16 APRILE 2021

VIRTUAL EVENT

#GLOBALAZURE ITALY 2021

BY COMMUNITY • FOR COMMUNITY

IOT IN SALSA SERVERLESS

MASSIMO BONANNI

@MASSIMOBONANNI

SPONSOR

managed/designs

il partner tecnologico per chi ha idee ambiziose. Innovazione pratica da 15 anni.

Microsoft

empower every person and every organization on the planet to achieve more.

Swag and more

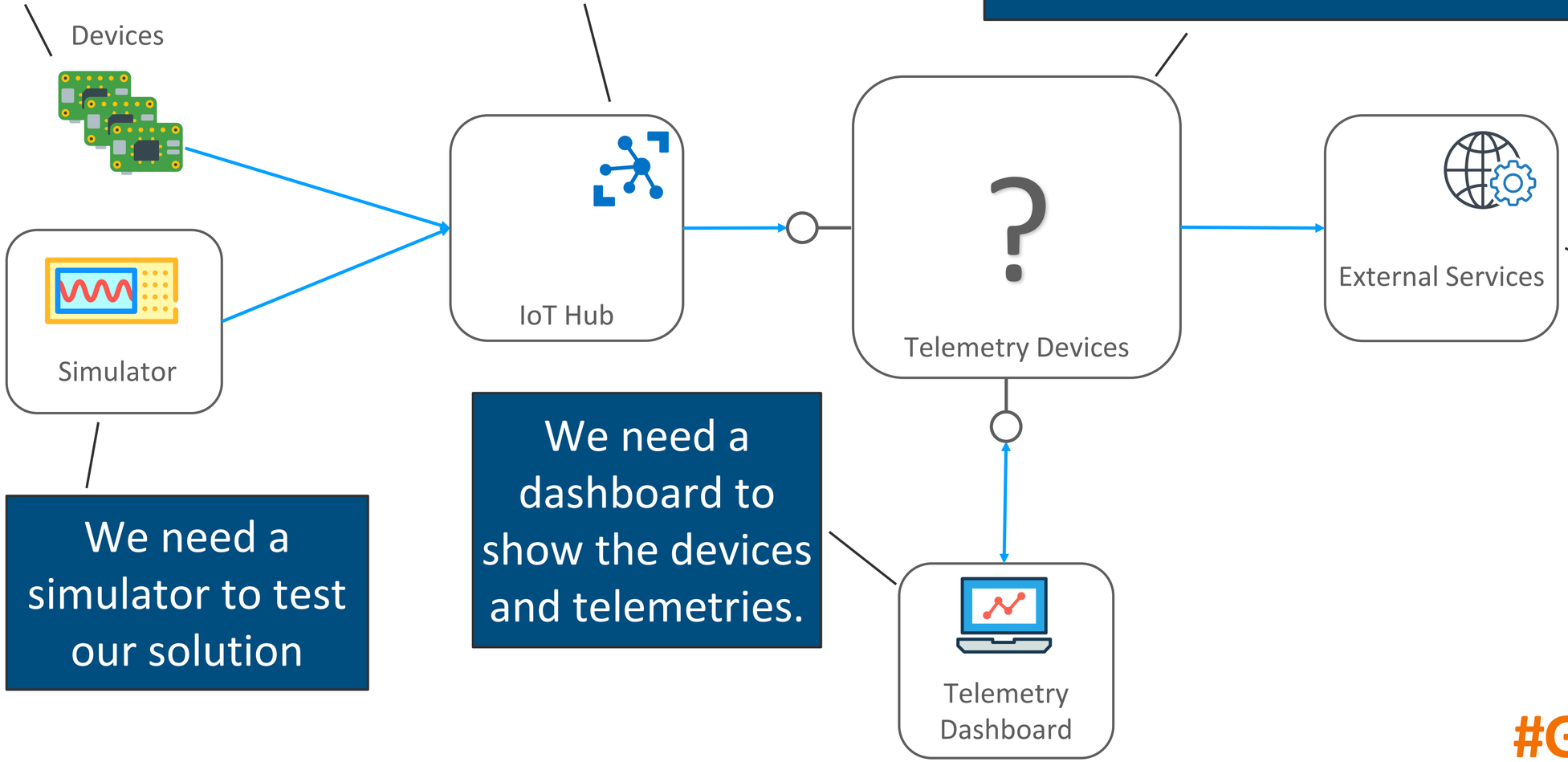
- Claim your attendee Learner Badge here:
- 30 Days to learn it: aka.ms/global-azure/30D2L
- Virtual background and ANOTHER Badge: blog.globalazure.net/Swag

A SIMPLE IOT SCENARIO.

