

	// put the arguments of the constructor in the $args array
	$args = func_get_args();
	$n = func_num_args();

	// switch cases for each possible Object creation
	switch($n){

		case 0:

			$this -> db = new Database();
			break;

		case 1:

			$this -> reservationCode = $args[0];
			$this -> db = new Database();
			break;

		case 5:

			$this -> reservationCode = $args[0];
			$this -> name = $args[1];
			$this -> address = $args[2];
			$this -> phoneNumber = $args[3];
			$this -> zoneId = $args[4];
			$this -> db = new Database();
			break;

		case 6:

			$this -> id = $args[0];
			$this -> reservationCode = $args[1];
			$this -> name = $args[2];
			$this -> address = $args[3];
			$this -> phoneNumber = $args[4];
			$this -> zoneId = $args[5];
			$this -> db = new Database();
			break;


		default:
			break;
	}
}
```

Snippet 30: Php, costruttore classe Pharmacy

```
/**
 *
 * Registration method - registers a Pharmacy Into the Database
 *
 * @param no param needed, class attributes are used
 * @return true | false bool
 *
 */
 public function register(){
 }

 /**
  * Delete Pharmacy method - deletes a pharmacy given the reservationCode
  *
  * @param $id int - Pharmacy id
  * @return true | false bool
  *
  */
 public function deletePharmacy($id){
 }

/**
 *
 * Getter methods
 *
 */
 // gets the Pharmacy Id from the database given the reservation code
 public function getIdFromDb(){
 }

 public function getId(){
 }

 public function getReservationCode()
 }

 public function getName(){
 }

 public function getAddress(){
 }

 public function getPhoneNumber(){
 }

 public function getZoneId(){
 }
```

```
/**
 *
 * Setter methods
 *
 */

 public function     setReservationCode($code){
  }

  public function setName($name){
  }

  public function setAddress($address){
  }

  public function setPhoneNumber($phoneNumber){
  }

  public function setZoneId($zoneId){
  }

  /**
  *
  * Update methods
  *
  * update a specific attribute in the database
  *
  */

  public function updateReservationCode($id, $newCode){
  }

  public function updateName($id, $newName){
  }

   public function updateAddress($id, $newAddress){
  }

   public function updatePhoneNumber($id, $newPhoneNumber){
  }

   public function updateZoneId($id, $newZoneId){
  }

 // updates entire row of pharmacy table
 public function updateRow($id, $newCode, $newName, $newAddress,
                           $newPhoneNumber, $newZoneId){
 }
```

Snippet 31: Php, prototipi metodi classe Pharmacy

5.4.5 PonyExpress

Questa classe estende la classe '`User`', ereditandone i metodi. I metodi della classe '`User`' possono essere usati in questa classe senza essere definiti. Il costruttore della classe sovrascrive quello della classe padre '`User`'.

Per registrare un oggetto di tipo '`PonyExpress`' nella tabella '`User`':
`$pony → register();`

Verrà invece registrato nella tabella '`PonyExpress`' con:
`$pony → registerPonyExpress();`

```
/**
 * PonyExpress class constructor, overrides User constructor
 *
 * PonyExpress object doesn't handle login/logout
 * can be created using:
 * $username or
 * $username, $password, $name, $surname, $phoneNumber, $zoneId or
 * $id, $username, $password (placeholder), $name, $surname, $phoneNumber,
 * $zoneId
 *
 */
public function __construct(){
      // type here is fixed to "pony" upon creation
      $this -> type = 'pony';

      // put the arguments of the constructor in the $args array
      $args = func_get_args();
      $n = func_num_args();

      // switch cases for each possible Object creation
      switch($n){

            case 0:

                  // Database object to query the db
                  $this -> db = new Database();
                  // Session object to handle user sessions
                  $this -> session = new Session();
                  break;

            case 1:

                  // delete or update: only $username needed
                  $this -> username = $args[0];
                  // Database object to query the db
                  $this -> db = new Database();
                  break;

            case 6:
```

```
                        // insert: every field needed
                        $this -> username = $args[0];
                        $this -> passwd = $args[1];
                        $this -> name = $args[2];
                        $this -> surname = $args[3];
                        $this -> phoneNumber = $args[4];
                        $this -> zoneId = $args[5];
                        // Database object to query the db
                        $this -> db = new Database();
                        break;

                case 8:

                        // used to retrieve all records from PonyExpress table
                        $this -> ponyExpressId = $args[0];
                        $this -> id = $args[1];
                        $this -> username = $args[2];
                        // args[3] contains password empty placeholder
                        $this -> name = $args[4];
                        $this -> surname = $args[5];
                        $this -> phoneNumber = $args[6];
                        $this -> zoneId = $args[7];
                        // Database object to query the db
                        $this -> db = new Database();
                        break;

                default:
                        break;
        }
}
```

Snippet 32: Php, costruttore classe PonyExpress

```
/**
 *
 * Registration method - registers a PonyExpress into the db
 *
 * Inserts the PonyExpress in the PonyExpress table
 *
 * @param no param needed, class attributes are used
 *
 * @return true | false bool
 *
 */
public function registerPonyExpress(){
}

