# SAFETY HOME

Proteggi la tua abitazione domestica.

Gruppo numero 4 : Franciscone Luca 20010057, Uejdis Dushi 20010884

https://youtu.be/WL9n2D0VLUE

# SAFETY HOME

Proteggi la tua abitazione domestica.

## *Architettura del progetto e componenti utilizzati*

- Dispositivo IOT : Aeon Lab MultiSensor 5 Gen
- Dispositivo IOT: Everspring AN-145 Holder Lampadina
- Dispositivo IOT: Everspring AN-158 Prese Elettrica
- Dispositivo IOT: Everspring SM-103 Door/Windows Detector
- Rete Z-WAVE : RaZberry (Z-Wave)
- Webcam integrata nel portatile
- Server REST : implementato utilizzando SPARK in locale sul proprio pc
- Assistente conversazione : API.ai
- Servizi API :  Telegram
- Servizi API : Google Calendar
- Software per offrire il servizio del server su indirizzo pubblico : NGROK

# Obiettivo

Il progetto nasce con l'idea di sfruttare la tecnologia IOT unita a servizi API per controllare la sicurezza della propria abitazione domestica.

L'utente ha la possibilità di interagire con l'ambiente domestico, effettuare del sensing e interazione tra i dispositivi presenti nell'abitazione mediante agenti conversazionali, monitorandolo ed attivando delle funzionalità avanzate:

- Monitoraggio dell'ambiente domestico e segnalazione di intrusione mediante un messaggio su Telegram;
- Monitoraggio dell'ambiente domestico ed invio di immagine se viene rilevata un'intrusione;
- Attivazione automatica del servizio di monitoraggio dell'ambiente quando viene trovato un impegno dell'utente annotato su Google Calendar (si suppone quindi che l'utente essendo occupato si trovi all'esterno di quella che è la propria abitazione domestica);
- Possibilità di simulare la popolazione dell'ambiente domestico mediante l'accensione di luci e di una radio.

# *Descrizione della realizzazione*

## Assistenti Conversazionali

L'assistente conversazionale scelto per comunicare con il nostro server REST è API.ai, disponibile all'indirizzo: https://api.ai/

Per facilitare le interazioni sono state definite un insieme di intent ognuno legato con una possibile iterazione con l'ambiente domestico:

- **Accendi/spegni la luce** (utilizzato per simulare la presenza di persone e per non degradare la qualità delle foto scattate con la webcam)
- **Accendi/spegni la presa** (utilizzato per simulare la presenza di persone collegando ad essa una radio)
- **Monitoraggio: con immagine**
- **Monitoraggio: senza immagine**
- **Verifica presenza** (Rileva la presenza di una persona all'interno dell'abitazione)
- **Stato porta** (in un dato istante mediante sensore di apertura porta)
- **Scatta foto** ( permette di scattare una foto la quale verrà caricata su Dropbox e inviata all'utente mediante un messaggio su Telegram)
- **Attiva/Disattiva simula presenza**

L'assistente conversazionale di API.ai invia chiamate POST all'indirizzo URL settato all'interno del suo campo fulfillment, l'indirizzo sul quale esporre il servizio è stato realizzato mediante l'utilizzo di ngrok, software che ha la funzionalità di rendere pubblico il server REST realizzato localmente per poter ricevere POST da API.ai mappando mediante un servizio di tunnelling, il comando utilizzato per settare ngrok è stato "ngrok http 4567 " .

Il server cattura le chiamate post ricevute, ne preleva il json contenuto all'interno del body della request con la rispettiva classe di appartenenza e demanda la logica al metodo doWebhook il quale analizza la action e si occupa della realizzazione dell'interazione, del settaggio dell'output che verrà codificato in json e restituito al chiamante.