Every single device produces telemetries about temperature and humidity

IoTHub ingests telemetry in Azure

We want to
- elaborate the telemetries
- store the last m minutes
- provide APIs to retrieve the telemetries
- having many types of devices with different logic



We need to interact with external services to provide alerts or notifications

We need a simulator to test our solution

We need a dashboard to show the devices and telemetries.

#GLOBALAZURE

We need:

- Event-driven scalability
- State management abstraction
- Asynchronous interaction
- Different types of devices

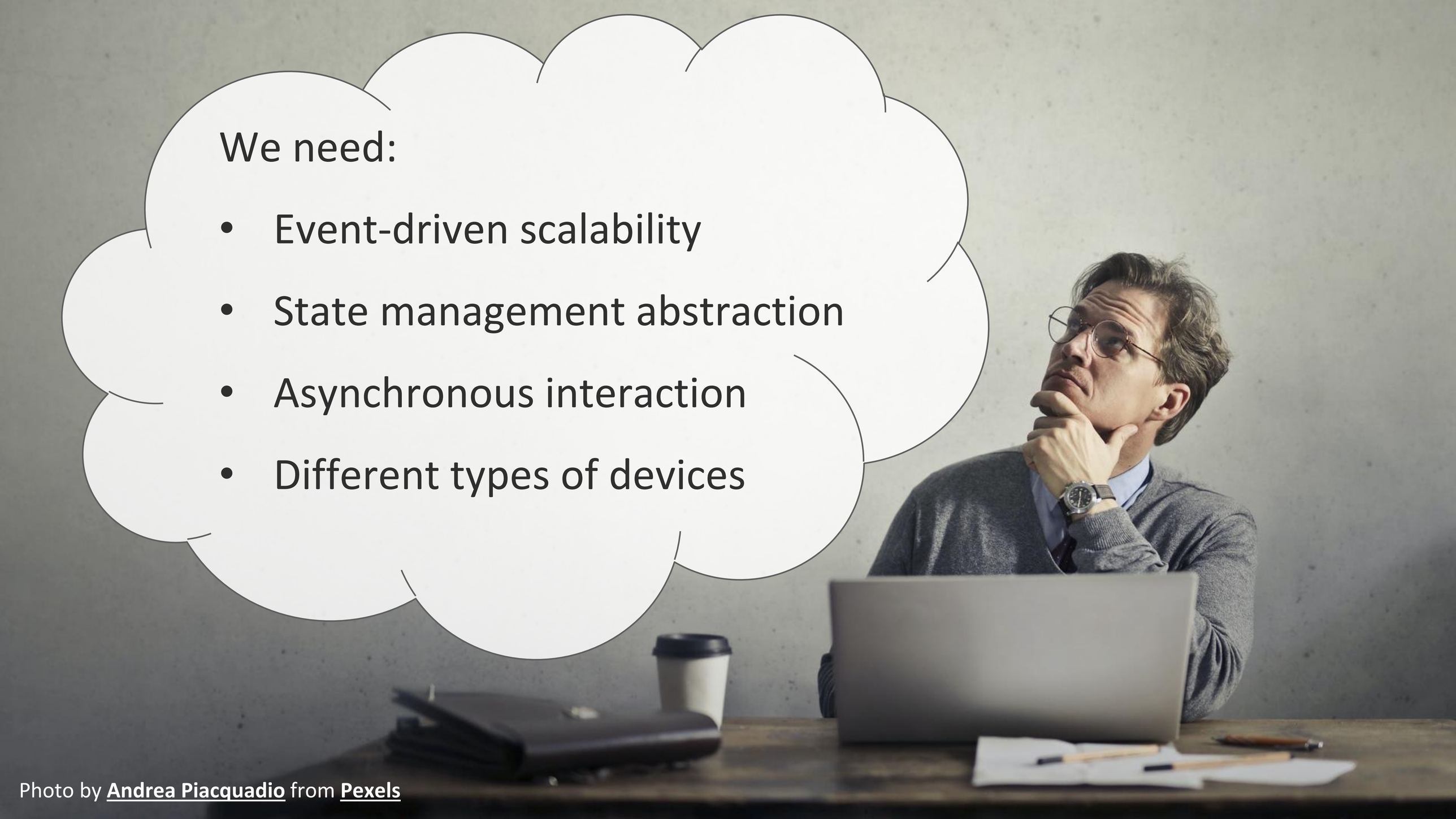


Photo by **Andrea Piacquadio** from **Pexels**

Photo by **Andrea Piacquadio** from **Pexels**

WHAT ARE DURABLE FUNCTIONS?

Azure Functions Extension	Durable Task Framework	Languages
Based on Azure Functions Adds new Triggers and Bindings Manages state, checkpoints, and restarts	Long running persistent workflows in C# Used within various teams at Microsoft to reliably orchestrate long running operations	C# JavaScript F# Python Powershell (few patterns)

COMPONENTS OF DURABLE FUNCTIONS

Client

It is the triggered functions that will create new instances of an orchestration.

It is the entry point for creating an instance of a durable orchestration

→ async →

Orchestrator

It is the heart of a durable function.

Orchestrator functions describe the way and order actions are executed.

→ async →

Activity

It is the basic unit of work in a durable orchestration.

An activity function must be triggered by an activity trigger.

DURABLE ENTITIES AKA ENTITY FUNCTIONS

Based on Durable Functions Functions with special trigger

Entity Functions define operations for reading and updating small piece of state (serialized in a storage table)

Entity Functions are accessed using:

- Entity Name
- Entity Key

Entity Functions operations can be accessed using:

- Entity Key
- Operation Name
- Operation Input
- Scheduled time

ENTITY ID

AN ENTITY ID IS SIMPLY A PAIR OF STRINGS THAT UNIQUELY IDENTIFIES AN ENTITY INSTANCE.