/**
*
* Getter methods
*
*/

public function getPonyExpressId(){
```

```
}

public function getName(){
}

public function getSurname(){
}

public function getPhoneNumber(){
}

public function getZoneId(){
}

// gets zoneId from the database
public function getZoneIdFromDb(){
}

/**
*
* Setter methods
*
*/

public function setName($name){
}

public function setSurname($surname){
}

public function setPhoneNumber($phone){
}

public function setZoneId($zoneId){
}

/**
 *
 * Update methods
 *
 * update a specific attribute in the database
 *
 */
 public function updateName($id, $newName){
 }

 public function updateSurname($id, $newSurname){
 }

 public function updatePhoneNumber($id, $newPhoneNumber){
 }

 public function updateZoneId($id, $newZoneId){
 }
```

```
 /**
  *
  * this method updates every field of the ponyExpress table
  *
  */

 public function updateRow($id, $newName, $newSurname, $newPhoneNumber,
                           $newZoneId){
 }
```
Snippet 33: Php, prototipi metodi classe PonyExpress

5.2.6 Pickup

La classe '`Pickup`' gestisce la cancellazione degli elementi dalla tabella e permette di ottenere tutti i campi necessari a produrre una tabella con le informazioni relative ai ritiri, che verranno usati nella classe seguente, '`UserInterface`', per produrre l'output da presentare all'utente.

```
/**
 * Pickup constructor
 *
 * @param $pharmacyCode string
 *
 * or
 *
 * $id int, $pharmacyName string, $pharmacyAddress string,
 * $pharmacyPhoneNumber           * string,
 * $pickupNo int, $timeSchedule string, $pharmacyZoneId int
 *
 *
 */
public function __construct(){

      // put the arguments of the constructor in the $args array
      $args = func_get_args();
      $n = func_num_args();

      // switch cases for each possible Object creation
      switch($n){

            case 0:
                  // create db obj
                  $this -> db = new Database();
                  break;
            case 6:
                  /* id and every other fields are passed. Used to get all
                  data from Pickup table */
                  $this -> id = $args[0];
                  $this -> pharmacyName = $args[1];
                  $this -> pharmacyAddress = $args[2];
                  $this -> pharmacyPhoneNumber = $args[3];
                  $this -> timeSchedule = $args[4];
```

```
                    $this -> pharmacyZoneId = $args[5];
                    // create db obj
                    $this -> db = new Database();
                    break;

            default:
                    break;
      }
}
```
Snippet 34: Php, costrutture classe Pickup

```
/**
 * Delete Pickup method - deletes a Pickup given the pickup id
 *
 * @param $id int - pickup id
 * @return true | false bool
 *
 */
public function deletePickup($id){
}

/**
*
* Getter methods
*
*/
public function getId(){
}

public function getTimeSchedule(){
}

public function getPharmacyName(){
}

public function getPharmacyAddress(){
}

public function getPharmacyPhoneNumber(){
}

public function getPharmacyZoneId(){
}
```
Snippet 35: Php, prototipi dei metodi della classe Pickup

5.2.7 UserInterface

Questa classe verrà usata nelle pagine di front-end per presentare l'output dei vari elementi presenti nel database. Sono presenti due tipologie di metodi: un metodo che per ogni tipo di tabella effettua una query nel database, ottiene i dati e li copia in un array; la seconda tipologia di metodi utilizza l'array ottenuto per stampare una tabella in html che conterrà l'output desiderato. Ad esempio per ottenere la tabella che visualizza la lista di tutti i Pony Express sarà sufficiente:

```
$interface = new UserInterface();
$interface → ponyExpressTable();

public function __construct(){

            $this -> db = new Database();
            $this -> session = new Session();
}
```

Snippet 36: Php, costruttore classe UserInterface

```
        /**
         *
         * getAllManagers method
         *
         * this method returns a User type array containing all the managers in the
         * db
         *
         */
        public function getAllManagers(){
        }

        /**
         *
         * getAllPonyExpresses method
         *
         * this method returns a PonyExpress type array containing all the pony
         * expresses in the db
         *
         */
        public function getAllPonyExpresses(){
        }
        /**
         *
         * getAllPharmacies method
         *
         * this method returns a Pharmacy type array containing all the pharmacies in
         * the db
         *
         */
        public function getAllPharmacies(){
        }
```

```
/**
 *
 * getAllPickups method
 *
 * this method returns a Pickup type array containing all the pickups in the
 * db
 *
 */
public function getAllPickups(){
}

public function getAllZones(){
}

/**
 *
 * getAllPickupsByZone method - this filters the pickups based on the logged
 * user's zoneId
 *
 */
public function getAllPickupsByZone(){
}

/* table methods */

/**
 * managerTable method - returns a string containing manager data to use in
 * the manager table
 *
 * @return $table string
 *
 */
public function managerTable(){
}

/**
 *
 * ponyExpress table method - returns a string containing pony express data
 * to use in the pony express table
 *
 * @return $table string
 *
 */
public function ponyExpressTable(){
}

