UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTA' DI INGEGNERIA

Corso di Laurea triennale Ingegneria Informatica e della Automazione

Realizzazione di un PBX su cloud

Creation of a PBX on the cloud

Relatore: Chiar.mo

Prof. Spalazzi Luca

Tesi di Laurea di:

Malloni Matteo

A.A. 2020 / 2021

INDICE

Introduzione

Il lavoro svolto durante il tirocinio, su cui si basa la mia tesi, è stato effettuato presso la New Assistent srl, azienda che grazie alle esperienze e alle collaborazioni acquisite negli anni, ha sviluppato sistemi di telecomunicazioni in grado di gestire in maniera efficace ed efficiente, le esigenze di aziende e studi professionali. Dal 2008 è presente sul mercato della telefonia con prodotti pensati per garantire affidabilità, flessibilità, diminuzione dei costi, e una qualità del servizio in continua crescita.

L'obiettivo dell'azienda e quindi del progetto, è quello di realizzare un centralino VoIP (Voice over IP: dove le chiamate passano su rete IP) su cloud in grado di evitare possibili disservizi, garantendo quindi un uptime del 99,9%, e mantenendo costi bassi per il cliente, cosa che nel centralino tradizionale non accade.

Il centralino tradizionale infatti, nasce come sistema chiuso. Esso viene gestito da un software installato sul server che utilizza linee telefoniche analogiche. Ogni centralino è diverso dall'altro poiché ogni azienda, avendo esigenze diverse, richiede una configurazione diversa e di conseguenza risorse e componenti hardware differenti. Il centralino così realizzato, genera, in caso di malfunzionamento, dei disservizi potenzialmente importanti, legati al fatto che il tecnico deve operare fisicamente sul server situato in azienda.

Il centralino virtuale pensato e sviluppato dalla New Assistent srl è un centralino VoIP su cloud multi-tenant installato su una macchina virtuale, presente nel nostro datacenter. I server del datacenter sono gestiti tramite Proxmox (che vedremo nel paragrafo 3.1), il quale ci permette di migrare o clonare tutte le macchine virtuali presenti su server differenti, in pochi secondi. Questo ci

garantisce un uptime elevato anche nel caso di un guasto hardware ad uno dei server.
Nel capitolo 1 vedremo nello specifico quali sono le debolezze di un centralino tradizionale che ci ha spinto alla creazione di un centralino virtuale su cloud.

Il mio contributo, all'interno di questo progetto, è stato quello di realizzare:

- Un real-time Interactive Panel che permette di visualizzare tutto quello che succede in tempo reale nel centralino, e di interagire con la parte telefonica tramite PC o smartphone (capitolo 2 e capitolo 3)

- Le procedure per una maggiore sicurezza che ci garantiscono protezione dei dati e immunità da eventuali attacchi informatici (capitolo 4)

Capitolo 1

Motivazioni

1.1 - Cos'è il VoIP e come funziona

Il termine VoIP, letteralmente, significa Voice over IP, cioè voce tramite il protocollo internet.

Si può quindi parlare di tecnologia VoIP, ogni volta che si effettua una telefonata utilizzando come rete di transito per la voce, la stessa che si utilizza per la navigazione in rete.

Più nel dettaglio, con VoIP si intende l'insieme dei protocolli di comunicazione di strato applicativo che rendono possibile questo tipo di comunicazione. Grazie a numerosi provider VoIP, è possibile effettuare telefonate anche verso la rete telefonica tradizionale (PSTN).

In realtà, più in generale, il VoIP consente una comunicazione audio-video su sistema real-time, su rete a pacchetto (es. videotelefonata, videochiamata e videoconferenza). La tecnologia in questione richiede che il segnale analogico venga preventivamente codificato in formato digitale tramite conversione analogico-digitale, eventualmente compresso tramite l'uso di codec audio e infine trasmesso a pacchetto in rete grazie all'utilizzo congiunto in parallelo di due tipi di protocolli di comunicazione:

- una per l'elaborazione della conversazione (che richiede fasi come la ricostruzione del frame audio, sincronizzazione, identificazione del chiamante).
- una per il trasporto dei dati (pacchetti voce su IP).

Le conversazioni VoIP non devono necessariamente viaggiare su Internet, ma possono anche usare come mezzo trasmissivo una qualsiasi rete privata basata sul protocollo IP, per esempio una LAN all'interno di un edificio o di un gruppo di edifici. I protocolli usati per codificare e trasmettere le conversazioni VoIP sono solitamente chiamati Voice over IP protocols.

Per il trasporto dei dati, nella grande maggioranza delle implementazioni VoIP, viene adottata una pila di protocolli di comunicazione quali RTP (Real-time Transport Protocol), UDP (User Datagram Protocol) e IP (Internet Protocol). L'utilizzo del protocollo UDP invece del protocollo TCP (Transmission Control Protocol), si giustifica con il fatto che una comunicazione real-time come la fonia non può tollerare ritardi o latenze aggiuntive o troppo elevate dovute alla ritrasmissione dei pacchetti persi e agli ACK.

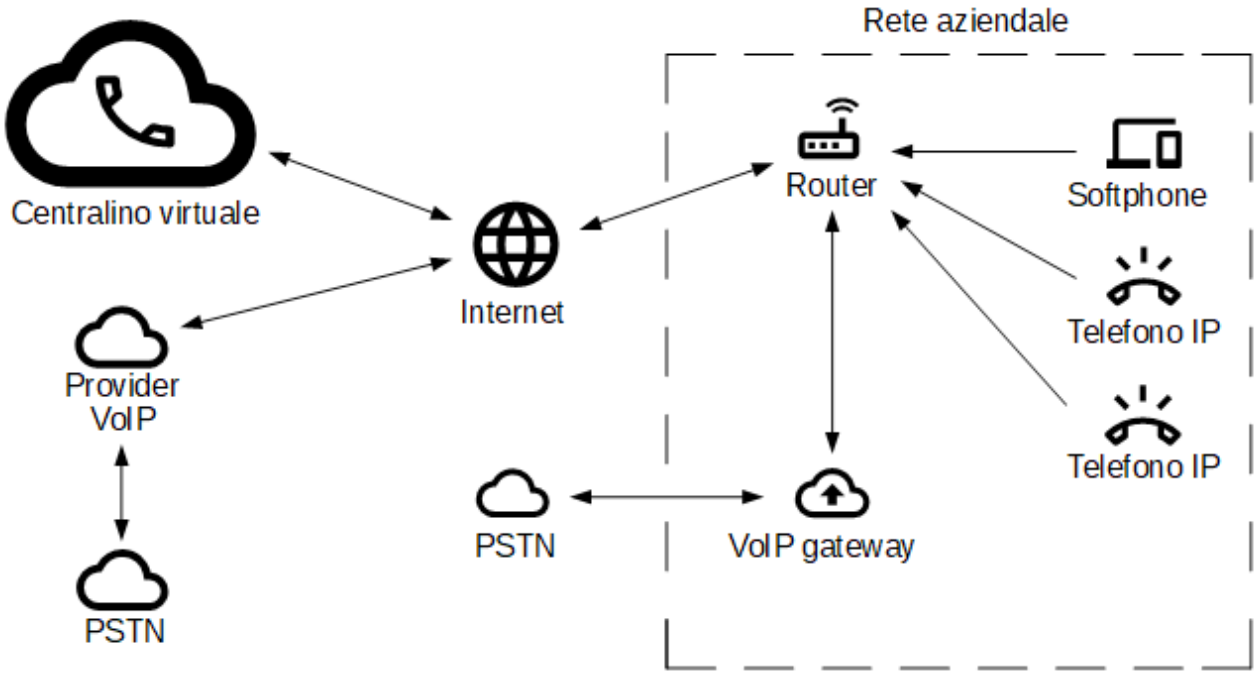
Per quanto vada bene qualunque connessione Internet per riuscire a telefonare, è utile sapere che il VoIP non richiede grandi dimensionamenti di banda; infatti con i metodi di codifica e decodifica moderni, una telefonata occupa circa 33kbps bidirezionali. In pratica con una linea adsl da 7 Mb che solitamente ha 512kbps in upload si riescono a fare circa 12 chiamate contemporanee; ovviamente considerando che oltre al traffico audio vero e proprio, una parte della banda sarà utilizzata per il traffico di gestione e segnalazione dei vari protocolli di rete necessari al funzionamento dell’intero sistema.