Entity Name	Entity Key
It is a name that identifies the type of the entity. This name must match the name of the entity function that implements the entity. It isn't sensitive to case	It is a string that uniquely identifies the entity among all other entities of the same name

DEFINE THE ENTITY

FUNCTION-BASED SYNTAX

Entities are represented as functions and operations are explicitly executed in the function body.

This syntax works well for entities with simple state, few operations, or a dynamic set of operations like in application frameworks.

This syntax doesn't catch type errors at compile time

```
[FunctionName("Counter")]
public static void Counter([EntityTrigger] IDurableEntityContext ctx)
{
    switch (ctx.OperationName.ToLowerInvariant())
    {
        case "add":
            ctx.SetState(ctx.GetState<int>() + ctx.GetInput<int>());
            break;
        case "reset":
            ctx.SetState(0);
            break;
        case "get":
            ctx.Return(ctx.GetState<int>());
            break;
    }
}
```

DEFINE THE ENTITY

CLASS-BASED SYNTAX

Entities are represented by classes.

This syntax produces more easily readable code and allows operations to be invoked in a type-safe way.

Function-based and Class –based variants can be used interchangeably in the same application.

```
[JsonObject(MemberSerialization.OptIn)]
public class Counter
{
    [JsonProperty("value")]
    public int CurrentValue { get; set; }

    public void Add(int amount) => this.CurrentValue += amount;

    public void Reset() => this.CurrentValue = 0;

    public int Get() => this.CurrentValue;

    [FunctionName(nameof(Counter))]
    public static Task Run([EntityTrigger] IDurableEntityContext ctx)
        => ctx.DispatchAsync<Counter>();
}
```

DEFINE THE ENTITY

CLASS-BASED SYNTAX

The class must be constructible

The class must be JSON-serializable

Operations must have at most one argument, and not have any overloads or generic type arguments.

Arguments and return values must be serializable values or objects.