/**
 *
 * pharmacy table method - returns a string containing pharmacy data to use
 * in the pharmacy table
 *
 * @return $table string
 *
 */
```

```
public function pharmacyTable(){
}

/**
 *
 * pickup table method - returns a string containing pickup data to use in
 * the pickup table
 *
 * @return $table string
 *
 */
public function pickupTable(){
}

/**
 *
 * pickup pony express table - pickups filtered by zoneId
 *
 */
public function pickupPonyExpressTable(){
}

/**
 * zone option method - returns a string containing html options with the
 * data from Zone table
 *
 */
public function zoneOption(){
}
```

Snippet 37: Php, prototipi metodi della classe UserInterface

5.3 Front-end

Nella parte di front-end dell'interfaccia verrà descritta l'organizzazione dei file per i vari tipi di utente e si esporranno le funzionalità delle pagine dedicate all'utente '`admin`', che includono tutte le funzionalità dell'interfaccia. Le utenze '`manager`' e '`pony`', infatti, includono le stesse funzionalità ma, ovviamente, con meno privilegi. I file e le directory sono organizzati in questo modo:

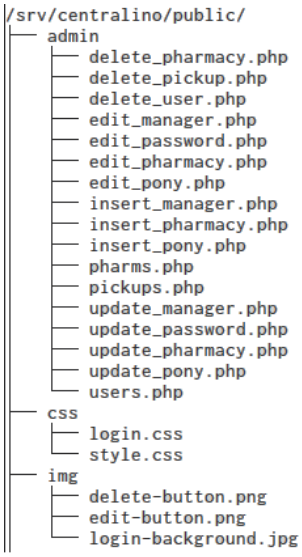


Figura 14: Front-end, vista ad albero, 1

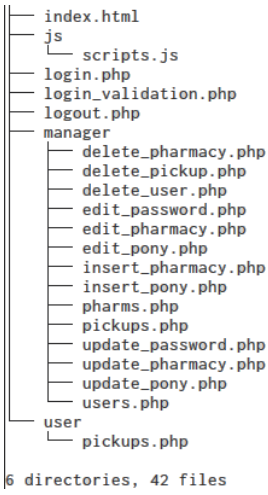


Figura 15: Front-end, vista ad albero, 2

La pagina `index.html` reindirizza semplicemente l'utente alla pagina di login. In ogni pagina, per ogni tipo di utente, si controlla che l'utente con sessione attiva abbia i privilegi necessari per visualizzare la pagina, e in caso non li avesse si provvede a reindirizzarlo alla pagina appropriata. In questo modo un utente con sessione di tipo '`pony express`' non potrà accedere alla pagina dell'amministratore inserendo '`admin/pickups.php`' nella barra degli indirizzi.

5.3.1 Login

Prima di presentare il codice HTML, viene controllato se l'utente ha già una sessione attiva nel database (attraverso la classe '`Session`'). Nel caso vi sia una sessione attiva, l'utente viene reindirizzato alla propria pagina iniziale. Ad esempio, il manager verrà reindirizzato a `manager/pickups.php`, mentre l'amministratore ad `admin/pickups.php`.
Nella pagina `login.php` è presente un form HTML in cui l'utente inserisce le credenziali d'accesso. Una volta premuto il tasto 'login' il form trasferirà i dati alla pagina

`login_validation.php` che controllerà le credenziali nel database. Nel caso di credenziali corrette, verrà creata una sessione per l'utente e questi sarà rediretto alla sua pagina iniziale, altrimenti verrà reindirizzato alla pagina di login con un messaggio di errore. Una volta creata la sessione, in tutte le pagine visitate, l'utente può effettuare il logout tramite un pulsante HTML, il quale porterà alla pagina `logout.php`, che si occuperà di cancellare la sessione dal database e di reindirizzare l'utente alla pagina di accesso. Tutti i form dell'interfaccia web vengono gestiti secondo questa logica: l'utente inserisce i dati nei vari campi del form e conferma; i dati verranno elaborati da una pagina intermedia che effettuerà l'operazione richiesta oppure verrà rediretto alla pagina del form con un messaggio d'errore. I campi dei form vengono controllati tramite javascript, definito nella cartella denominata '`js`' e le cui funzioni verranno esposte nei paragrafi seguenti.

5.3.2 Password

Le password non vengono memorizzate in chiaro nel server: in fase di registrazione, modifica della password e login, prima di comunicare con il database, viene calcolato l'hash della password mediante l'algoritmo '`md5`'. È importante notare che se il protocollo http del server non viene cifrato tramite protocolli di trasporto sicuri, quali TLS/SSL (connessione https), la password verrà inviata al server in chiaro, dato che l'hash viene calcolato nel server, e chiunque intercetti il pacchetto contenente la password potrà entrarne in possesso. È quindi buona norma implementare una connessione https per il proprio server.

5.3.3 Admin

L'amministratore, come anche il manager, ha a disposizione tre menù nell'interfaccia web: Ritiri, Utenti e Farmacie. Ogni tabella è presentata tramite la classe `UserInterface`, trattata precedentemente nella sezione relativa al back-end.

Nella pagina '`Ritiri`' si può visualizzare la lista delle prenotazioni effettuate dalle farmacie di ogni zona della città, con la possibilità di rimuovere una prenotazione dalla lista.

Nella pagina '`Utenti`' viene visualizzata la lista degli utenti, divisi per tipologia: `Manager` e `Pony Express`. L'amministratore, tramite questa pagina, ha la possibilità di modificare ogni campo dell'utente, oltre che la password d'accesso. Per la modifica, verrà aperta una nuova scheda nel browser (tramite javascript) contenente un form già compilato con i campi relativi all'utente da modificare. All'invio del form, i dati verranno elaborati e si verrà rediretti al menù '`Utenti`' in caso di successo, oppure verrà visualizzato un messaggio di errore per ogni campo da correggere. L'amministratore ha la possibilità di inserire nuovi utenti, sia di tipo '`Manager`' che di tipo '`Pony Express`' seguendo la stessa logica che si applica agli altri form HTML già descritti. Si presume che il manager, ovvero l'utenza che gestisce i corrieri,

sia unica, come l'amministratore. Per questo, i dati anagrafici vengono richiesti solamente per gli utenti `Pony Express`. Per completezza, si è ritenuto corretto permettere all'amministratore di poter aggiungere più utenti di tipo `Manager`, nel caso fosse necessario. L'amministratore ha inoltre facoltà di eliminare un utente dal database.

In 'Farmacie' la vista presentata è la stessa sia per l'utente '`Admin`' che per l'utente '`Manager`': in essa viene presentata una tabella contenente i dati di ogni farmacia. Come per la pagina '`Utenti`', è possibile modificare ogni campo della tabella e cancellarne gli elementi dal database.

5.3.4 Manager

L'utenza `Manager` ha gli stessi privilegi dell'amministratore, descritti in precedenza, ma non può creare, modificare o cancellare altri utenti di tipo `Manager`.

5.3.5 Pony Express

Per l'utenza `Pony Express` è visibile solamente il menù 'Ritiri', in cui viene visualizzata la tabella delle prenotazioni, filtrata per zona. In questo modo, il corriere potrà visualizzare soltanto i ritiri da effettuare nella propria zona di appartenenza. L'utente `Pony Express` non può cancellare un ritiro dalla lista.

5.3.6 Javascript

Javascript è il linguaggio più utilizzato, soprattutto per il front-end, nei siti web moderni. Data la semplicità dell'interfaccia da proporre, verrà utilizzato soltanto per controllare la correttezza dell'input nei form. In caso di errore, verrà impedito all'utente di procedere alla pagina di elaborazione dati e verrà stampato nei campi errati del form un 'placeholder' HTML contenente il messaggio di errore. Le varie funzione Javascript vengono definite nel file: `js/scripts.js`