Nelle aziende, spesso viene utilizzata la stessa connessione sia per i dati che per la voce. Quando però ci sono molti interni e molti numeri attivi, è consigliabile dedicare una connettività al servizio voce per renderlo indipendente dalla connessione principale. Questo perché ad oggi, visto il notevole aumento dei servizi in cloud, accade spesso che in azienda si effettuino pesanti operazioni di

upload o download che portano quindi alla saturazione della connessione internet. Per questo motivo è consigliabile separare la rete dati da quella telefonica, o fisicamente utilizzando appunto connessioni separate, oppure tramite un Router/Firewall (noi ad esempio, per le nostre aziende utilizziamo pfSense).

Di seguito un esempio di configurazione tipica aziendale.

Schema 1 [1]



1 : Le icone presenti nel diagramma, sono le Material icons di Google visibili e scaricabili al link: https://fonts.google.com/icons

1.2 - Centralino virtuale o tradizionale? Pro e contro

Il centralino telefonico può contribuire in maniera determinante alla produttività e redditività dell'azienda.
Il centralino tradizionale, viene installato in un server o un PC situato fisicamente all'interno dell'azienda. Nella maggior parte dei casi però, è gestito da fornitori esterni all'azienda, ed offre le stesse funzionalità da sempre: gestione di più linee e delle eventuali code di chiamata, risponditore automatico, trasferimento della conversazione ad un altro interno telefonico.

Questo tipo di centralino, ha numerosi limiti e svantaggi:

- Il *costo* può essere proibitivo specialmente per le piccole e medie imprese. È necessario acquistare hardware (server, cuffie o telefono, computer, terminale fax), licenze e predisporre lo spazio fisico in ufficio. Inoltre, è necessario considerare i costi ricorrenti di gestione: in media, le installazioni tradizionali devono essere sostituite ogni 5-7 anni a causa dell'invecchiamento dell'hardware e dei progressi del software.

- *Mancanza di flessibilità.* Una volta installato, il centralino tradizionale è difficile da gestire: aggiungere agenti o operatori all'azienda comporta l'acquisto di nuovi telefoni e la modifica dell'hardware. Queste modifiche creano costi aggiuntivi a breve termine. Se è necessario ridimensionarli a causa della stagionalità o di eventi imprevedibili, l'azienda resterà con un surplus di dispositivi non necessari. Per quanto riguarda la mobilità, i

telefoni da scrivania devono essere fisicamente collegati al server in ufficio, rendendo praticamente impossibile lo smart working.

- *Affidabilità debole.* Molte aziende si affidano ai centralini tradizionali per evitare la latenza o la qualità delle chiamate instabili. Mentre i problemi relativi alla qualità della voce possono essere potenzialmente evitati utilizzando il servizio telefonico tradizionale, fare affidamento all'hardware fisico è un punto debole di per sé. L'apparecchiatura potrebbe rompersi o diventare incompatibile con il software. La tecnologia del centralino cloud al contrario è immune dai problemi hardware. Inoltre, la maggior parte dei problemi di affidabilità e qualità della chiamata comunemente attribuiti al VoIP non sono dovuti a guasti della tecnologia stessa, ma piuttosto a una connessione internet debole o instabile.

- *Raramente può integrarsi* con i software aziendali, ma nei casi in cui è possibile, il costo è abbastanza elevato, in quanto bisogna considerare licenze e tempo materiale del tecnico che dovrà effettuare l'installazione e le varie configurazioni.

Questi limiti, ci hanno spinto a realizzare la nostra versione di centralino telefonico VoIP in cloud, il quale, a differenza del centralino tradizionale, ha i seguenti vantaggi:

- *Economico:* Non occorre acquistare hardware, basta avere una semplice connessione ad Internet e dispositivi con accesso a Internet. Di norma, gli

utenti vengono fatturati su base mensile o annuale. Il software per centralino in cloud è molto più facile da scalare, gestire e personalizzare. Aggiungere o rimuovere utenti è molto semplice come gestire un qualsiasi abbonamento.

- *Flessibile:* Un centralino cloud, può essere configurato autonomamente tramite un'interfaccia web, permette di gestire un'infinità di casi, e per questo motivo può adattarsi a qualsiasi tipo di azienda. Inoltre, non dipendendo da hardware fisico, può essere utilizzato, senza alcun problema, lontano dalla sede aziendale (ad esempio in smart working).

- *Integrazioni facili:* Nella maggior parte dei casi, si possono integrare facilmente sistemi CRM, ticket di helpdesk, modelli di sondaggio e molto altro.

- *Manutenzione nulla:* la manutenzione dei server e dell'hardware è a cura del provider cloud. L'azienda non deve più preoccuparsi delle manutenzioni hardware, in quanto non ha nulla di fisico presso la sede aziendale. Questo inoltre, nel nostro caso, garantisce ai clienti un uptime del servizio del 99,9% (su base annuale).

- *Sicurezza e privacy.*

- *Approccio ecologico:* Accumulare, sostituire ed eliminare l'hardware non è ecologico. Senza infrastrutture e apparecchiature proprie, le azienda che utilizzano risorse cloud diventano più green.

Pro e Contro del Centralino Tradizionale

Pro	Contro
✔ Pieno controllo di tutte le sue funzioni e della sua gestione ✔ I dati sono memorizzati nel server interno all'azienda ✔ Privacy	✖ Costi elevati ✖ Mancanza di flessibilità ✖ Affidabilità debole ✖ Difficoltà di integrazione con i CRM aziendali

Pro e Contro del Centralino VoIP in Cloud

Pro	Contro
✔ Economico ✔ Flessibile ✔ Facile integrazione con i CRM aziendali ✔ Costi manutenzione nulli ✔ Privacy ✔ Ecologico	✖ I dati dell'infrastruttura telefonica sono in mano a un soggetto esterno all'azienda ✖ Connessione internet di buona qualità

Capitolo 2

Tecnologie utilizzate

2.1 – Asterisk PBX

Per la realizzazione del core del nostro centralino cloud, abbiamo deciso di utilizzare Asterisk, che è un software open-source che permette di gestire una rete telefonica privata e di connetterla a servizi di telefonia PSTN o VoIP.
Le componenti principali del core di Asterisk sono:

- *Command Line Interface (CLI)*: consente di monitorare e controllare l'esecuzione di Asterisk, sia se questo viene lanciato come demone, sia se viene lanciato come applicazione stand-alone dalla linea di comando.

- *Dialplan*: è l'insieme di regole applicate dal centralino in risposta agli eventi telefonici (sia entranti che uscenti).

- *Applicazioni*: consentono al centralino di effettuare diverse azioni in risposta agli eventi del dialplan, estendendo i possibili servizi disponibili dal centralino. Questi sono, a tutti gli effetti, i "comandi" utilizzabili nel dialplan. Le applicazioni (che sono personalizzabili) consentono, ad esempio, di inviare e ricevere fax, di usufruire di una segreteria telefonica (voice mail), di lanciare comandi di sistema, e così via.

- *Risorse e moduli*: sono estensioni del centralino che è possibile caricare in memoria o rilasciare durante l'esecuzione di Asterisk, senza la necessità di riavviare il servizio. Molte delle applicazioni sono fornite sotto forma di moduli (eccetto quelle basilari che sono naturalmente integrate nel core di Asterisk).

Asterisk svolge anche le funzioni di SIP server o SIP registrar, e quindi si occupa di gestire sia le registrazioni dei vari endpoint (tramite username e password), sia di gestire tutte le chiamate SIP in cui sono coinvolti gli endpoint registrati.
Più nel dettaglio, il SIP server può:

- settare una sessione tra due o più endpoint (peer o interni telefonici).

- negoziare i parametri multimediali e le specifiche per la sessione di ciascun endpoint utilizzando il protocollo SDP (Session Description Protocol).