You can define an interface for the entity

WHAT AN OPERATION CAN DO

- Reads or updates entity state
- Makes an external I/O operation
- Uses context to interact with other entities
- Uses context to start an orchestration

DEFINE AN INTERFACE FOR AN ENTITY

- Entity interfaces must only define methods.
- Entity interfaces must not contain generic parameters.
- Entity interface methods must not have more than one parameter.
- Entity interface methods must return void, Task, or Task<T>

```
public interface IDeviceEntity
{
    void SetConfiguration(DeviceEntityConfiguration config);
    void TelemetryReceived(DeviceTelemetry telemetry);
}
```

ACCESSING THE ENTITIES

Calling

(round-trip)

Two-way (**round-trip**) communication.

You send an operation message to the entity, and then wait for the response message before you continue.

→ **Orchestrator**

Signaling

(fire-and-forget)

One-way (**fire and forget**) communication.

You send an operation message but don't wait for a response.

→ **Orchestrator**
Client
Entity

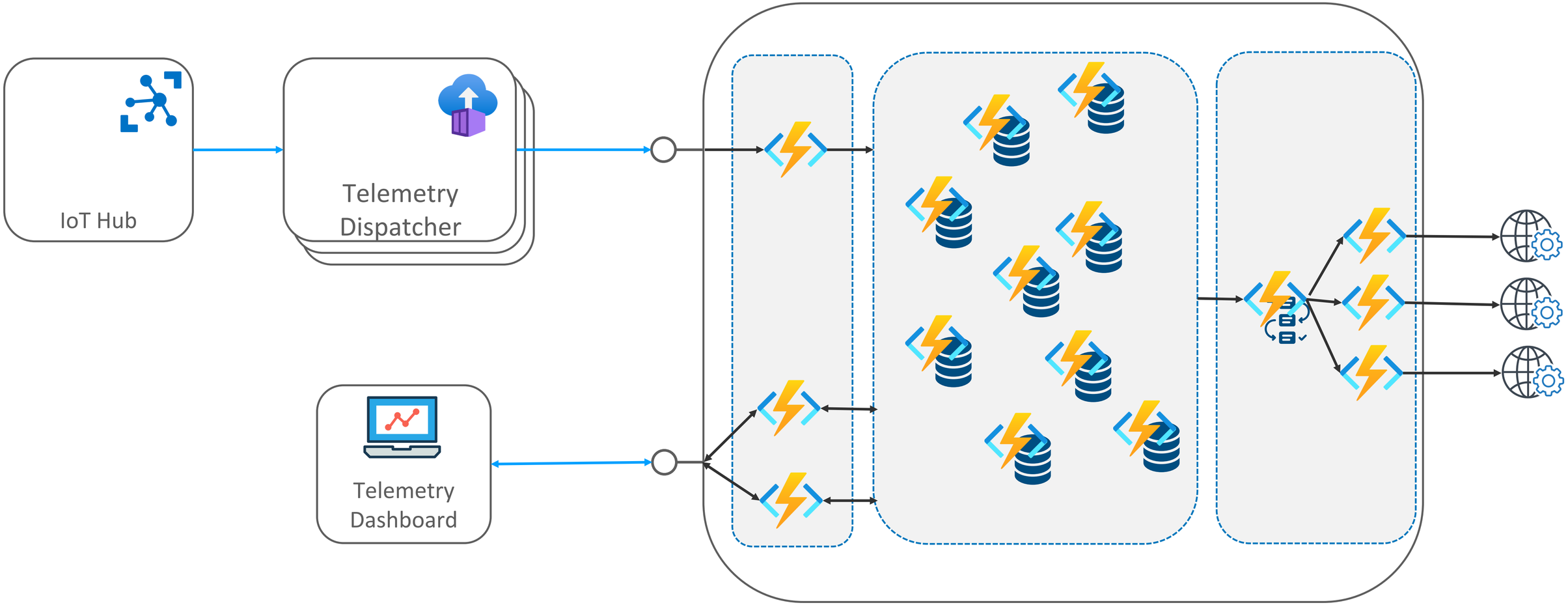
State

Two-way communication.