```
function checkNumbers(x){
}
function validatePassword(p1, p2, validation){
}
function editPasswordValidation(){
}
function loginValidation(user, pass){
}
function editManagerForm(){
}
function ponyForm(n){
}
```

```
function pharmForm(n){
}
function deleteAlert(link, m){
}
```
Snippet 38: Javascript, prototipi funzioni

I controlli effettuati sono i seguenti:

- Per ogni form, verrà controllato che tutti i campi siano compilati (sia per il form di login che per quelli di registrazione e modifica).
- Nell'inserimento di una nuova password, viene chiesto di inserire la password due volte e si controlla che le due password coincidano.
- Per i campi numerici, ad esempio il numero di telefono si controlla che non vengano inseriti caratteri diversi da cifre. L'input viene confrontato con l'espressione regolare[9] "`^[0-9]+$`".
- Per la cancellazione di record dal database (colonna 'Elimina' delle tabelle) viene presentato un pop-up per confermare o annullare la cancellazione.

5.3.7 CSS

Non essendo l'interfaccia grafica il focus principale del progetto per quanto riguarda la parte di CSS è stato utilizzato un template gratuito dal sito: https://templated.co che è stato successivamente modificato per visualizzare le pagine nella maniera desiderata. I file CSS che definiscono la grafica del sito sono `css/login.css` e `css/style.css`. Per permettere una visualizzazione appropriata su dispositivi con risoluzione diversa da un monitor classico (smartphone e tablet, i dispositivi che verranno utilizzati dai corrieri) si è utilizzata la seguente regola CSS:

```
@media screen and (max-width: 1080px) {
    .wrapper {
        max-width: 100%;
        float: none;
    }
    .input, .table, .right {
        width: 100%;
    }
}
```
Snippet 39: CSS, regola per schermi mobile.

La regola permette agli schermi con larghezza massima pari a 1080 pixel di visualizzare l'interfaccia senza che le varie parti che la compongono (input, tabella, logout) si

9 Sequenza di simboli che identifica un insieme di stringhe attraverso una funzione che prende in input la stringa da esaminare, la confronta con il modello definito dall'espressione regolare (la sequenza di simboli) e restituisce un valore booleano (vero/falso) in caso di appartenenza, o meno, della stringa al modello.

sovrappongano tra loro. Il resto del CSS definisce i colori e lo stile da utilizzare per bottoni, tabelle e menù e come ogni elemento HTML (`div`) viene posizionato rispetto al resto della pagina.

5.3.8 Screenshot Front-end

Di seguito si riportano gli screenshot principali delle pagine front-end dell'interfaccia web.

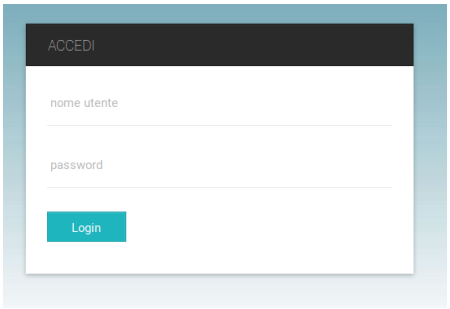


Figura 16: Interfaccia web, schermata di Login

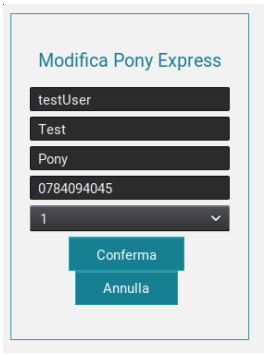


Figura 17: Interfaccia web, modifica utente

Figura 18: Interfaccia web, menù Farmacie

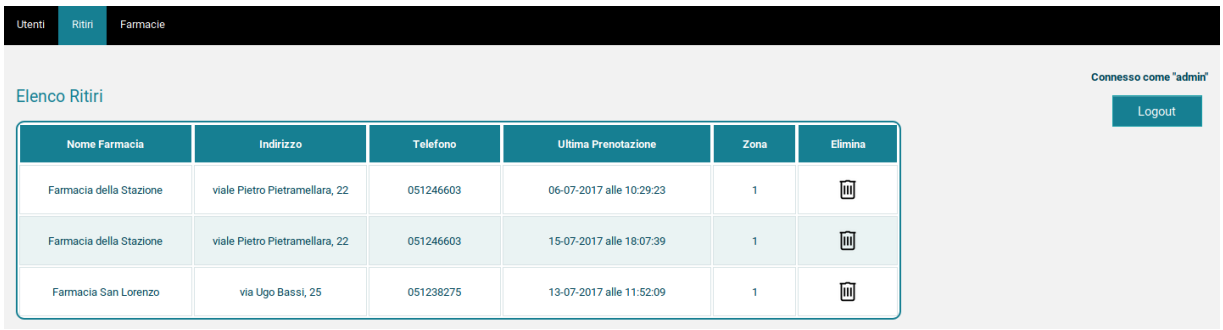


Figura 19: Interfaccia web, menù Ritiri

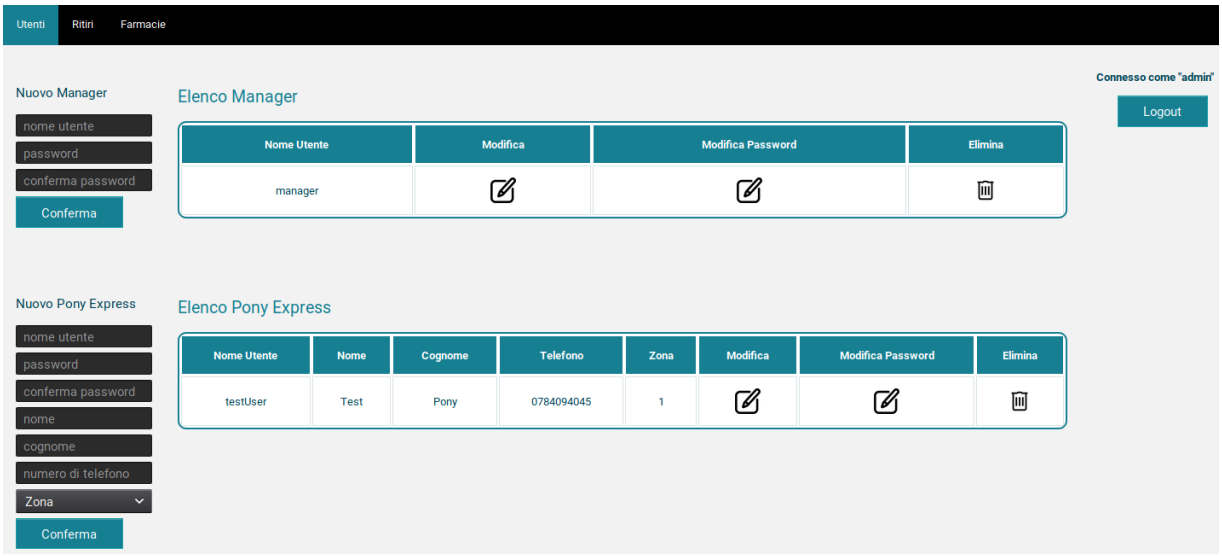


Figura 20: Interfaccia web, menù utenti

5.4 Apache

Per abilitare l'interfaccia web, bisognerà definire un file di configurazione in Apache in cui si indicheranno l'indirizzo del sito e la `DocumentRoot`, ossia la directory a partire dalla quale saranno disponibili le pagine di front-end da servire all'utente le impostazioni per i file di log. Il file di configurazione per l'interfaccia web si trova al percorso:
`/etc/apache2/sites-available/centralino.conf`

Di seguito vedremo, per ogni sezione del file le opzioni abilitate.

5.4.1 General

