* Introduzione:
  + Descrizione progetto per visualizzazione statistiche bar
* Pattern Pub/sub
  + Spiegazione generale
  + Differenze con Polling e altri pattern
  + Server Side Events
  + MQTT
  + Redis
* Analisi iniziale progetto:
  + Analisi dei requisiti
  + Descrizione principali componenti e linguaggi utilizzati (PC, Raspberry/Arduino e Python)
  + Descrizione e scelta protocolli: Server-sent events, MQTT, Redis
* Implementazione:
  + Spiegazione fasi di sviluppo e relativo codice documentato
  + Test effettuati
* Risultato e conclusioni:
  + Riflessioni sul risultato ottenuto
  + Eventuali migliorie ed implementazioni future

# Introduzione

Con il seguente elaborato, descrivo l’utilizzo di pattern architetturale publish/subscribe usando il protocolli MQTT e Server Side Events per la comunicazione asincrona tra sistemi eterogenei all’interno della rete. In particolare vado a descriverne il funzionamento tramite un progetto realizzato con lo scopo di visualizzare le statistiche di vendita di un bar sia con un’ interfaccia web-based, che con un Arduino che comanda un device IoT.

# Analisi iniziale

Il progetto è formato da più componenti hardware e software.

Partendo dal database usato dal software gestionale, che è un instanza di PostgreSQL 11 sulla quale, non avendo pieno accesso, mi sono dovuto limitare a leggere i dati in polling ogni tot secondi.

L’applicazione invece si divide in una parte di backend scritta in Python usando il framework Flask e una minima parte di visualizzazione frontend composta da semplici pagine HTML con aggiunta di bootstrap per curare la parte grafica.   
Il backend ha il compito di leggere i dati da database usando la libreria psycopg2, elaborarli e avviare una comunicazione SSE per renderli accessibili al frontend, che a sua volta, usando la libreria Chart.js li visualizza in grafici.  
Altro compito del backend è quello di rendere i dati accessibili anche tramite protocollo MQTT perché, in questo caso, c’è un Esp32 in ascolto che riceve i dati e li visualizza.

Poi l’applicazione è stata racchiusa in un container, usando docker, per permettere l’avvio combinato con l’istanza locale di Redis. L’uso di Redis è fondamentale per l’utilizzo dei Server Side Events che, essendo asincroni, hanno bisogno di un pool di dati sempre presente e pronto per essere letto (a differenza del polling su database). Inoltre mi ha permesso di rendere più flessibile l’accesso ai dati elaborati, permettendo di avere sempre disponibili i valori già formattati in base al tipo di visualizzazione scelta lato frontend.

### Scelta dei protocolli

Nel seguente paragrafo, andrò ad analizzare i protocolli utilizzati facendone il confronto con le loro controparti maggiormente in uso.

Partendo dal protocollo SSE