## Telegram

Per inviare comandi ad API.ai ed avviare degli intent è stato utilizzato il client di messaggistica Telegram; inizialmente è stato creato un Bot mediante BothFather chiamato UpoSafetyHomeBot che si occupa dell'interazione con l'utilizzatore; una volta creato e restituito il token è stato settato all'interno di API.ai per attivare il servizio di Webhook, successivamente è stato abilitato il campo fulfillment all'interno di tutti gli intent che ne richiedono di eseguire delle action.

Il servizio di API.ai ha un timeout di 4 secondi, una volta terminato automaticamente fa scadere la connessione con il server REST e restituisce la risposta settata di default per un dato intent.

Per ovviare a questo problema abbiamo deciso di sfruttare le chiamate REST offerte dal servizio di Telegram per poter far comunicare il server con il bot. Prima di tutto è stato disattivato temporaneamente il servizio collegato al Webhook di API.ai mediante la chiamata GET all'indirizzo https://api.telegram.org/bot**TOKEN/deleteWebHook** con l'utilizzo del software POSTMAN. Successivamente è stato inviato un messaggio da parte di ogni singolo utente presente nel chatbot ed è stato catturato il Json contenente gli ultimi messaggi non visualizzati mediante una chiamata GET https://api.telegram.org/botTOKEN/getUpdates **,** nel body di essa è stato recuperato il json contenente il chatID dei singoli utenti che verrà successivamente utilizzato per l'invio dei messaggi. Dopo aver recuperato i singoli chatID è stato riattivato il servizio di webhook per Telegram per permetterne nuovamente l'utilizzo con API.ai.

Dovendo utilizzare un servizio di aggiornamento costante sono quindi stati creati dei metodi appositi per permette di effettuare delle chiamate POST all'indirizzo del bot: https://api.telegram.org/bot**423930159:AAF3ES_GcBxl5HmrV5HdfF137_XCfLXc1ZU/send Message**" contenenti nel body un json così strutturato:

{ "text" : "testoConvertitoInUTF8" , "chat-id" : chatIdDell'utente }

- **sendMessage**: metodo che si occupa dell'invio del messaggio, prende in input un messaggio da inviare, un chatId e la Url di telegram verso la quale effettuare la POST, esegue il metodo per la conversione del messaggio in UTF8, il risultato viene inserito

in una stringa che poi verrà convertita come json e successivamente passata al metodo eseguiPost.
- **convertToUtf**: metodo che prende in input una stringa e la converte in UTF-8
- **eseguiPost**: metodo che prende in input un url e un json e si occupa di settare il metodo della richiesta in tipo "POST" e settare il Content-Type col tipo "application/json" successivamente ne esegue la POST.

## Google Calendar

Per permettere l'attivazione automatica del servizio di monitoraggio dell'abitazione domestica quando l'utente è al di fuori di essa abbiamo sfruttato il servizio di API di Google Calendar per verificare se in un dato momento è presente o meno un impegno sul calendario dell'utente, supponendo che se l'utente crea un evento sia occupato al di fuori della mura domestiche.

Per abilitare il servizio è stato creato un progetto pubblico con id safetyhome-173418 all'interno della console developers di Google all'indirizzo https://console.developers.google.com , successivamente sotto la voce Credenziali è stato generato un Json contenente le Informazioni per l'autenticazione OAuth2, più precisamente l' "ID client"e il "Client secret" i quali sono stati inseriti all'interno della cartella "resources" all'interno del file "client_secrets.json".

E' stata utilizzata la classe QuickStart per creare il file delle credenziali all'interno della cartella " C:\Users\Luca Franciscone\.credentials\calendar-java-quickstart" contenenti il certificato OAuth2.0.