```
ServerName dominio
ServerAdmin webmaster@localhost
DocumentRoot /srv/centralino/public
```

Snippet 40: Apache, centralino.conf General

In questa sezione si definisce il nome del server, che può essere il dominio a cui l'indirizzo è associato, oppure l'indirizzo IP esterno, nel caso in cui non si disponga di un dominio. Inoltre, è possibile impostare l'email dell'amministratore di sistema e la `DocumentRoot` del server. Accedendo all'indirizzo indicato in `ServerName`, Apache andrà alla `DocumentRoot` e cercherà al suo interno un file chiamato `index`, che sarà la pagina iniziale.

5.4.2 Directories

```
<Directory /srv/centralino/public>
      Options -Indexes +FollowSymlinks
      AllowOverride None
      Require all granted
      RewriteEngine on

      # 404 any direct .php request
      RewriteCond %{THE_REQUEST} \.php[/\s?] [NC]
      RewriteRule !^error - [R=404,L]

      # redirect to PHP
      # e.g. example.com/foo will show contents of example.com/foo.php
      RewriteCond %{REQUEST_FILENAME} !-f
      RewriteCond %{REQUEST_FILENAME} !-d
      RewriteRule ^(.*)$ $1.php [L]
</Directory>
```

Snippet 41: Apache, centralino.conf Directory

In questa sezione si definiscono le impostazioni relative ai file offerti dal server a partire dalla `DocumentRoot`. L'opzione `-Indexes` impedisce che vengano indicizzati i contenuti delle directory del sito, restituendo un errore di tipo '`403: Forbidden`' invece che la lista dei file presenti nell'interfaccia. Qui vengono definite anche le opzioni di Rewrite, che attraverso il modulo '`modrewrite`' di Apache, permettono di manipolare la visualizzazione delle pagine web. In particolare vengono definite regole per cui l'estensione delle pagine '`.php`' non venga visualizzata all'utente. Ogni richiesta esplicita di una pagina con estensione '`.php`' (se ad esempio si aggiunge '`.php`' nella barra degli indirizzi ad una pagina esistente) restituirà un errore di tipo '`404: Not found`'. Per ogni pagina richiesta senza estensione, ad esempio `admin/pickups` il server mostrerà il contenuto della pagina con estensione '`.php`'.

Maggiori informazioni sulla sintassi da utilizzare si possono trovare agli indirizzi:
https://httpd.apache.org/docs/current/mod/mod_rewrite.html
https://gist.github.com/mudge/22877
https://www.digitalocean.com/community/questions/apache-remove-php-and-html-file-extensions-using-mod_rewrite-in-httpd-conf

5.4.3 Logs

