FIND NEIGHBOUR

INTRODUZIONE

Questo software è stato realizzato utilizzando Visual Studio Code e il linguaggio di programmazione Python. È parte integrante del progetto “Neighborhood Delivery” creato per l’esame di IoT & 3D Intelligent Systems dal gruppo Kaido, Merolla e Calvano.

Il software viene eseguito su EC2, gestita dal sistema di Amazon AWS ed è uno degli script che compongono la parte Cloud del progetto. È collegato al database MySQL presente sulla stessa macchina e comunica con lo script del bot di Telegram e con il Bridge. È in ascolto sulla porta 8001.

FUNZIONAMENTO

Lo script Python è un server http e ha l’obiettivo di trovare un vicino disponibile al reperimento del pacco in consegna qualora il proprietario del suddetto non fosse in casa.

Il software rimane in attesa di una chiamata da parte di un Bridge, che chiede se è disponibile un vicino, inviando i dati necessari per trovarlo. Quando arriva il messaggio, il codice accede al database per trovare le informazioni del proprietario del pacco (il cui Bridge ha inviato la richiesta) e quelle di un vicino in grado di ricevere la merce. Le informazioni vengono inviate come risposta http al Bridge ed inoltre vengono fornite allo script responsabile dell’invio dei messaggi tramite Telegram.

In caso di errori durante l’esecuzione della ricerca, oppure della mancata individuazione del vicino, il Bridge viene notificato con opportuna risposta.

CODICE E FUNZIONI PRINCIPALI

Il codice, scritto in Python, è scomposto in diverse funzioni ognuna con lo scopo di portare a termine uno dei compiti di cui è incaricato.

La parte principale è data dalla classe requestHttpReceiver che stabilisce come rispondere alle chiamate GET del Bridge.

Qui di seguito sono elencate le funzioni principali del software.

* ***def run()***

Viene istanziato il server http e si rimane in attesa delle richieste esterne.

* ***def do\_GET()***

Definito all’interno della classe requestHttpReceiver ed è la funzione richiamata quando un Bridge fa una richiesta GET tramite protocollo http. Genera la risposta chiamando funzioni esterne e gestendo eventuali errori.

* ***def get\_neighbour(id)***

Richiamata da do\_GET utilizza il codice (id) fornito dal Bridge per reperire le informazioni del proprietario e per cercare nel suo vicinato una persona disponibile alla ricezione.

* ***def send\_message\_bot(data\_prop, data\_neig)***

Invia le informazioni reperite dal database precendentemente (creando un dizionario) verso lo script incaricato di messaggiare proprietario del pacco e vicino ricevente mediante protocollo http, metodo POST.

* ***def AI\_help(params, data\_prop)***

Quando non è disponibile un vicino, tenta di trovarne uno utilizzando il modulo “AI-ML”. Esegue una richiesta http GET verso di esso e ottiene il risultato richiesto, controllando eventuali errori.

DIPENDENZE

Il codice Python utilizza alcune librerie esterne per svolgere i suoi compiti.

1. **pymysql**, per comunicare con il database e reperire le informazioni.
2. **requests**, per inviare la richiesta POST allo script del bot Telegram.
3. **http.server**, per istanziare il server http in ascolto.
4. **http.client**, per eseguire le chiamate al server http “AI-ML”