E' stato reso pubblico il calendario con id 20010057@studenti.uniupo.it per permetterne la lettura degli eventi dall'esterno ed è stata creata la classe **UtilCalendario** che ha il compito di scaricare il calendario precedente creando una lista di eventi. Mediante il metodo **attivazioneServizio** viene verificato se il momento dell'attivazione del servizio coincide con un evento, in tal caso si avvia il servizio di sorveglianza. Per realizzare questo servizio sono state utilizzate le seguenti guide:
https://developers.google.com/google-apps/calendar/quickstart/java

https://developers.google.com/google-apps/calendar/v3/reference/

## Webhook

Il servizio di cattura delle chiamate POST è stato realizzato all'interno della classe **SafetyHomeWebHook** utilizzando il server messo a disposizione dal docente. All'inizio della classe sono state inserite una serie di variabili statiche con lo scopo di sostituire tutti i magic number e le stringhe ripetute all'interno del codice.

Una volta avviato il **main** della classe viene scelto se far partire il server o il servizio di sorveglianza che si appoggia alle API di Google Calendar che ha la durata passata come parametro ( nel nostro caso un periodo di test di 10 secondi) . Il server ha il compito di rimane in ascolto di chiamate GET e POST. Le POST provenienti dal servizio di API.ai vengono quindi catturate, viene recuperato il Json contenuto al loro interno, viene chiamato il metodo **doWebhook** (che si occupa dell'analisi della action contenuta e di implementare la parte logica , su necessità demandarla a un metodo ) , infine viene preparata la risposta da restituire sia al chiamante sia al servizio di Telegram mediante l'utilizzo del metodo **sendMessage** presente all'interno della classe **Util**.

## Webcam

Per poter integrare l'utilizzo del servizio di monitoraggio con immagine abbiamo utilizzato le "Webcam Capture API" utilizzando la webcam integrata direttamente all'interno del nostro server REST.

La nostra webcam in base alla action lanciata potrà quindi interagire con i nostri dispositivi IOT attivandosi e attivandoli in base alle condizioni ambientali.

Per la realizzazione dell'integrazione abbiamo seguito la seguente guida : http://webcam-capture.sarxos.pl/

## Interazione tra i dispositivi

- Dispositivo IOT : Aeon Lab MultiSensor 5 Gen
- Dispositivo IOT: Everspring AN-145 Holder Lampadina
- Dispositivo IOT: Everspring AN-158 Prese Elettrica
- Dispositivo IOT: Everspring SM-103 Door/Windows Detector
- Webcam

Per il controllo ed il sensing dell'ambiente domestico sono stati utilizzati differenti dispositivi in base alla funzionalità da attivare, di seguito farò una breve descrizione dei devices utilizzati sulla base del servizio richiesto

- **Accendi/spegni luce**: metodo che si occupa dell'accensione e dello spegnimento della luce all'interno dell'abitazione, per la realizzazione è stato utilizzato un Everspring AN-145 Holder Lampadina.

- **Accendi/spegni presa**: metodo che si occupa della gestione dell'accensione del device Everspring AN-158 Prese Elettrica, usato per collegare una radio o una televisione per simulare la presenza di persone all'interno delle mura

- **Scatta Foto** : metodo che quando viene invocato si occupa di scattare una foto dell'ambiente domestico, caricarla su Dropbox ed inviarne il link tramite Telegram. Il metodo utilizza il sensore di luminosità per avere sempre una qualità alta dell'immagine anche in presenza di bassa luminosità ambientale; quando la luminosità ambientale sarà minore di 200 lux verrà accesa luce all'interno dell'ambiente. I devices utilizzati sono : webcam, Everspring AN-145 Holder Lampadina, Aeon Lab MultiSensor 5 Gen.

- **Monitoraggio: con immagine** : metodo che si occupa di gestire il servizio di monitoraggio con immagine. Dato che API.ai va in timeout dopo 4 secondi è stato pensato di ovviare il problema lanciando un thread chiamato "ThreadSorveglianzaConImmagine" che continui a girare in background per permettere la sorveglianza dell'abitazione. Durante la creazione viene specificato il

