|  |  |
| --- | --- |
| **GlobalAzure 2021** | Abstract  **#CloudGen #GlobalAzure2021**  4Solid S.r.L. |

Index

[Definizione di IIoT 2](#_Toc69028487)

[API REST 4](#_Toc69028488)

[REST 4](#_Toc69028489)

# Definizione di IIoT

L’IIoT è costituito da apparecchiature e piattaforme di analisi avanzate connesse a Internet che elaborano i dati prodotti. I dispositivi IIoT spaziano da piccoli sensori ambientali a robot industriali complessi. Nonostante la parola “*industriale*” possa far pensare a magazzini, cantieri navali e fabbriche, le tecnologie IIoT si annunciano molto promettenti per un’ampia gamma di settori, tra cui l’agricoltura, la sanità, i servizi finanziari, la vendita al dettaglio e la pubblicità.

L’IIoT (Industrial Internet of Things) è una sottocategoria dell’IoT. Il termine si riferisce alle tecnologie IoT applicate all’ambiene industriale, ovvero nelle infrastrutture del settore manifatturiero. L’IIoT è una tecnologia chiave chiave nell’industria 4.0, la nuova fase della rivoluzione industriale.

Lo scopo dell’industria 4.0 è quello di enfatizzare le tecnologie intelligenti, i dati, l’automazione, l’interconnetività, l’intelligenza artificiale e numerose altre tecnologie e funzionalità. Queste tecnologie stanno rivoluzionando il modo in cui le fabbriche e le organizzazioni industriali funzionano.

E’ possibile integrare sensori intelligenti su macchinari produttivi, sistemi energetici e infrastrutture come tubazioni e cablaggi. Tali sensori, grazie ai dati che raccolgono, e le funzioni avanzate che offrono, possono aiutare le attività industriali a migliorare in efficienza, produttività, sicurezza del personale e molto altro.

L’IoT industriale migliora la comunicazione macchina-macchina. Fornisce ai gestori degli impianti dati e con essi una panoramica chiara di come sta funzionando il loro impianto. Attraverso la continua raccolta di dati granulari, le aziende industriali possono meglio monitorare l’energia, l’acqua e le altre risorse utilizzate quando le macchine sono in funzione e come esse stanno producendo. Gli operatori possono poi apportare modifiche manuali o gli impianti possono regolarsi automaticamente per ottimizzare le proprie operazioni.

Azure IoT Hub

Consente la comunicazione a sicurezza e affidabilità elevate tra l’applicazione IoT ed i dispositivi gestiti. IoT Hub di Azure offre un back-end per soluzioni ospitate sul cloud per connettere praticamente qualsiasi dispositivo.

Offre un canale di comunicazione con sicurezza migliorata per l’invio e la ricezione di dati da dispositivi IoT.

Permette la distribuzione OTA (Over-the-Air) degli aggiornamenti per mantenere i dispositivi IoT aggiornati e sicuri.

Integrazione completa con Azure EventGrid e funzioni serverless per semplificare lo sviluppo di applicazioni IoT

Compatibilità con Azure IoT Edge e Azure Stack per la creazione di applicazioni IoT ibride

An **IoT Edge** is a smart device equipped with compute power and designed to execute services in the field. By processing data at their origin, a lot of transmission and network traffic can be avoided. Thus both costs and risks can be minimized.

A **device twin** is the digital representation of your connected device. It’s a JSON file in the cloud managed by IoT Hub, and there is one of them for each of your IoT devices connected to IoT Hub.

# API REST

Un'API REST, nota anche come API RESTful, è un'interfaccia di programmazione delle applicazioni conforme ai vincoli dell'architettura REST. REST è l'acronimo di REpresentational State Transfer.

Un'[interfaccia di programmazione delle applicazioni (API)](https://www.redhat.com/it/topics/api) è un set di definizioni e protocolli per la compilazione e l'integrazione di software applicativi. Può essere considerata come un contratto tra un fornitore di informazioni e l'utente destinatario di tali dati: l'API stabilisce il contenuto richiesto dal consumatore (la chiamata) e il contenuto richiesto dal produttore (la risposta). Ne è un esempio l'API di un servizio meteorologico, in cui l'utente invia una richiesta contenente un codice postale al quale il produttore fornisce una risposta in due parti, dove la prima indica la temperatura massima e la seconda la minima.

Se, in altre parole, si desidera interagire con un computer o un sistema per recuperare informazioni o eseguire una funzione, un'API facilita la comunicazione con il sistema che può così comprendere e soddisfare la richiesta.

