




ANNO ACCADEMICO 2018/2019

# **Relazione Elaborato**

Internet of Things Based Smart System

DAVIDE DIGENTI – 055000372  
GABRIELE RUIZ – 055000351





Sommario

Obiettivi dell’elaborato ..... 2

Architettura..... 4

Scelte Implementative ..... 5

Caso D’Uso ..... 6

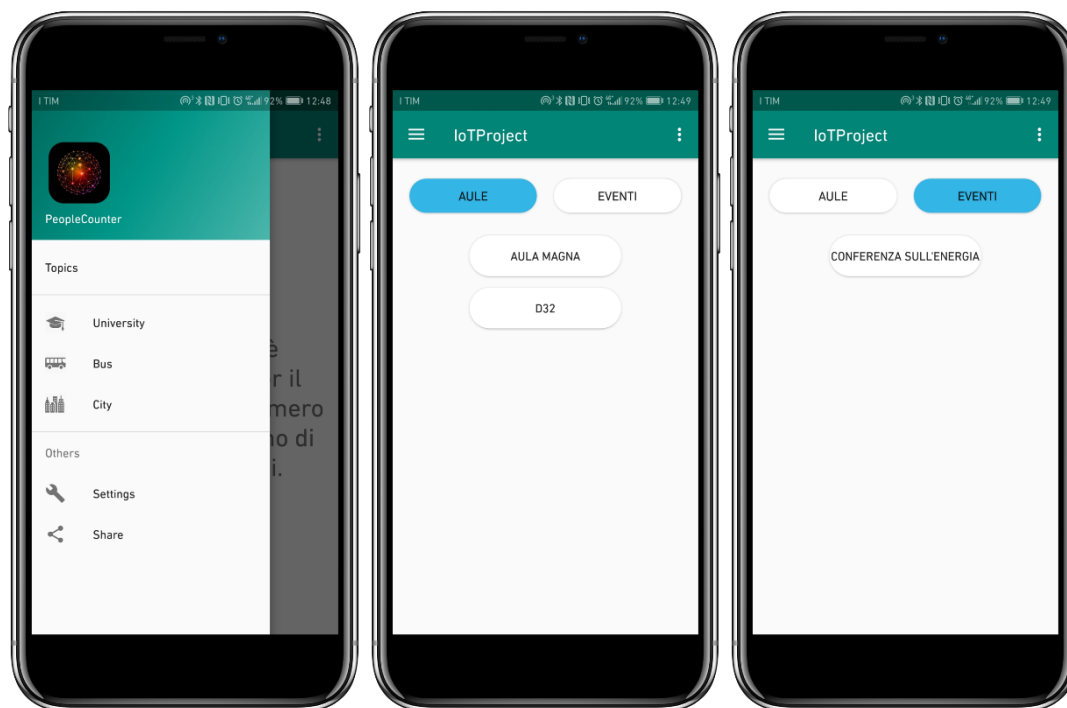
# Obiettivi dell'elaborato

Gli obiettivi dell'elaborato riguardano il monitoraggio del numero di persone all'interno di una qualunque struttura. Abbiamo realizzato ciò sfruttando un algoritmo di computer vision in grado di riconoscere le persone dall'alto. Questo ha permesso di avere una maggiore precisione rispetto all'uso di sensori di prossimità o tecnologie analoghe, in quanto il riconoscimento tramite computer vision riesce a rilevare anche i casi di attraversamento contemporaneo da parte di più persone.

Ciò viene realizzato attraverso il posizionamento di una telecamera posta sopra un qualunque ingresso/uscita, collegata ad un Arduino. Non abbiamo necessità di memorizzare le immagini video provenienti dalla telecamera in quanto l'elaborazione riguardante il numero di persone all'interno della struttura viene effettuata interamente dall'Arduino, il quale si occuperà di pubblicare esclusivamente i dati riguardanti il numero di persone presenti, sfruttando il protocollo MQTT. Questo approccio garantisce la privacy delle persone in quanto nessuna immagine verrà memorizzata né tantomeno inviata in rete.

Per rendere disponibili i dati raccolti abbiamo realizzato un'applicazione Android in cui abbiamo immaginato di fornire questo tipo di servizio per tre diversi scenari:

- Università
- Autobus
- Eventi cittadini



Per quanto riguarda l'università, abbiamo implementato un servizio di monitoraggio delle aule in grado di dare informazioni puntuali riguardanti il numero di presenze, ed un servizio che fornire risultati statistici per quanto riguarda le conferenze. Offriamo quindi la possibilità all'utente di scegliere se consultare i dati relativi a uno specifico luogo, oppure esaminare direttamente gli eventi concernenti quello scenario. Questo genere di informazioni può essere utile agli organizzatori di un evento (come ad esempio una conferenza) sia per avere un riscontro sul numero di partecipanti, sia per avere informazioni sul grado di interesse del pubblico. Un'altra possibile applicazione potrebbe essere il monitoraggio di un'aula durante lo svolgimento di una lezione: in tal caso, nell'eventualità in cui si noti una propensione degli studenti ad arrivare in ritardo, il professore potrà decidere di posticipare l'inizio della lezione di qualche minuto in modo da venire incontro alle esigenze degli studenti.