```
# Level: info and above
LogLevel info

# Error logfile
ErrorLog ${APACHE_LOG_DIR}/centralino-error.log
```

```
# Access logfile
# Pipe the access logfile to call rotatelogs to handle rotation
# usage is: rotatelogs /path/to/logfile seconds-before-rotating
# set to rotate every 24hours
CustomLog   "|/usr/bin/rotatelogs   ${APACHE_LOG_DIR}/centralino-access.log   86400"
combined
```
Snippet 42: Apache, centralino.conf Logs

In questa sezione viene definito il formato dei log per l'interfaccia web e la quantità di informazioni (verbosità) che essi conterranno. I log si dividono in '`error`' e '`access`' e per entrambi si è impostato il livello '`info`'; ciò significa che verranno registrati i messaggi in ordine di 'gravità' pari o superiore al livello '`info`'. I vari livelli per i file di log di Apache possono essere consultati all'indirizzo:
https://httpd.apache.org/docs/2.4/mod/core.html#loglevel
Per il log di tipo '`access`', viene utilizzata l'utility di apache '`logrotate`', che comprime e archivia i log più vecchi di un tempo specificato in secondi (in questo caso 24 ore, cioè 86400 secondi) gestendone la rotazione, per risparmiare spazio sul disco e mantenere i file leggibili.

5.4.4 Abilitare il sito in Apache

Per rendere effettivi i cambiamenti è necessario abilitare i moduli richiesti, abilitare la configurazione per 'centralino', controllare che la sintassi della configurazione sia corretta e riavviare il webserver Apache.

Si abilita il modulo rewrite:
```
sudo a2enmod rewrite
```

Si abilita l'interfaccia web (Apache creerà un collegamento in `/etc/apache2/sites-enabled/`):
```
sudo a2ensite centralino.conf
```

Si controlla che non vi siano errori di sintassi prima di riavviare apache:
```
sudo apache2ctl configtest
```

Si riavvia Apache per rendere effettivi i cambiamenti:
```
sudo systemctl restart apache2
```

5.5 Permessi Cartelle e File

Nel server, oltre all'utente `voip` e all'utente `asterisk` è presente l'utente `www-data` che viene usato per eseguire apache e php. Questi due utenti, '`asterisk`' e '`www-data`' non hanno accesso a una shell né una password, non è possibile quindi utilizzarli per accedere al

server. In questo modo, se un malintenzionato dovesse compromettere uno di questi servizi, non avrebbe comunque privilegi di amministrazione. La struttura dei file è la seguente:

Directory	Proprietario:Gruppo	Descrizione
`/srv/db`	`voip:database`	Classe e file di configurazione per la connessione con il database.
`/srv/centralino`	`voip:www-data`	Cartella '`public`' (front-end del sito) e cartella '`class`' (classi di back-end).

Tabella 3: Permessi interfaccia web

Gli utenti '`www-data`' e '`asterisk`' verranno aggiunti al gruppo '`database`' in modo che possano effettuare operazioni su di esso. I permessi che verranno impostati per i file saranno di lettura e scrittura per il proprietario, lettura per il gruppo e nessun permesso per gli altri utenti:
`chmod_files 640 /`*`percorso`*`/directory`

Per le cartelle, i permessi saranno di lettura, scrittura ed esecuzione per il proprietario, di lettura ed esecuzione per il gruppo e nessun permesso per gli altri utenti:
`chmod_dir 750 /`*`percorso`*`/directory`

Le classi di back-end Php `Pickup.php` e `Pharmacy.php` hanno come proprietario '`asterisk`' e come gruppo '`www-data`'. Questo è necessario poiché vengono utilizzate dall'Asterisk Gateway Interface per la prenotazione del ritiro e allo stesso tempo dall'interfaccia web.

6. Miglioramenti e Considerazioni

In questo capitolo conclusivo si indicheranno i miglioramenti che si possono apportare al server PBX sviluppato e delle considerazioni finali sullo svolgimento del progetto.

6.1 Possibili Miglioramenti

Pur avendo soddisfatto le richieste iniziali, l'implementazione scelta presenta degli aspetti migliorabili e permette di essere ampliata in futuro, se fosse necessario.

6.1.1 Protocolli Sicuri per Asterisk e l'Interfaccia Web

Un miglioramento essenziale per la sicurezza di qualsiasi sistema che utilizza la rete internet.
In Asterisk, infatti, il protocollo SIP (usato per stabilire una connessione VoIP) supporta il Transport Layer Security (TLS) ed è possibile quindi cifrare il traffico SIP con i dispositivi VoIP collegati, come con il provider SIP. Inoltre, è possibile utilizzare il protocollo SRTP (Secure Real-Time Transport Protocol) in modo che anche i pacchetti multimediali delle chiamate VoIP siano cifrati, garantendo un buon livello di sicurezza nel sistema PBX. Nella documentazione di Asterisk è disponibile una guida per questa configurazione, a questo indirizzo: https://wiki.asterisk.org/wiki/display/AST/Secure+Calling+Tutorial
Se si utilizza un provider SIP con il proprio server PBX, come in questo caso, anche il provider deve fornire queste funzionalità (SIP TLS e SRTP), altrimenti l'implementazione 'sicura' non è realizzabile. Al momento dello sviluppo del progetto non è stato possibile trovare documentazione sul sito di MessageNet, riguardo al supporto e alla configurazione del TLS per SIP e del protocollo SRTP.

Anche per l'interfaccia web è necessario utilizzare SSL/TLS per avere una connessione cifrata in modo che i pacchetti che l'utente scambia col server (come le informazioni di login) non vengano trasmessi in chiaro. Ci sono due modi per ottenere una connessione https sul server:

- Se non si dispone di un dominio registrato, si può creare un certificato SSL/TLS self-signed.