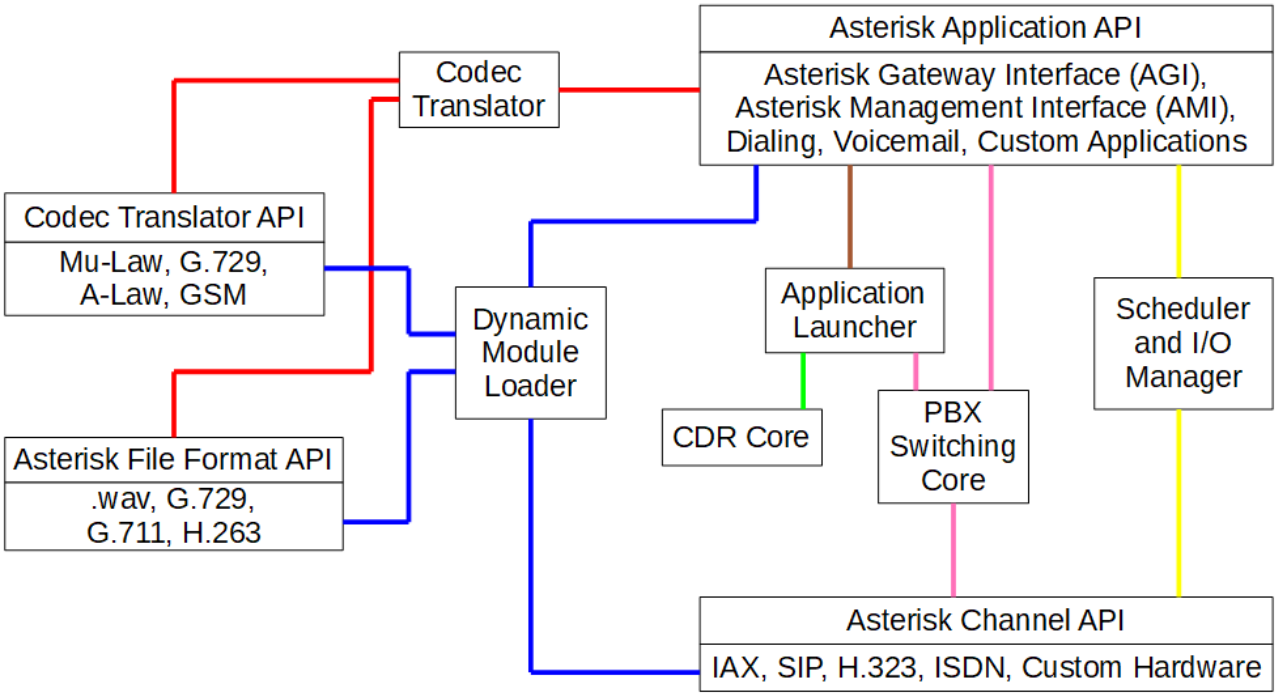
- regolare i parametri multimediali e le specifiche di una sessione mentre la sessione è in corso (ad esempio quando un interno mette una chiamata in attesa).

- sostituire un endpoint con un nuovo endpoint (è il caso in cui un interno telefonico trasferisce la chiamata ad un altro interno).

- terminare la sessione.

Per interfacciare Asterisk con il server CTI (Computer Telephony Integration) e di conseguenza con l'Interactive Panel, abbiamo utilizzato il protocollo AMI (Asterisk Manager Interface), il quale consente di comunicare/interagire con gli endpoint, le chiamate, gli agenti, il dialplan, ecc.

Diagramma dell'architettura Asterisk



2.2 - Integrazione CTI (Computer Telephony Integration)

Per la realizzazione dell'Interactive Panel, abbiamo deciso di utilizzare l'integrazione CTI.
Con questo termine, viene indicata qualsiasi tecnologia che permette interazioni tra un telefono ed un computer consentendone il coordinamento integrato.
Le funzioni desktop comunemente fornite dalle applicazioni CTI includono:

- Screen popping: Consiste nella visualizzazione delle informazioni della chiamata, e Screen Pop alla risposta, con o senza l'utilizzo dei dati della linea del chiamante. Normalmente questo viene utilizzato per la ricerca dei dettagli del chiamante in un'applicazione business.

- Composizione: Composizione automatica e composizione controllata dal computer (Power Dialer, Preview Dialer, e Predictive Dialer).

- Controllo del telefono: Include il controllo della chiamata e il controllo delle caratteristiche.

- Trasferimenti: Trasferimenti telefonici e di dati, coordinati tra due parti.

- Call center: Permette agli utenti di effettuare il login come agenti di call center e controllare il proprio stato di agente.

- Call routing: Instradamento automatico delle chiamate ad una nuova destinazione.

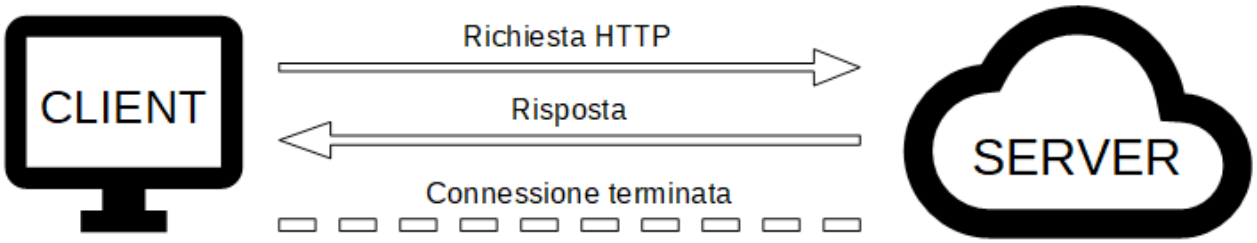
- Funzioni di report avanzato della chiamata: Utilizzo dei dati dettagliati forniti da CTI per offrire report migliori sulle chiamate.

- Integrazione con la registrazione vocale: Utilizzo dei dati provenienti da CTI per arricchire i dati memorizzati per ogni chiamata registrata.

2.3 – Protocollo HTTP e WebSocket

Nel nostro caso, la parte riguardante l'integrazione CTI è stata realizzata tramite un pannello web per l'interfaccia utente, che utilizza un WebSocket per la connessione tra il browser e il server CTI su cui è installato *amiws* (che vedremo nel paragrafo 2.4).
HTTP e WebSocket sono entrambi protocolli di comunicazione utilizzati nella comunicazione client-server.
Il protocollo HTTP è di tipo unidirezionale in cui il client invia una richiesta e il server invia la relativa risposta, chiudendo la connessione subito dopo.
Quindi ad ogni richiesta HTTP o HTTPS viene stabilita una nuova connessione client-server e dopo che il client ha ricevuto la risposta, la connessione viene terminata automaticamente.
HTTP è un protocollo *"stateless"* eseguito sulla parte superiore del TCP (Transmission Control Protocol) che è un protocollo orientato alla connessione e garantisce la consegna del trasferimento dei pacchetti di dati, utilizzando i metodi di handshake a tre vie e ritrasmette i pacchetti persi.

Schema 2 [2]



Il protocollo WebSocket invece, è bidirezionale. Esso infatti è un protocollo full-duplex che viene utilizzato nello stesso scenario di comunicazione client-server, ma a differenza del protocollo HTTP inizia da ws:// o wss://.

È un protocollo *"stateful"*, ciò vuol dire che la connessione tra client e server rimarrà attiva (con il continuo scambio di messaggi in modalità bidirezionale) fino a quando non verrà terminata da una delle parti (il client o il server). Se la connessione viene chiusa dal client o dal server, la connessione viene terminata per entrambi.

Di solito è consigliabile utilizzare un WebSocket nei seguenti casi:

- Applicazioni Web in tempo reale: è il nostro caso, e riguarda appunto la creazione dell'Interactive Panel. Le applicazioni Web in tempo reale utilizzano un WebSocket per stabilire una connessione al server di backend e mostrare tutti i dati che il client riceve. Con il WebSocket infatti, tutti i dati vengono continuamente trasmessi nella connessione

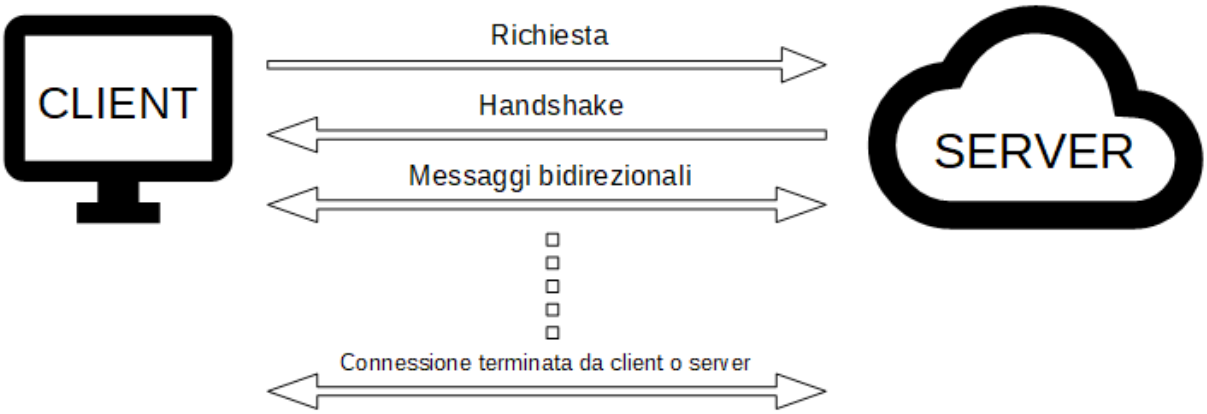
2 Le icone presenti nel diagramma, sono le Material icons di Google visibili e scaricabili al link: https://fonts.google.com/icons

già aperta, e questo migliora notevolmente le prestazioni dell'applicazione.