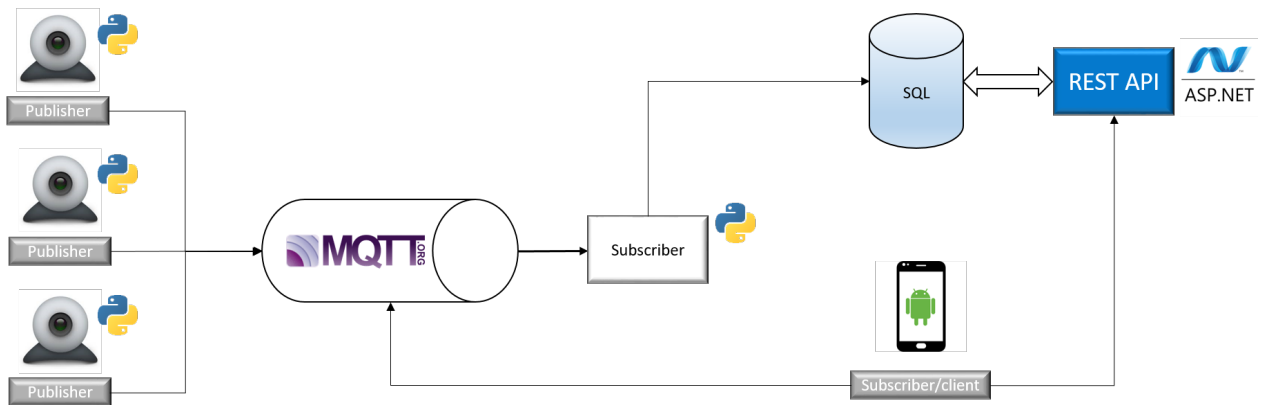


Per quanto riguarda gli eventi cittadini, esattamente come accade nello scenario universitario, prevediamo di fornire informazioni sia in tempo reale, riguardanti il numero di persone presenti in un determinato luogo pubblico (a patto che in esso siano presenti ingressi e uscite ben definite e delimitate) che informazioni statistiche inerenti al numero di persone che hanno partecipato ad un determinato evento.

Per gli autobus abbiamo immaginato di monitorare esclusivamente il numero di persone all'interno degli stessi, che può essere utile per un duplice scopo: l'utente sarà in grado di sapere se un determinato autobus è affollato o meno, e il gestore dell'autobus, sulla base di questi dati potrà prendere decisioni riguardanti l'organizzazione delle linee urbane.



# Architettura



L'architettura proposta è composta da diverse entità: le due principali sono Publisher e Subscriber, che comunicano in maniera indiretta utilizzando MQTT.

Il ruolo di Publisher può essere interpretato da una piattaforma hardware (es. Arduino) al quale è collegato una webcam, che verrà posta nel punto più alto della porta. Il Publisher, tramite un algoritmo di Computer Vision tratterà quindi una linea in corrispondenza della soglia della porta, e rileverà le persone che la attraverseranno, riconoscendo se queste stanno entrando od uscendo.

Ogni qualvolta una persona attraversa la soglia, il Publisher pubblica il dato relativo al numero di persone presenti all'interno della struttura su una coda MQTT utilizzando uno specifico *topic*, che identifica il contesto in cui esso sta operando (es. /University/Aula Magna oppure /City/Piazza Duomo).

Il Subscriber (in Python) preleva i dati relativi a tutti i topic presenti nella coda e si occuperà di scriverli su un database SQL.

I dati salvati sul database possono essere utilizzati per tracciare uno storico relativo ad ogni evento, in modo da poter essere consultato successivamente. Questi dati vengono esposti tramite REST API da un WebServer.

L'applicazione Android svolge il duplice ruolo di Subscriber diretto alla coda MQTT per il monitoraggio real-time dei dati pubblicati, e di Client per le REST API.

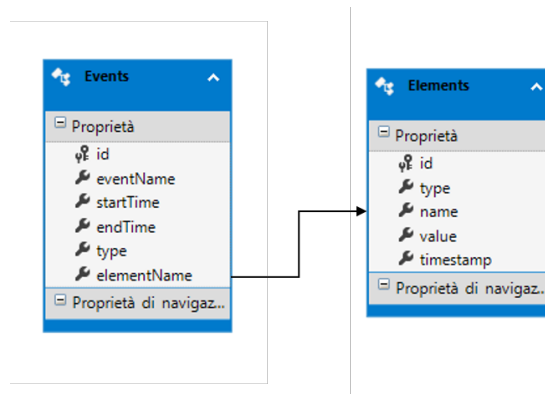
# Scelte Implementative

Per il seguente elaborato abbiamo realizzato delle applicazioni in Python, ASP.NET e Android.