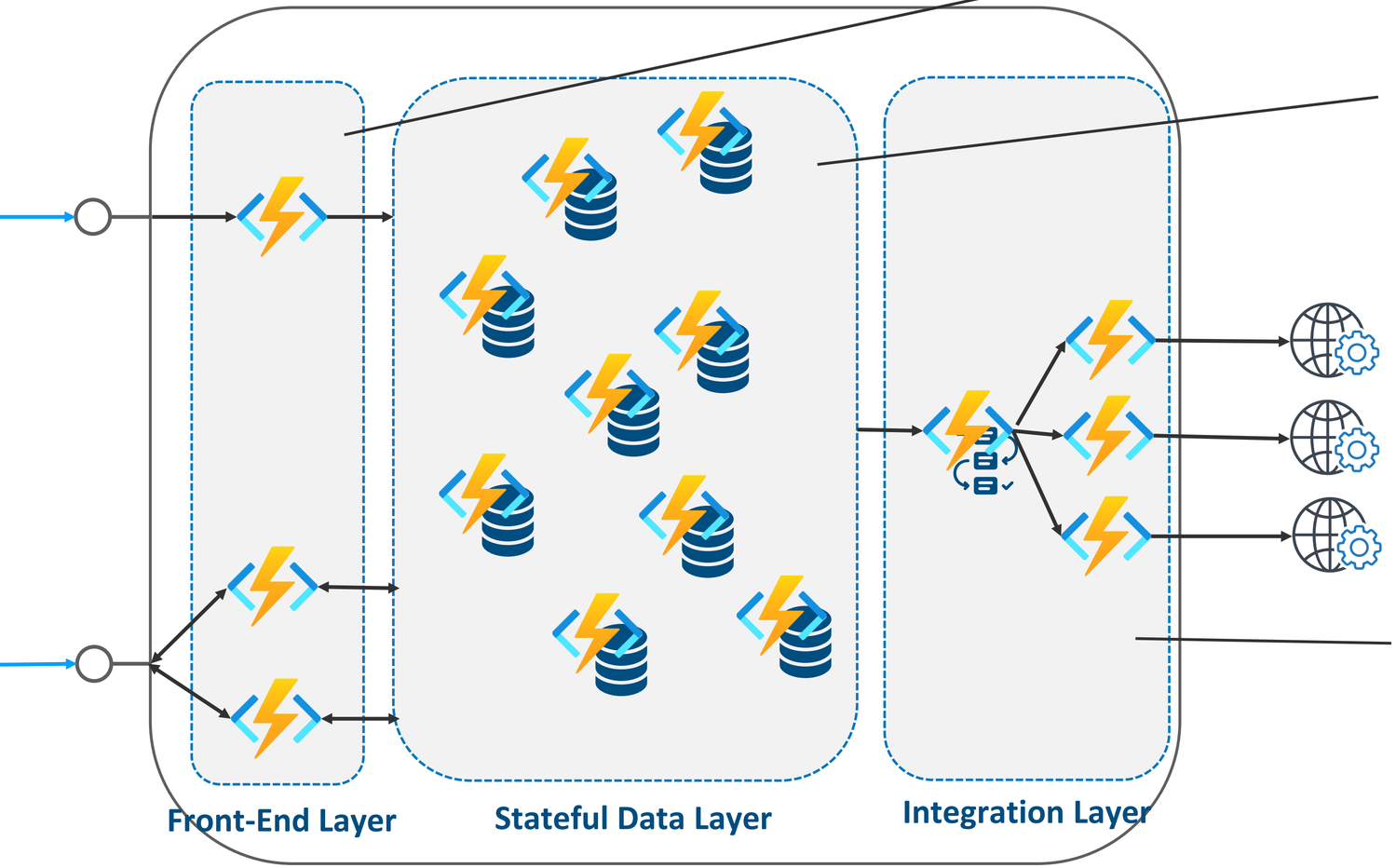
You can retrieve the state of an entity

→ **Client**

A POSSIBLE SOLUTION….



A POSSIBLE SOLUTION....