- Applicazioni di gioco: in un'applicazione di gioco, i dati vengono continuamente inviati dal server di backend al browser, e senza aggiornare l'interfaccia utente, si avranno degli output a video che consentiranno all'utente di giocare con elevate prestazioni.

- Applicazioni di messaggistica (chat): anche in questo caso è consigliato l'utilizzo del WebSocket. Si stabilisce la connessione una sola volta, e poi lo scambio dei messaggi tra i vari utenti, avverrà in tempo reale, proprio come nel caso delle applicazioni real-time.

Schema 3 [3]

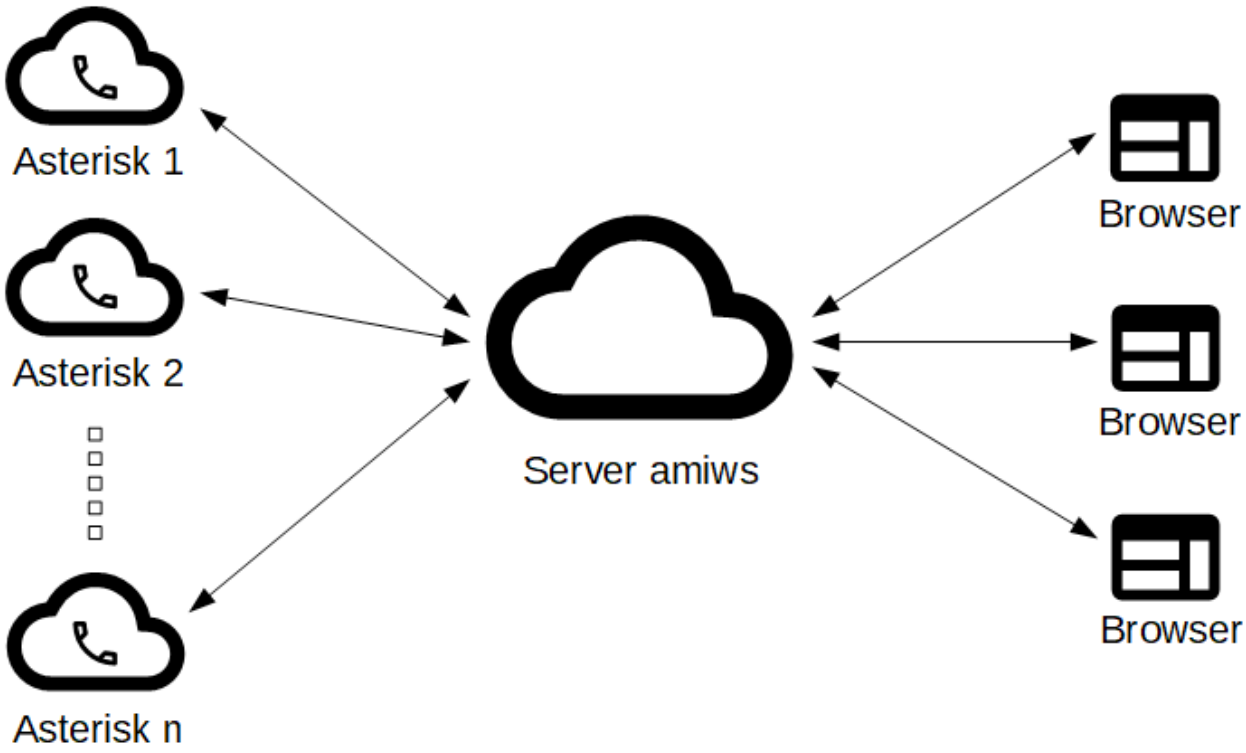


3 Le icone presenti nel diagramma, sono le Material icons di Google visibili e scaricabili al link: https://fonts.google.com/icons

2.4 – A*miws:* Asterisk Manager Interface (AMI) to WebSocket proxy

Amiws è un semplice proxy da AMI (Asterisk Manager Interface) a WEB.
Esso può connettersi ad uno o più server Asterisk tramite l'interfaccia AMI, e può sia ricevere o leggere eventi AMI dello stream, che inviare comandi o azioni. Gli eventi ricevuti vengono analizzati e convertiti in JSON (JavaScript Object Notation), i quali (tramite un'interfaccia HTTP/WebSocket) vengono poi inviati a tutti gli utenti collegati tramite HTTP (quindi agli utenti che sono loggati nel nostro Interactive Panel su browser web).

Schema 4 [4]



4 Le icone presenti nel diagramma, sono le Material icons di Google visibili e scaricabili al link: https://fonts.google.com/icons

Di seguito è riportato un esempio di evento ricevuto da Asterisk e convertito in JSON da *amiws* che sarà poi inviato all'Interactive Panel:


```
{
  "type": 3,
  "server_id": 1,
  "server_name": "asterisk01.local",
  "ssl": false,
  "data": {
        "Event": "Hangup",
        "Privilege": "call,all",
        "Channel": "SIP/17550-00000a6a",
        "ChannelState": "6",
        "ChannelStateDesc": "Up",
        "CallerIDNum": "1234567890",
        "CallerIDName": "Chiara",
        "ConnectedLineNum": "<unknown>",
        "ConnectedLineName": "<unknown>",
        "AccountCode": "",
        "Context": "from-interni-175",
        "Exten": "0734123456",
        "Priority": "8",
        "Uniqueid": "1495919040.5315",
        "Linkedid": "1495919040.5315",
        "Cause": "0",
        "Cause-txt": "Unknown"
  }
}
```

Il “type” corrisponde al tipo di messaggio ricevuto e può assumere i seguenti valori:

- 0: Unknown
- 1: Prompt
- 2: Azione
- 3: Evento
- 4: Risposta
- 5: Risposta con output
- 6: Elenco code. AMI restituisce l'elenco delle code non come pacchetto AMI ma come testo (come ad esempio accade con il comando “queue show”).

Il “server_id” corrisponde all’id univoco del server sul quale è installato Asterisk, ed è generato da *amiws* in fase di configurazione (infatti ad un server *amiws* possiamo collegare N server Asterisk).

Il “server_name” corrisponde al nome del server Asterisk assegnato nel file di configurazione che ha come id, il valore del server_id visto sopra.

Il flag “ssl” può assumere valore 1 o 0, per indicare se la connessione AMI è criptata con SSL, oppure no.

Il campo “data” è un messaggio AMI che contiene tutte le informazioni dell’evento ricevuto.

Al contrario, tramite l'Interactive Panel è possibile inviare un messaggio JSON al server *amiws,* che si occuperà di convertirlo ed inviarlo al server Asterisk.
Per inviare il messaggio ad un determinato server Asterisk, è necessario settare nell'header del messaggio il valore del campo "AMIServerID" che corrisponde al campo "server_id" settato precedentemente nel file di configurazione.

Di seguito un esempio di messaggio JSON:

```
sock.send(JSON.stringify({"Action": "CoreStatus", "AMIServerID": 1}));
```

Capitolo 3

Centralino come servizio Cloud

3.1 – Architettura del centralino

Come già anticipato, tutti i nostri server necessari al funzionamento della parte telefonica, sono gestiti tramite Proxmox.

Proxmox è una piattaforma di gestione server completa e open-source per la virtualizzazione aziendale. Integra al suo interno l'hypervisor KVM (Kernel-based Virtual Machine) e Linux Containers (LXC), storage definito dal software e funzionalità di rete, il tutto in un'unica piattaforma.

Il nostro centralino virtuale è stato quindi configurato su una macchina virtuale, installata e gestita tramite Proxmox, e come sistema operativo, è stata utilizzata la distribuzione Linux Debian.

Abbiamo assegnato una singola CPU con 4 core, 4GB di memoria RAM e 32GB di memoria su Hard Disk SSD. Questi componenti ci permettono di gestire, senza alcun problema, fino a 1000 peer o endpoint (cioè fino a 1000 interni telefonici).

Oltre il centralino vero e proprio (dove è installato ed eseguito Asterisk), abbiamo un secondo server (collegato al principale dove risiede Asterisk) che funge da server CTI (sul quale è installato *amiws*), ed un terzo server (comunicante sia con il server Asterisk che con il server CTI) che funge da Hosting e sul quale sono memorizzati i file del pannello web.