Per quanto riguarda il Publisher, si è scelto un approccio conservativo per quanto riguarda la pubblicazione dei dati, per cui questa avviene solo nel momento in cui c'è un cambiamento nella somma totale delle persone presenti ad un determinato evento. Questo per evitare che il database venga popolato da dati ridondanti che, oltre a non aggiungere contenuto informativo, ridurrebbero le performance.

Si è scelto di implementare un unico Subscriber che è sottoscritto a tutti i Topic presenti nella coda, in modo da renderlo generico e garantirne il funzionamento anche in seguito all'aggiunta di nuovi Topic.

Le REST API sono state sviluppate in ASP.NET MVC, sfruttando il modello Entity-Framework, che ha permesso di modellare in maniera immediata la struttura del database e le classi che si mappano sugli elementi che esso contiene.



All'interno del database sono presenti due tabelle:

la tabella Elements modella il dato salvato dal Subscriber:

- ❖ Type – descrive il contesto da cui proviene il dato (University, Bus, City)
- ❖ Name – descrive il nome dell'elemento (Aula Magna, BRT, Piazza Duomo)
- ❖ Value – rappresenta il numero di persone presenti in un determinato istante di tempo
- ❖ Timestamp – rappresenta l'istante di tempo considerato

la tabella Events modella l'evento di cui si vuole tenere traccia:

- ❖ eventName – descrive il nome dell'evento
- ❖ StartTime – rappresenta l'istante di tempo in cui l'evento inizia
- ❖ EndTime – rappresenta l'istante di tempo in cui l'evento termina
- ❖ Type – descrive il contesto da cui proviene il dato (University, Bus, City)
- ❖ ElementName – descrive il nome dell'elemento (Aula Magna, BRT, Piazza Duomo)

Sia il database che il WebServer sono stati deployati sulla piattaforma Azure in modo da avere un endpoint sempre disponibile per l'applicazione Android.

**N.B.**

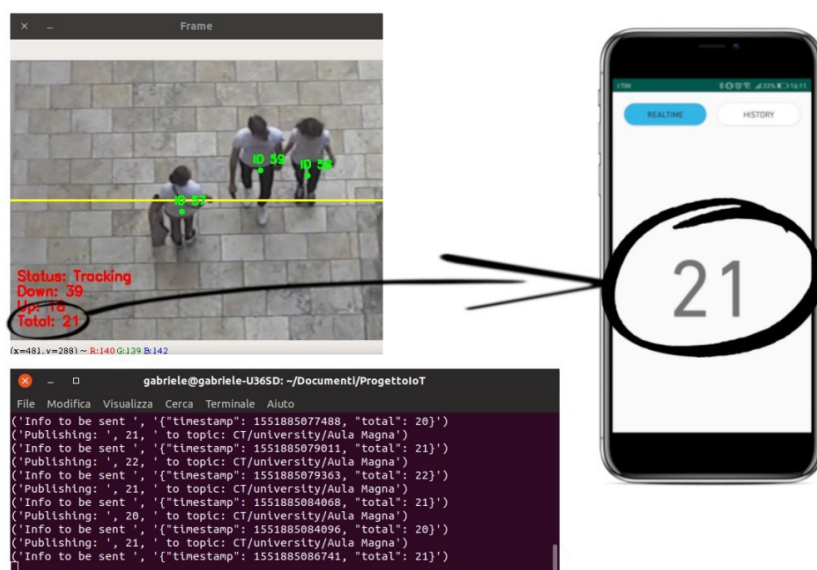
*Non sono state implementate API per l'inserimento di Eventi.*

# Caso D'Uso

Consideriamo il caso in cui vogliamo monitorare l'affluenza relativa ad una conferenza che si tiene in Aula Magna.

Nell'immagine vediamo il funzionamento dell'algoritmo di Computer Vision, il quale tiene traccia dell'ingresso e dell'uscita delle persone, e la pubblicazione di questa informazione tramite protocollo MQTT.

È mostrato inoltre il funzionamento dell'applicazione Android in modalità Subscriber, la quale ha una visione real-time delle persone presenti all'interno della struttura.



Per effettuare la simulazione abbiamo utilizzato un video che riprende le persone dall'alto, che riproduce in maniera soddisfacente le reali condizioni per cui il software è stato progettato.

Abbiamo scelto di utilizzare un video per la simulazione in virtù delle difficoltà riguardanti il posizionamento della webcam per un corretto funzionamento dell'algoritmo di Computer Vision.