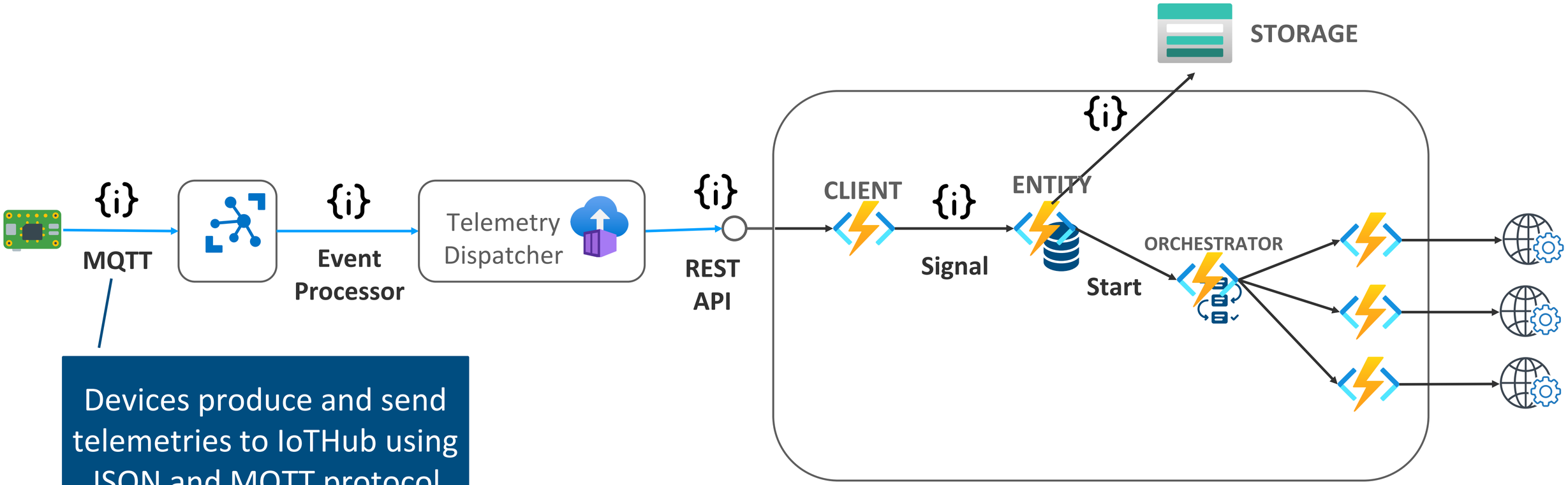


Front-End Layer:
Azure Functions expose Rest APIs to send telemetry, search devices and retrieve status for each device

Stateful Data Layer:
Durable Entities provide stateful layer to manage status for each device.

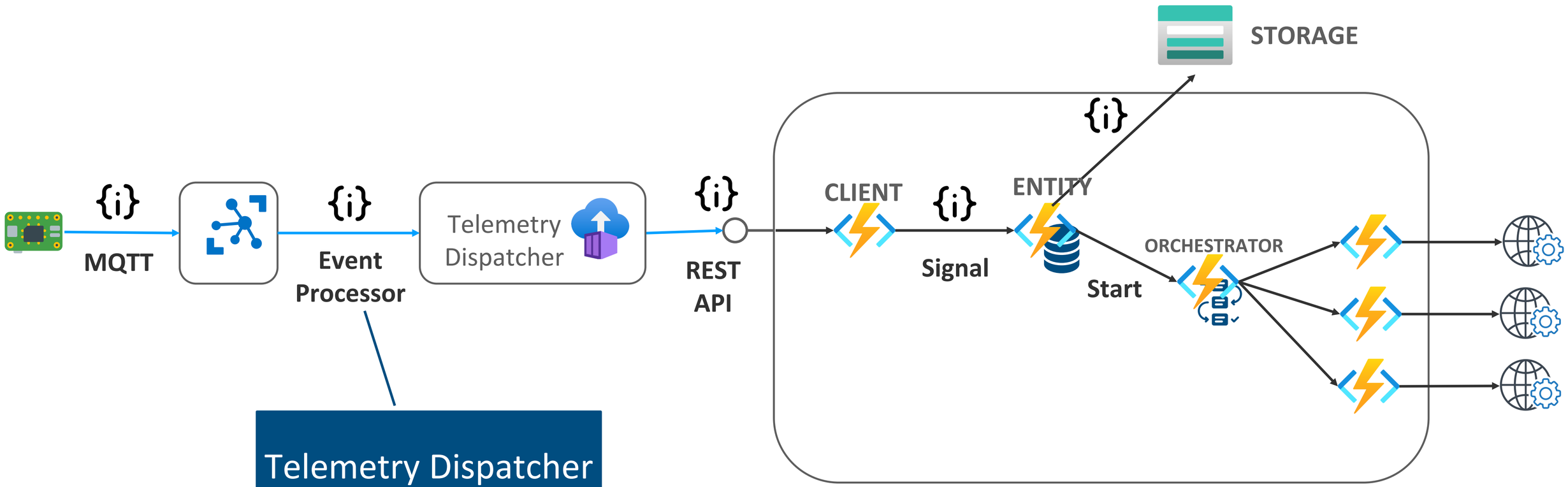
Integration Layer:
Durable Functions provide integration with external services and allow you to create complex workflow to integrate more than one external service.