Per la seconda macchina virtuale che costituisce il server CTI, abbiamo assegnato una singola CPU con 4 core, 8GB di memoria RAM e 32GB di memoria su Hard Disk SSD. Con questi componenti, siamo riusciti a configurare

in *amiws* fino a 10 server Asterisk, e non abbiamo notato alcun tipo di problema o rallentamento.
Anche in questo caso, abbiamo utilizzato come sistema operativo, Debian.
Infine, per la terza macchina virtuale che costituisce il server web, abbiamo assegnato una singola CPU con 8 core, 16GB di memoria RAM e 128GB di memoria su Hard Disk SSD. Per la configurazione e la gestione di tutta la parte web, abbiamo utilizzato Plesk, e come sistema operativo, Ubuntu server.
Plesk è un pannello tecnico che permette di gestire con un'unica interfaccia web, tutte le possibili funzionalità legate all'utilizzo dei domini caricati (creazione e gestione dei database e dei relativi utenti, gestione di Apache, impostazioni PHP, ecc).

Tutta la parte telefonica è personalizzabile, e il tutto è gestito appunto tramite un pannello web di configurazione.
E' possibile caricare linee VoIP, aggiungere interni telefonici, settare le regole di instradamento in ingresso e in uscita (inbound e outbound) e poi procedere con la programmazione vera e propria.
Le funzionalità messe a disposizione del cliente sono in continua evoluzione, e le principali ad oggi disponibili sono le seguenti:

- Condizione temporale: Consente di deviare la chiamata ad un messaggio oppure ad un interno, dopo aver verificato se la chiamata è pervenuta in orario di apertura o di chiusura.

- Risponditore automatico: E' chiamato anche IVR (Interactive Voice Response) e consente di riprodurre un messaggio al chiamante ed

eseguire delle azioni in base ai tasti digitati. E' possibile ad esempio indirizzarlo verso un interno nel caso il chiamante digiti il tasto 1 o verso un altro interno nel caso digiti il tasto 2.

- Gruppo di chiamata: Permette di raggruppare uno o più interni e consente di far squillare tali interni contemporaneamente.

- Coda di chiamata: Consente di gestire diversi operatori e far ricevere loro chiamate in entrata. A differenza di un gruppo di chiamata, consente di gestire l'ordine di arrivo delle chiamate, far avanzare determinate chiamate in base a delle priorità e riprodurre al chiamante il tempo di attesa stimato prima di parlare con un operatore.

- Stanza di conferenza: Con questo modulo è possibile aprire delle stanze di conferenza raggiungibili anche da numeri esterni.

- Stanza di videoconferenza: Con questo modulo è possibile aprire delle stanze di videoconferenza raggiungibili anche da numeri esterni. Per la realizzazione, abbiamo integrato il servizio Jitsi Meet che è un'applicazione JavaScript WebRTC open-source che utilizza Jitsi Videobridge, ossia un componente server (anch'esso open-source) progettato per essere in grado di eseguire contemporaneamente migliaia di flussi video su un singolo server.

- Riconoscimento codice utente: Consente di inserire nella programmazione telefonica la lettura di un codice identificativo personale

(PIN). A livello pratico, si potrà fare in modo che ciascun chiamante, prima di contattare un determinato punto della programmazione telefonica, debba inserire il suo codice identificativo di riconoscimento. Il controllo di questo codice sarà effettuato tramite la connessione su un database mysql esterno correttamente configurato.

- Riconoscimento del chiamante: Può essere utilizzato per bloccare o garantire l'accesso a determinati punti della programmazione telefonica, da parte di determinati numeri telefonici. Ci permette quindi di creare blacklist e whitelist.

- Fax to Mail: Consente la ricezione dei fax in formato elettronico direttamente su una casella mail.

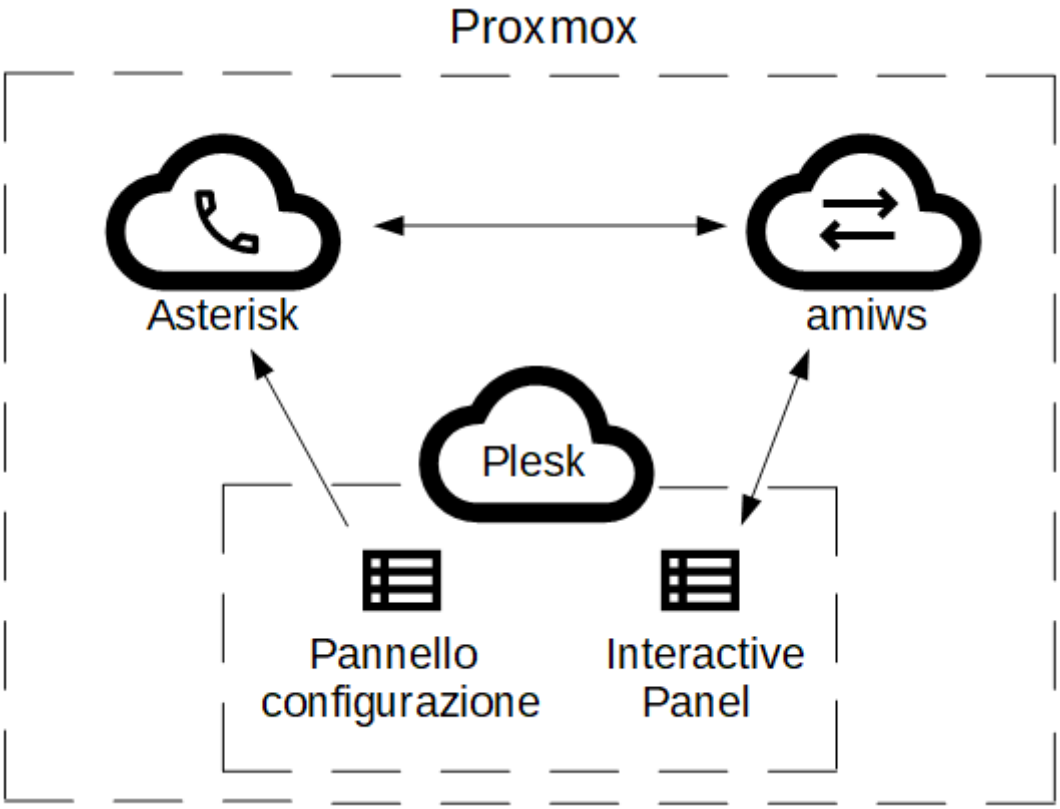
- Rubrica centralizzata: E' possibile caricare o importare da file CSV, un elenco di contatti in rubrica. Questa rubrica, oltre ad essere utilizzabile da web tramite l'Interactive Panel, è sincronizzata automaticamente in tutti gli interni telefonici, ed è quindi utilizzabile anche dal telefono fisico.

- Text to Speech (TTS): Tramite l'utilizzo delle Google API, è possibile scrivere un messaggio di testo, che verrà convertito in file audio, e che sarà possibile inserire in qualsiasi punto della programmazione telefonica.

- Registrazione audio: E' possibile caricare file audio in formato wav.

Terminata la configurazione della programmazione telefonica, è possibile "applicare la configurazione". Il sistema va quindi a "leggere" la programmazione dal database MySQL, e poi andrà a generare i file di configurazione per Asterisk (che verranno inseriti nella directory /etc/asterisk). Fatto ciò, la configurazione è immediatamente operativa.

Schema 5 [5]



5 Le icone presenti nel diagramma, sono le Material icons di Google visibili e scaricabili al link: https://fonts.google.com/icons

3.2 – Real-time Interactive Panel

Come anticipato nei capitoli precedenti, l'Interactive Panel realizzato, consente di monitorare in ogni momento tutto ciò che sta succedendo nel centralino telefonico. Ad esempio possiamo controllare quanti interni sono occupati e da quanto tempo, quanti chiamanti sono in attesa nelle code telefoniche e molto altro.
All'atto pratico, per la realizzazione di questo pannello, abbiamo utilizzato un nostro framework, cioè un'architettura logica di supporto utilizzata per la progettazione e la realizzazione di un software, che facilita e velocizza lo sviluppo da parte del programmatore.

I linguaggi di programmazione utilizzati sono:

- PHP, Javascript, Jquery per la parte web.

- SQL per l'inserimento, la modifica e la lettura dei dati dal database.

- C++ per *amiws* sul server CTI.

Per la parte grafica inoltre, abbiamo integrato ed utilizzato Bootstrap.
Bootstrap è una raccolta di strumenti grafici, stilistici e di impaginazione che permettono di avere a disposizione una grande quantità di funzionalità e di stili modificabili e adattabili a seconda delle proprie esigenze.