quantitativo di tempo per il quale il thread deve rimanere attivo; se durante l'inizializzazione trova tutti i devices necessari all'azione richiesta si avvia, in caso contrario invia un messaggio di notifica a Telegram specificando il tipo di errore incontrato. Il thread se rileva che il sensore di apertura della porta è su stato "on" o se il sensore di presenza è "triggered" sa che probabilmente qualcuno è presente all'interno dell'abitazione domestica, quindi ne verifica la luminosità ambientale ( se è < di 200lux ) accende una luce per permettere una maggiore qualità di ripresa della cam che si occuperà di scattare la foto. Una volta scattata la foto verrà posizionata in una cartella interna alla Dropbox e il link verrà restituito mediante Telegram all'utente, per poter visionare se la persona all'interno del locale è di sua conoscenza o si tratta di un intruso. I devices utilizzati per questa interazione sono : Aeon Lab MultiSensor 5 Gen, Everspring AN-145 Holder Lampadina , Everspring SM-103 Door/Windows Detector, webcam.

- **Monitoraggio: senza immagine** : metodo che si occupa di gestire il servizio di monitoraggio. Viene lanciato un thread in background "ThreadSorveglianzaSenzaImmagine" che si occupa di gestire la sorveglianza per un determinato numero di secondi impostato nel costruttore. La presenza viene rilevata controllando i valori forniti dal sensore di apertura porta e dal sensore di movimento, se uno dei due restituisce un risultato positivo viene inviato un messaggio su Telegram di "Rilevata presenza". I dispositivi utilizzati in questa iterazione sono : Everspring SM-103 Door/Windows Detector, Aeon Lab MultiSensor 5 Gen, Everspring.

- **Verifica presenza**: il metodo si occupa di eseguire un'interrogazione sul dispositivo di rilevazione di presenza e restituirne il valore catturato inviando l'esito dell'interrogazione sul Bot. Device utilizzato: Aeon Lab MultiSensor 5 Gen.

- **Stato porta**: il metodo si occupa di eseguire un'interrogazione sul sensore di apertura porta/finestre restituendone il valore inviandolo tramite un messaggio su Telegram. Device utilizzato : Everspring SM-103 Door/Windows Detector.

- **Calendario: Monitoraggio con/senza immagine**: il metodo si occupa di controllare se sono presenti degli eventi in un dato momento mediante l'ausilio della classe **UtilCalendario** attivando se rileva eventi in corso il servizio di monitoraggio con/senza immagine ( quindi si suppone che se è presente un evento in corso l'utilizzatore si trovi all'esterno delle proprie mura domestiche ) . La classe **UtilCalendario** si occupa di richiedere tutti gli eventi presenti all'interno del calendario 20010057@studenti.uniupo.it verificando se uno degli eventi coincide con l'ora attuale di sistema.

Esistono due tipologie principali di eventi:
  - Eventi con un inizio e termine all'interno della giornata
  - Eventi che occupano l'intera giornata

  Abbiamo cercato di gestire entrambe le categorie degli eventi, ognuna con un formato differente di rappresentazione della data mediante un metodo interno chiamato **attivazioneDelServizioInterno**, una volta interrogato l'evento per capire a quale categoria appartiene viene parsificato il contenuto del suo Json e confrontato con la data attuale, se l'evento è compreso tra l'inizio e la fine dell'evento sul calendario viene quindi restituito un valore true che verrà utilizzato dai metodi del livello superiore per attivare il servizio di videosorveglianza.
- Devices utilizzati: Everspring SM-103 Door/Windows Detector, Aeon Lab MultiSensor 5 Gen, Everspring.

- **Attiva/disattiva simula presenza**: il metodo si occupa di simulare la presenza di persone all'interno del locale mediante l'accensione della luce e di una presa pilotata alla quale collegheremo una radio o un dispositivo che genera rumore acustico per simulare le normali attività giornaliere quali guardare la tv oppure ascoltare la radio. I devices utilizzati per l'iterazione sono : Everspring AN-145 Holder Lampadina, Everspring AN-158 Prese Elettrica.

Il link del video di presentazione è accessibile al seguente indirizzo:

https://youtu.be/WL9n2D0VLUE