L'API funge quindi da elemento di intermediazione tra gli utenti o i clienti e le risorse che questi intendono ottenere. È anche un mezzo con il quale un'organizzazione può condividere risorse e informazioni assicurando al contempo sicurezza e controllo, poiché stabilisce i criteri di accesso.

L'utilizzo dell'API inoltre non impone all'utente di conoscere le specifiche con cui le risorse vengono recuperate o la loro provenienza.

## REST

REST è un insieme di principi architetturali, non un protocollo né uno standard. Chi sviluppa API può implementare i principi REST in diversi modi.

Quando una richiesta viene inviata tramite un'API RESTful, questa trasferisce al richiedente uno stato rappresentativo della risorsa. L'informazione, o rappresentazione, viene consegnata in uno dei diversi formati tramite HTTP: JSON (Javascript Object Notation), HTML, XLT o testo semplice. Il formato JSON è uno dei più diffusi, perché indipendente dal linguaggio e facilmente leggibile da persone e macchine.

Affinché un'API sia considerata RESTful, deve rispettare i criteri indicati di seguito.

* Un'architettura client-server composta da client, server e risorse, con richieste gestite tramite HTTP.
* Una comunicazione client-server [stateless](https://www.redhat.com/it/topics/cloud-native-apps/stateful-vs-stateless), che quindi non prevede la memorizzazione delle informazioni del client tra le richieste; ogni richiesta è distinta e non connessa.
* Dati memorizzabili nella cache che ottimizzano le interazioni client-server.
* Un'interfaccia uniforme per i componenti, in modo che le informazioni vengano trasferite in una forma standard. Ciò impone che:
  + le risorse richieste siano identificabili e separate dalle rappresentazioni inviate al client;
  + le risorse possano essere manipolate dal client tramite la rappresentazione che ricevono poiché questa contiene le informazioni sufficienti alla manipolazione;
  + i messaggi autodescrittivi restituiti a un client contengano le informazioni necessarie per descrivere come il client deve elaborare l'informazione;
  + le informazioni siano ipermediali, ovvero accedendo alla risorsa il client deve poter individuare, attraverso hyperlink, tutte le altre azioni disponibili al momento.
* Un sistema su più livelli che organizza ogni tipo di server (ad esempio quelli responsabili della sicurezza, del bilanciamento del carico, ecc.) che si occupa di recuperare le informazioni richieste in gerarchie, invisibile al client.
* Codice on demand (facoltativo): la capacità di inviare codice eseguibile dal server al client quando richiesto, estendendo le funzioni del client.

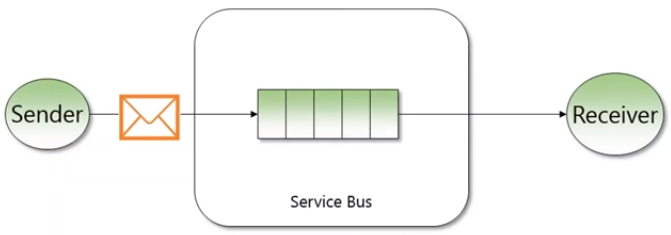
Sebbene l'API REST debba essere conforme a questi criteri, il suo impiego è comunque considerato più semplice rispetto a quello di un protocollo prescrittivo come SOAP (Simple Object Access Protocol), che presenta requisiti specifici come la messaggistica XML e la conformità integrata di sicurezza e transazioni, che lo rendono più lento e pesante.

Al contrario, REST è un insieme di linee guida applicabili quando necessario, il che rende le API REST più rapide e leggere, ottimali quindi per l'[Internet of Things (IoT)](https://www.redhat.com/it/topics/internet-of-things) e [lo sviluppo di app mobili](https://www.redhat.com/it/topics/mobile).

# Azure Service Bus

## Azure Service Bus Queues

[Azure Service Bus Queues](https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions) follow a very simple messaging protocol – a message sender delivers a message to a Queue that will be residing in a Service Bus Namespace, which would then be received by a receiver in initial or later stage. Queues follow asynchronous communication model where messages tend to go only in one direction. The messages in the Queue are processed via FIFO (First-In-First-Out) pattern, which means the first message delivered into the Queue will be the first one to be retrieved by the receiver. One of the biggest advantages of Azure Service Bus Queues is that a message stays in a queue as long as the receiver picks it up.



## Azure Service Bus Topics

Topics are almost similar to Queues, but it varies on a slight factor where Topics can have **multiple, independent subscriptions**. A subscriber to a Topic can receive a copy of each message sent to that Topic. Multiple receivers could be allocated to a subscription. A Topic can have zero or more subscriptions, but a subscription can only belong to one Topic. If you do not want all subscribers to read all the messages, then you can apply rules to specify additional actions to filter or to modify message properties.