La sua funzione principale è il responsive web design, cioè è in grado di adattarsi dinamicamente in base alla grandezza ed alle caratteristiche del dispositivo che viene utilizzato per navigare sul web.
Questa sua peculiarità, ci è risultata molto comoda nello sviluppo, perché grazie ad essa siamo riusciti ad avere un pannello utilizzabile da PC, Smartphone e Tablet, senza avere alcuna distinzione di funzionalità tra un dispositivo e l'altro.

Per accedere all'Interactive Panel, è necessario effettuare un'autenticazione tramite username e password, che ovviamente sono differenti per ogni interno telefonico. Queste credenziali sono generate automaticamente dal sistema quando, in fase di configurazione, viene caricato e configurato un nuovo interno telefonico.
Una volta effettuata la corretta autenticazione quindi, il software ci reindirizzerà al pannello vero e proprio, dove automaticamente in background (se il browser utilizzato supporta HTML5 con l'utilizzo dei WebSocket) verrà effettuata la connessione WebSocket al server CTI.
Se tutto va a buon fine, la connessione è stabilita, e il server CTI inizierà ad inviare tutti i messaggi (che converte in JSON dopo averli ricevuti dal server Asterisk) al browser dell'utente collegato.

Di seguito una porzione del codice utilizzato, eseguito in background:

```
var wsproto = "wss://",
wsprotook = "",
hostname = "cti.nwpbx.it",
port = 8777;

function debug(b) {
	console.log(b);
}

function pre_init() {
	debug("inizio");
	if ("WebSocket" in window) {
		debug("Il browser ha HTML5 web sockets!");
	} else {
		debug("Websocket non abilitato");
	}
	connectXML();
}

function connectXML() {
	debug("Connessione xml");
	if ("WebSocket" in window) {
		var b = wsproto + hostname + ":" + port + "/?&amiserverid=" + pbx_amiserverid + "/clientid=" + pbx_clientidurl + "/";
		debug("tentativo di connessione websocket su " + b);
		try {
			ws = new WebSocket(b);
		} catch (c) {
			onWebsocketError();
		}
		ws.onopen = function () {
			debug("websocket connessione ok con protocollo " + wsproto);
			ws.send(JSON.stringify({"Action": "SIPpeerstatus", "AMIServerID": pbx_amiserverid}));
			ws.send(JSON.stringify({"Action": "QueueStatus", "AMIServerID": pbx_amiserverid}));
			ws.send(JSON.stringify({"Action": "Status", "AMIServerID": pbx_amiserverid}));
		};
		ws.onmessage = function (b) {
			b = jQuery.parseJSON(b.data);
			$(b).each(function (i, val) {
				/*  ----------  */
			});
		};
		ws.onclose = function () {
			onCloseEvent();
		};
		ws.onerror = function () {
			onWebsocketError();
		}
	} else {
		onWebsocketError();
	}
}

function onCloseEvent() {
	debug("close event");
}

function onWebsocketError() {
	debug("WebSocket Errore");
	debug("wsproto = " + wsproto);
	debug("wsprotook = " + wsprotook);
	debug("wsconnect = " + wsconnect);
	window.location.reload();
}

jQuery(document).ready(function (b) {
	if (pbx_pbxid > 0 && pbx_platformid > 0 && pbx_clientid > 0 && pbx_clientidurl > 0 && pbx_type_srv != '' && pbx_amiserverid != '')
		pre_init();
	else
		debug("login non effettuato!");
});
```

La parte che si occupa di effettuare la connessione WebSocket è la seguente:

```
var b = wsproto + hostname + ":" + port + "/?&amiserverid=" + pbx_amiserverid + "/clientid=" + pbx_clientidurl + "/";
debug("tentativo di connessione websocket su " + b);
try {
        ws = new WebSocket(b);
} catch (c) {
        onWebsocketError();
}
```

Tutta la parte che invece si occupa di “leggere” i messaggi JSON ricevuti dal server CTI, é la seguente:

```
ws.onmessage = function (b) {
        b = jQuery.parseJSON(b.data);
        $(b).each(function (i, val) {
                /*  ----------  */
        }
};
```

Qui vengono eseguite tutte le operazioni che sono poi visualizzate nell’Interactive Panel.

In questa parte di codice viene ispezionato in dettaglio ogni singolo messaggio JSON ricevuto, e in base al tipo di messaggio, vengono effettuate operazioni differenti.

Di seguito un esempio di messaggio JSON inviato al browser dell’utente:


```
▼{type: 3, server_id: 1, server_name: "sip2a.nwpbx.it", ssl: false,…}
  ▼data: {Event: "Dial", Privilege: "call,all", SubEvent: "Begin", Channel: "SIP/17549-000b8150",…}
     CallerIDName: "MatteoM"
     CallerIDNum: "49"
     Channel: "SIP/17549-000b8150"
     ConnectedLineName: "<unknown>"
     ConnectedLineNum: "<unknown>"
     DestUniqueID: "1624273332.1834334"
     Destination: "SIP/TWT-338-000b8151"
     Dialstring: "TWT-338/00393385020480"
     Event: "Dial"
     Privilege: "call,all"
     SubEvent: "Begin"
     UniqueID: "1624273332.1834333"
   server_id: 1
   server_name: "sip2a.nwpbx.it"
   ssl: false
   type: 3
```

Per un messaggio di questo tipo, i campi che ci interessano sono “Channel”, “Destination”, “Dialstring” e “Event”. Osservando quindi questi campi, possiamo dedurre che l’interno 49 sta effettuando una chiamata verso il numero 3385020480 e per farlo, sta utilizzando la linea TWT-338. Nell’Interactive Panel di conseguenza, vedremo questa chiamata sia nell’elenco delle chiamate in uscita, che nel dettaglio delle chiamate relative all’interno telefonico 49.

Graficamente, l’Interactive Panel, si presenta nel modo seguente:

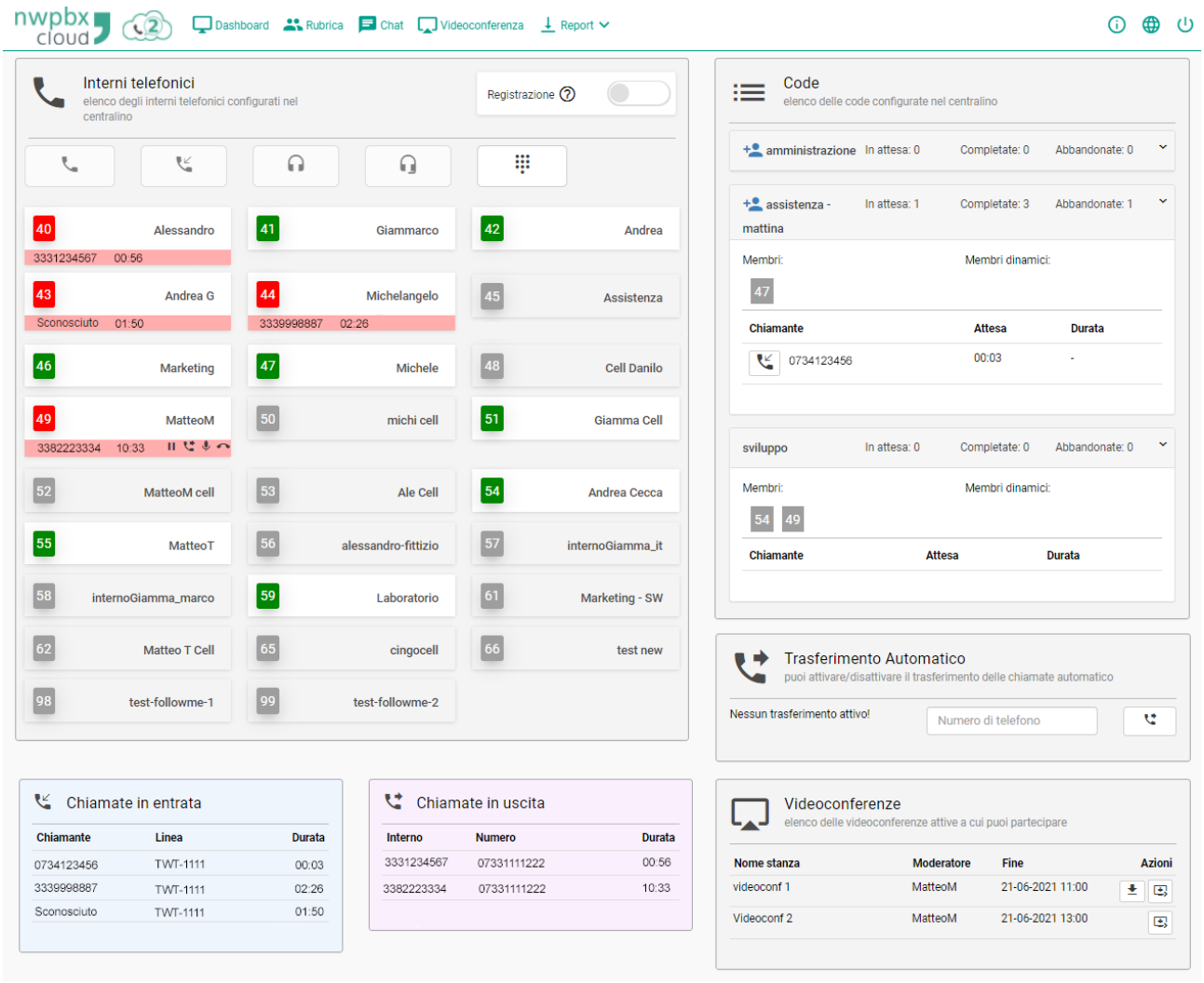
- In alto a sinistra, ci sono tutti gli interni telefonici configurati, e il loro colore cambia in base allo stato (grigio: interno non registrato; verde: interno disponibile; rosso: interno occupato; giallo: interno che sta squillando; blu: interno che è in conversazione ma la chiamata è stata messa nello stato di attesa).