  In questo modo le connessioni saranno cifrate, ma i browser presenteranno un messaggio di errore nel visitare il sito, dicendo che il certificato è probabilmente scaduto o invalido e si dovrà aggiungere un'eccezione per il sito; questo avviene perché i certificati devono essere approvati da una CA (Certificate Authority): degli enti che rilasciano certificati e che sono riconosciuti da tutti i browser, aumentando la sicurezza e la 'fiducia' della connessione ad un sito.

- Ottenere un certificato da una CA è sicuramente la scelta migliore ma per fare ciò è necessario avere un dominio registrato per il sito. Non è possibile farsi rilasciare un certificato approvato da una CA per un indirizzo IP. Se si dispone di un dominio registrato, è disponibile la CA 'Let's Encrypt!' che rilascia certificati SSL/TLS in maniera gratuita.

6.1.2 Script di upload, backup e manutenzione

Si possono implementare diversi script per monitorare l'attività del server, inviare periodicamente i log e il backup della configurazione all'indirizzo email dell'amministratore di sistema (ciò richiede la configurazione del protocollo SMTP), inviare email all'amministratore in caso di eventi anomali.
Per facilitare la modifica dei vari file che compongono l'interfaccia web e la configurazione di Asterisk si può implementare uno script che stabilisce una connessione al server (tramite SSH, sftp o come si preferisce) e carica i file nelle cartelle appropriate, impostando permessi e ownership (proprietario e gruppo) adeguati.

6.1.3 Interfaccia web

Per quanto riguarda l'interfaccia web, è possibile migliorare la pagina relativa ai ritiri includendo:

- la possibilità di selezionare più elementi da eliminare per ogni tabella, particolarmente utile nel caso della tabella dei ritiri;
- la possibilità di ordinare la tabella in più modi (per farmacia, orario prenotazione, via, ecc…);
- i link a Google Maps con le indicazioni stradali per raggiungere la farmacia;
- e la possibilità per il Pony Express di segnare un ritiro come 'preso in consegna' o 'ritirato' e tramite Asterisk avvisare la farmacia del ritiro (sarà necessario stabilire con il provider VoIP un piano a pagamento per effettuare chiamate in uscita/inviare SMS).

6.1.4 Fail2ban

Fail2ban è un programma che monitora i log di sistema secondo dei filtri (preimpostati o personalizzati attraverso espressioni regolari) per intercettare comportamenti illeciti.
È possibile installarlo e configurarlo per monitorare Asterisk, Apache e tutti i servizi offerti dal server. Se il programma intercetta un possibile tentativo di intrusione l'IP in questione verrà bannato (escluso) dal server per un determinato periodo di tempo. Fail2ban è utilizzato, tra gli altri, anche da FreePBX e pur non essendo garanzia di sicurezza assoluta è uno strumento utile che potrebbe essere aggiunto al server.

6.1.5 PBX

Per la parte del server relativa al PBX, sarà sicuramente possibile espandere l'infrastruttura collegando uno o più telefoni VoIP, nel caso fosse necessario rispondere di persona alle chiamate in arrivo sul centralino. Se la configurazione PBX da sviluppare diventasse piuttosto estesa, si potrebbe pensare di implementare un'interfaccia web simile a quella di FreePBX per gestire le impostazioni di Asterisk o di utilizzare direttamente una distribuzione come FreePBX, se si riterrà opportuno utilizzarne le funzionalità offerte e/o i moduli a pagamento.

Altri miglioramenti riguardanti questa parte del server possono essere:

- Dotando i corrieri di un dispositivo di geolocalizzazione (ad esempio V-Auto, sviluppato da Vodafone) consentire al farmacista di effettuare una prenotazione urgente: il server, attraverso Asterisk, invierà un SMS al corriere più vicino alla farmacia per notificargli il ritiro urgente appena prenotato. Per permettere questa funzionalità sarà necessario attivare il servizio SMS presso il provider VoIP scelto.
- Aggiungere nel dialplan di Asterisk la possibilità per il farmacista di controllare lo stato della prenotazione e di annullarla nel caso questa fosse errata.

6.2 Considerazioni Finali

Lo sviluppo di questo progetto ha permesso di entrare in contatto con realtà mai affrontante prima come la tecnologia VoIP, Asterisk e l'implementazione di un Server PBX.

La possibilità di mettere insieme linguaggi e tecnologie differenti (database MySQL, programmazione in php, javascript, SSH, Apache, script bash) per realizzare un singolo prodotto finale è risultata molto stimolante e formativa.

La difficoltà maggiore si è rivelata capire il funzionamento della tecnologia VoIP in un sistema PBX, principalmente perché le informazioni, ad eccezione del sito di Asterisk, sono molto frammentate e a volte poco aggiornate. Inoltre sono presenti diverse compagnie che offrono servizi di questo tipo e molto spesso le informazioni reperibili sono volutamente poco esaustive, al fine di guadagnare su consulenze e supporto.

Sitografia

https://www.cisco.com/c/en/us/products/unified-communications/what-is-voip.html
http://www.voipvoice.it/faq.html
https://wiki.asterisk.org/wiki/
https://www.voip-info.org/wiki/
https://wiki.freepbx.org/
https://messagenet.com/it/voip/sw/asterisk.html
https://tools.ietf.org/html/rfc3261
https://www.tldp.org/HOWTO/Shadow-Password-HOWTO-2.html
http://www.linfo.org/partition.html
http://www.linfo.org/mount_point.html
http://www.linfo.org/swap_space.html
https://www.centos.org/docs/5/html/Deployment_Guide-en-US/ch-lvm.html
https://stribika.github.io/2015/01/04/secure-secure-shell.html
https://www.digitalocean.com/community/tutorials/
https://dev.mysql.com/doc/
http://www.culttt.com/2013/02/04/how-to-save-php-sessions-to-a-database/
https://secure.php.net/manual/en/book.pdo.php
https://www.regexbuddy.com/regex.html
https://www.stackoverflow.com
https://letsencrypt.org/
https://www.draw.io/
http://fromtexttospeech.com
http://www-db.deis.unibo.it/courses/SIG/ER6.pdf