FROM DEVICE TO ENTITIES…



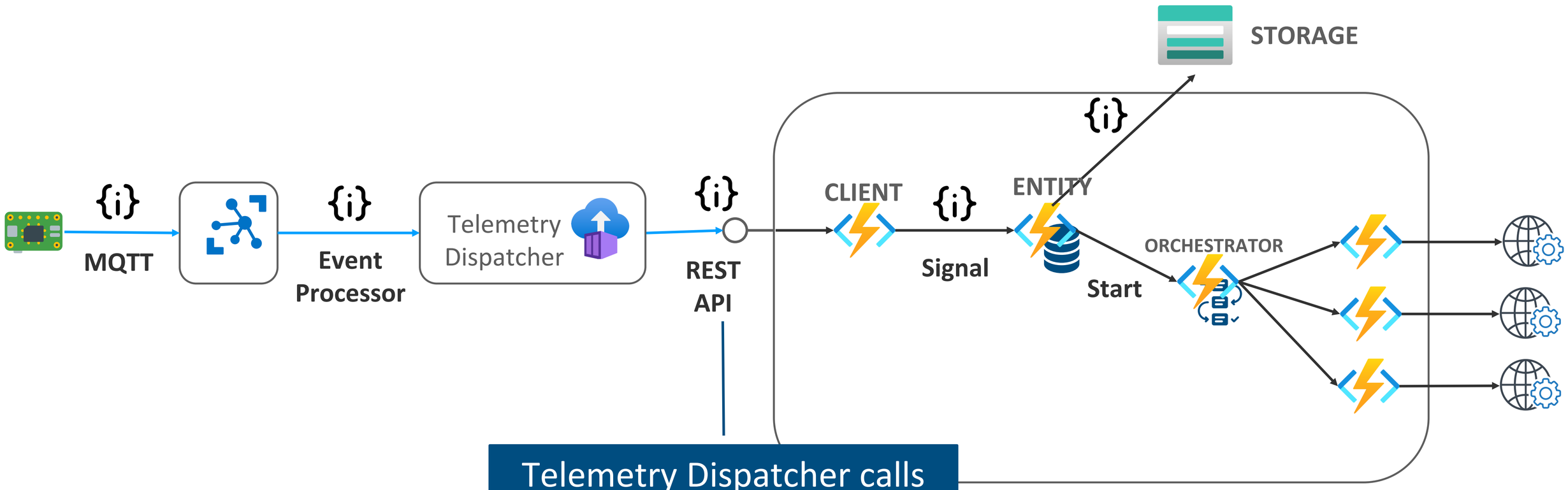
Devices produce and send telemetries to IoTHub using JSON and MQTT protocol