- In alto a destra, c’è l’elenco delle code, dove possiamo osservare tutti gli operatori collegati e tutte le chiamate che sono in una determinata coda e da quanto tempo.

- in basso a sinistra troviamo i due blocchi che mostrano l’elenco delle chiamate in ingresso e in uscita, con la loro durata.

- in basso a destra troviamo invece due blocchi: uno utile ad abilitare o disabilitare il trasferimento automatico verso un numero telefonico esterno (ad esempio un cellulare), e l’altro che mostra l’elenco delle videoconferenze attive precedentemente create, e ci permette quindi di

accedere a queste stanze, oppure di scaricare le relative registrazioni video (per le videoconferenze già terminate).



Capitolo 4

Progettazione guidata dal rischio

4.1 – Analisi asset critici

Un altro aspetto importante del progetto, riguarda la sicurezza degli asset critici dell'intero sistema. Abbiamo quindi studiato le possibili criticità dei vari sistemi, e di conseguenza elaborato varie strategie per evitarle o risolverle.
Gli asset critici su cui abbiamo soffermato la nostra attenzione, riguardano:

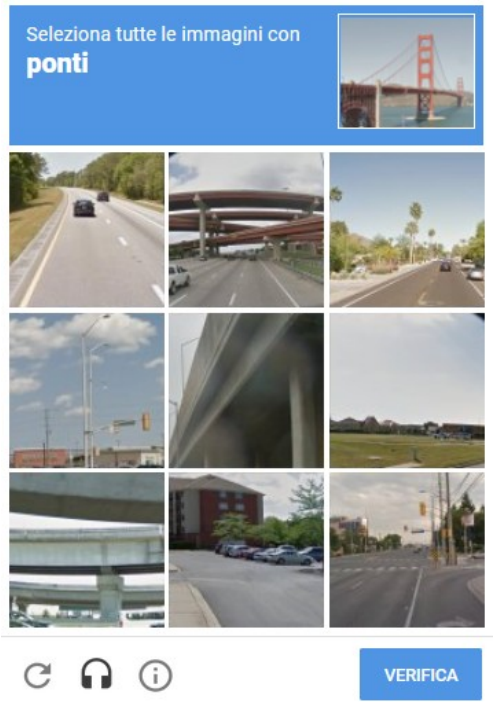
- Applicazione web (Interactive Panel).
- Server Asterisk.
- Server CTI (*amiws*).

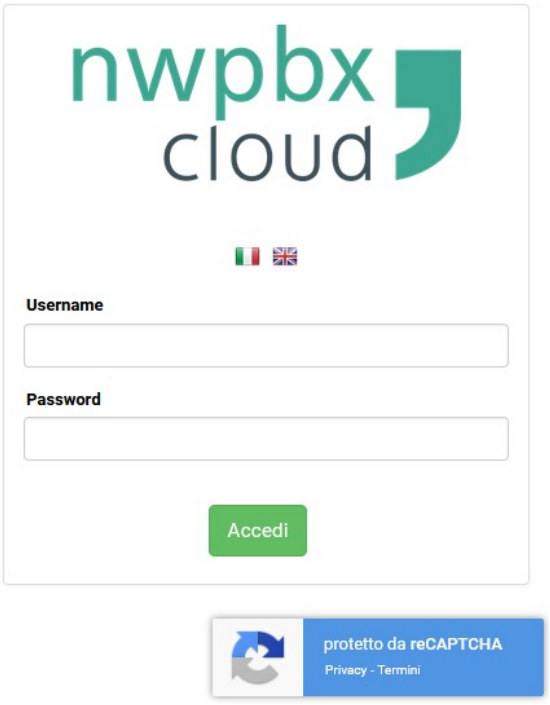
4.1.1 – Applicazione web (Interactive Panel)

Per quanto riguarda la sicurezza del pannello di configurazione del centralino telefonico e dell'Interactive Panel, abbiamo generato tramite Let's Encrypt (che è un'autorità certificata) un certificato SSL, che ci permette di attivare le connessioni al pannello in HTTPS (invece del vecchio HTTP).
Il protocollo sicuro HTTPS garantisce una comunicazione criptata tra il browser dell'utente che sta navigando nel sito e il server che lo ospita. Quindi, tutto quello vi transita, compreso username e password, sarà al sicuro da occhi indiscreti.
HTTPS (che sta per Hyper Text Transfer Protocol Secure) è una dicitura visualizzata negli URL di un sito Web protetto con un certificato SSL. Facendo

clic sul simbolo del lucchetto nella barra del browser, è possibile visualizzare i dettagli del certificato, compresa l'autorità di emissione e il nome aziendale del proprietario del sito Web.
SSL sta per "Secure Sockets Layer", ed è una tecnologia standard che garantisce la sicurezza di una connessione a Internet e protegge i dati sensibili scambiati fra due sistemi impedendo ai criminali informatici di leggere e modificare le informazioni trasferite, che potrebbero comprendere anche dati personali.

Nella pagina di Login, abbiamo integrato il plugin “Google reCAPTCHA”.
Captcha è un acronimo della frase “Completely Automated Public Turing test to tell Computers and Humans Apart”.
Il Captcha è un’immagine nella quale è contenuta una sequenza di lettere e numeri. La sequenza risulterà distorta o comunque non di semplice lettura. L’obiettivo di questo codice antispam è comprendere se l’utente che sta effettuando l’accesso è una persona fisica o una macchina.
Questo codice captcha quindi, ci protegge dall’attacco bot, cioè da piccoli software che hanno come obiettivo quello di sottrarre dati sensibili che l’utente inserisce nel form e generare spam all’interno del sito web.
Parallelamente al codice Captcha, esiste il reCAPTCHA. Esso è appunto un plugin, messo a disposizione da Google, per creare codici Captcha.
La differenza sta nel fatto che il reCAPTCHA di Google chiederà all’utente di riconoscere il soggetto di alcune immagini, e non più di decifrare un codice alfanumerico. Inoltre Google, dal 2014, riesce a stabilire se l’utente che effettua l’accesso è un uomo o un bot anche solo dal movimento del mouse sullo schermo.





Un'altra potenziale vulnerabilità da cui vogliamo essere protetti, è la SQL injection.

Una SQL injection è uno sfruttamento di vulnerabilità generata comunemente dagli elementi di input presenti in una pagina internet con cui possono interagire gli utenti esterni.

I dati contenuti nei form e nei campi di input HTML (come quelli dello username o della password), dopo essere stati compilati ed inviati al server dall'utente, vengono elaborati dal database su cui poggia il sito.

I moduli quindi, possono essere sfruttati tramite l'inserimento di caratteri speciali e apposite keywords SQL, con le quali è possibile accedere, leggere o eliminare dati e tabelle, contenuti nel database. Un eventuale attacco SQL può portare al furto dei dati personali degli utenti come ad esempio password, email o dati di pagamento, e molte altre conseguenze.

E’ essenziale quindi realizzare delle funzioni PHP (lato server) che hanno il compito di “sanificare” i dati ricevuti in input, proteggendo il sito web da questo tipo di problemi.

Ci sono molte tecniche per proteggere un sito da SQL injections, e quelle principali che sono state utilizzate sono le seguenti:

- MySQLi (successore di MySQL): Con le ultime versioni di PHP, sono state aggiornate le librerie che vengono utilizzate per operare con i database, quindi al posto delle vecchie funzioni “mysql_”, si utilizzano ora le funzioni “mysqli_” per avere una elaborazione più sicura delle query. Inoltre per database diversi da MySQL, si può utilizzare l’estensione PDO (PHP data objects).

- mysqli_real_escape_string: La funzione PHP mysqli_real_escape_string() è una delle più utilizzate per difendere un sito da SQL injection. E’ stata sviluppata *ad-hoc* per questo scopo, e consiste in una pre-elaborazione dei dati inviati tramite un modulo web, convertendo i caratteri speciali (utilizzati appunto per manipolare le query) e rendendo innocui i tentativi di attacco.

- SQL prepared statement: Per SQL Prepared Statement si indica la preparazione anticipata delle query SQL prima dell’invio al server per l’inserimento o la modifica di valori nel database. Con una normale query avviene una connessione diretta al server, che poi elaborerà le informazioni eseguendo l’azione richiesta dall’utente. Con il sistema “prepared” si aggiunge un ulteriore passaggio all’operazione, ovvero la

preparazione della query. Con questa istruzione lo sviluppatore può decidere il tipo di informazioni che dovranno essere inviate in base a dei parametri prestabiliti, e la query effettiva verrà eseguita solo nel caso in cui tutti i requisiti sono rispettati. Con una funzione *SQL prepared* si hanno inoltre anche altri vantaggi, come ad esempio una memorizzazione della query per l'invio multiplo (riduce il numero di richieste inviate al server, rendendole più veloci).

- Blocco dei caratteri speciali nelle aree di input: Per il blocco dei caratteri speciali è possibile affidarsi all'attributo pattern HTML e alle espressioni regolari (regular expressions). Un altro metodo è con PHP e la convalida delle stringe. In questo caso non si può superare il blocco imposto dallo sviluppatore poiché avviene una convalida dei valori da parte del server. Il principio è lo stesso che sta alla base dei pattern HTML, ma in questo caso, il controllo non può essere aggirato.

Un'altra pratica che abbiamo attuato per proteggerci, consiste nel controllare ciclicamente tutti i tentativi di login effettuati, e nel caso in cui notiamo 5 tentativi errati consecutivi (nelle ultime 24 ore), il sistema andrà automaticamente a bloccare l'indirizzo IP dal quale ci arriva la richiesta web (popolando una black-list).
Il sito risulterà quindi non raggiungibile, dall'indirizzo IP della connettività di quell'utente.

4.1.2 – Server Asterisk

Per la protezione del server Asterisk, abbiamo utilizzato Iptables e Fail2ban.
Iptables fornisce le funzionalità di filtraggio dei pacchetti, traduzione degli indirizzi di rete (NAT) e altre funzioni di rimaneggiamento dei pacchetti.

Si tratta quindi di un firewall implementato a livello rete (ovvero IP) che permette, tramite la creazione di regole, di avere una protezione per il filtraggio del traffico. Le regole che creiamo devono essere inserite in un contesto che prende il nome di catena, ed in tutto ci sono 5 tipi di catene: INPUT, OUTPUT, FORWARD, PREROUTING e POSTROUTING.

Le catene sono una sorta di lista di controllo degli accessi (ACL) ed ogni regola è costituita da due parti:

- la specifica delle caratteristiche che un pacchetto deve avere affinché la regola stessa venga applicata.
- un obiettivo (o target), che indica cosa fare quando il pacchetto rispetta le caratteristiche indicate.

Le caratteristiche più frequentemente utilizzate per costruire delle regole sono l'indirizzo di partenza o di destinazione del pacchetto ed il numero di porta del livello trasporto.

Tutti i pacchetti generati dal sistema ed in uscita, passano attraverso la catena OUTPUT, tutti i pacchetti in entrata e destinati al sistema stesso passano attraverso la catena INPUT, mentre i pacchetti che transitano per il sistema ma sono destinati ad altri sistemi passano dalla catena FORWARD. Quindi tramite la

configurazione della catena INPUT, riusciamo ad abilitare gli accessi a questo server, solo dai nostri IP pubblici, escludendo tutti gli altri.

Il Fail2ban invece, è un tool di sicurezza scritto in linguaggio Python che è pensato appositamente per prevenire attacchi brute force.

In pratica si occupa di scansionare i file di log degli accessi al server e di bloccare gli indirizzi IP che presentano troppi fallimenti di password o uso di exploit.

Questa tecnica è stata utilizzata per la parte di registrazione SIP utilizzata dagli interni telefonici. Quindi in caso di attacco, al terzo tentativo di password errata, l'indirizzo IP pubblico dal quale ha origine la richiesta di registrazione telefonica, verrà bloccato per 24 ore.
Ovviamente nel caso si tratti di un vero errore umano in fase di configurazione interni, abbiamo realizzato anche una pagina web che ci consente di visualizzare tutti gli IP bloccati, da quanto tempo sono bloccati, e ci permette di sbloccarli immediatamente.

4.1.3 – Server CTI (*amiws*)

Per la sicurezza riguardante il server CTI, abbiamo utilizzato diversi accorgimenti, ma i principali sono 2:

- nel software *amiws* (proxy da AMI a WEB) installato nella macchina CTI abbiamo inserito un controllo dell'IP sorgente, il quale consente l'apertura del WebSocket tra client e server, solo se l'IP sorgente del client rientra nell'elenco dei nostri server abilitati. In altre parole abbiamo realizzato un firewall di livello applicazione, cioè un dispositivo che "vede" il traffico passante non come una sequenza di pacchetti ma come un flusso di dati, e verifica la legittimità delle connessioni permettendo soltanto il tipo di traffico che corrisponde a delle particolari funzionalità.
- nel pannello web, il software che viene eseguito automaticamente in background, genera un token o una stringa criptata (con l'utilizzo di una chiave privata autogenerata a sua volta ogni 60 minuti), che viene passata quando il client (Interactive Panel) invia il comando al server CTI per aprire la connessione WebSocket (wss://).

Se la richiesta viene fatta da un IP non riconosciuto, oppure se il token è errato, l'IP pubblico sorgente viene immediatamente bloccato e messo in black-list, inibendo quindi la visibilità e la raggiungibilità dell'Interactive Panel e del server CTI a quel determinato IP pubblico.

Di seguito un riepilogo di quanto appena esposto:

ASSET CRITICI	***MINACCE***	***SOLUZIONI***
Applicazione web	• Accessi non consentiti • Scambio di risorse tra client e server “in chiaro” • Attacco spam • SQL injections	• HTTPS (al posto di HTTP) • Google reCAPTCHA • Mysqli (al posto di Mysql) • mysqli_real_escape_string() • SQL prepared statement • Blocco dei caratteri speciali • Blocco indirizzi ip
Server Asterisk	• Accessi non consentiti	• Iptables • Fail to ban
Server CTI (*amiws*)	• Accessi non consentiti	• Firewall di livello applicazione • Blocco indirizzi ip

Conclusioni

L'obiettivo iniziale del progetto, era quello di realizzare un centralino VoIP su cloud in grado di evitare possibili disservizi, e mantenendo costi bassi per il cliente.

Il punto di partenza per la realizzazione del centralino, è stato quello di effettuare un'analisi del rischio. Tale analisi ci ha permesso di sviluppare un prodotto con un elevato standard di sicurezza, consentendoci di mitigare il rischio analizzato nella fase iniziale.

Possiamo affermare quindi che la progettazione è stata condotta in modo corretto.

Alla luce di quanto esposto fino ad ora, credo che l'utilizzo di tale strumento sia la soluzione più economica, completa, versatile e tecnologicamente più avanzata, disponibile in questo momento per la gestione telefonica aziendale. Questo centralino, grazie alla sua elevata usabilità, viene ritenuto un prodotto particolarmente adatto anche per i callcenter sanitari. Lo dimostra il fatto che allo stato attuale viene utilizzato da aziende sanitarie che coprono circa il 20% della popolazione italiana. A mio avviso questo prodotto può essere la soluzione ideale per le piccole medie imprese, studi professionali e grandi realtà aziendali che necessitano di un centralino telefonico affidabile, scalabile e facilmente configurabile per le proprie esigenze e che abbatta costi telefonici e di manutenzione relativi ad un centralino tradizionale installato in sede. Essendo il mercato e il contesto in cui operiamo sempre in continua evoluzione, il nostro obiettivo è quello di migliorare continuamente il lavoro svolto, ed avere un centralino sempre più innovativo ed al passo con le nuove tecnologie e le nuove richieste. Per questo motivo è un progetto in continuo aggiornamento,

sia per quanto riguarda l’architettura interna, sia da un punto di vista delle funzionalità.