FROM DEVICE TO ENTITIES…



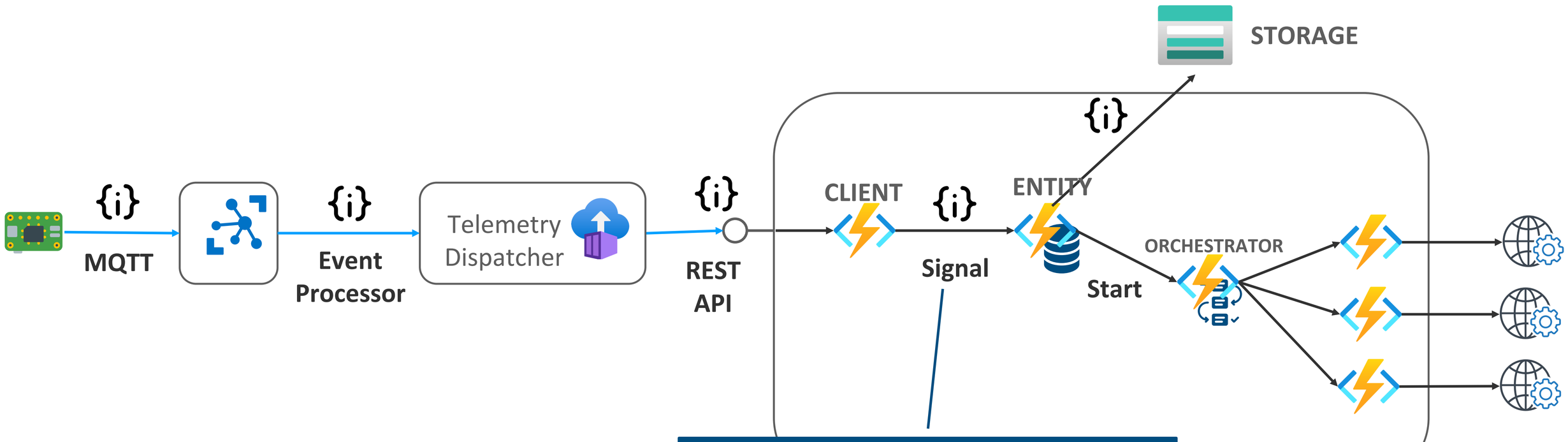
Telemetry Dispatcher retrieves telemetries from IoTHub

FROM DEVICE TO ENTITIES…



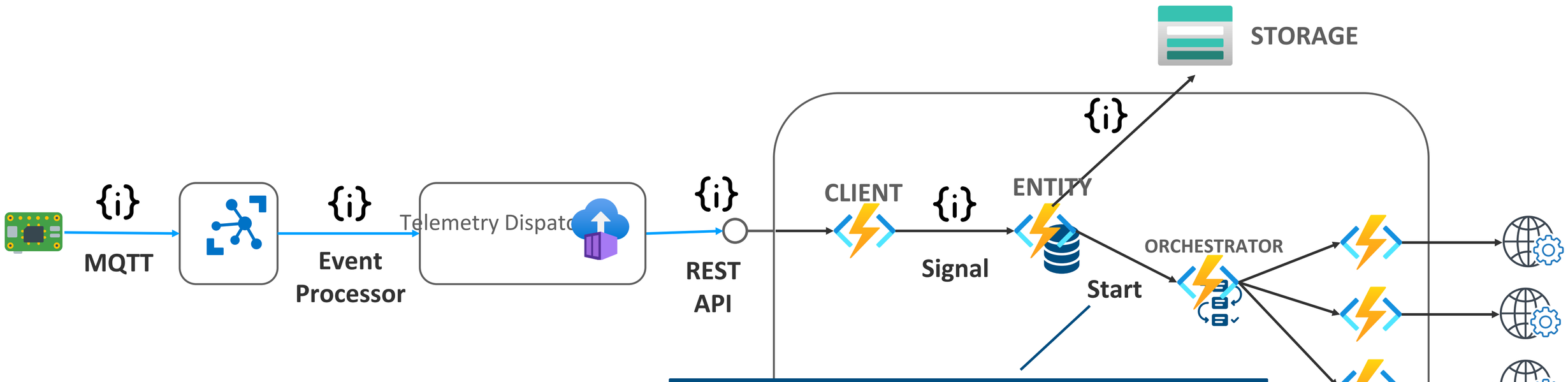
Telemetry Dispatcher calls the REST APIs exposed by the Azure Functions (Client Function)

FROM DEVICE TO ENTITIES…



Each client function signals the telemetry to the specific device (Entity Function)

FROM DEVICE TO ENTITIES…



The Entity manages the status (last telemetries) and start an Orchestrator Function if an alert occurs

SERVERLESS IOT PLATFORM

We need:

- Event-driven scalability
- State management abstraction
- Asynchronous interaction
- Different types of devices

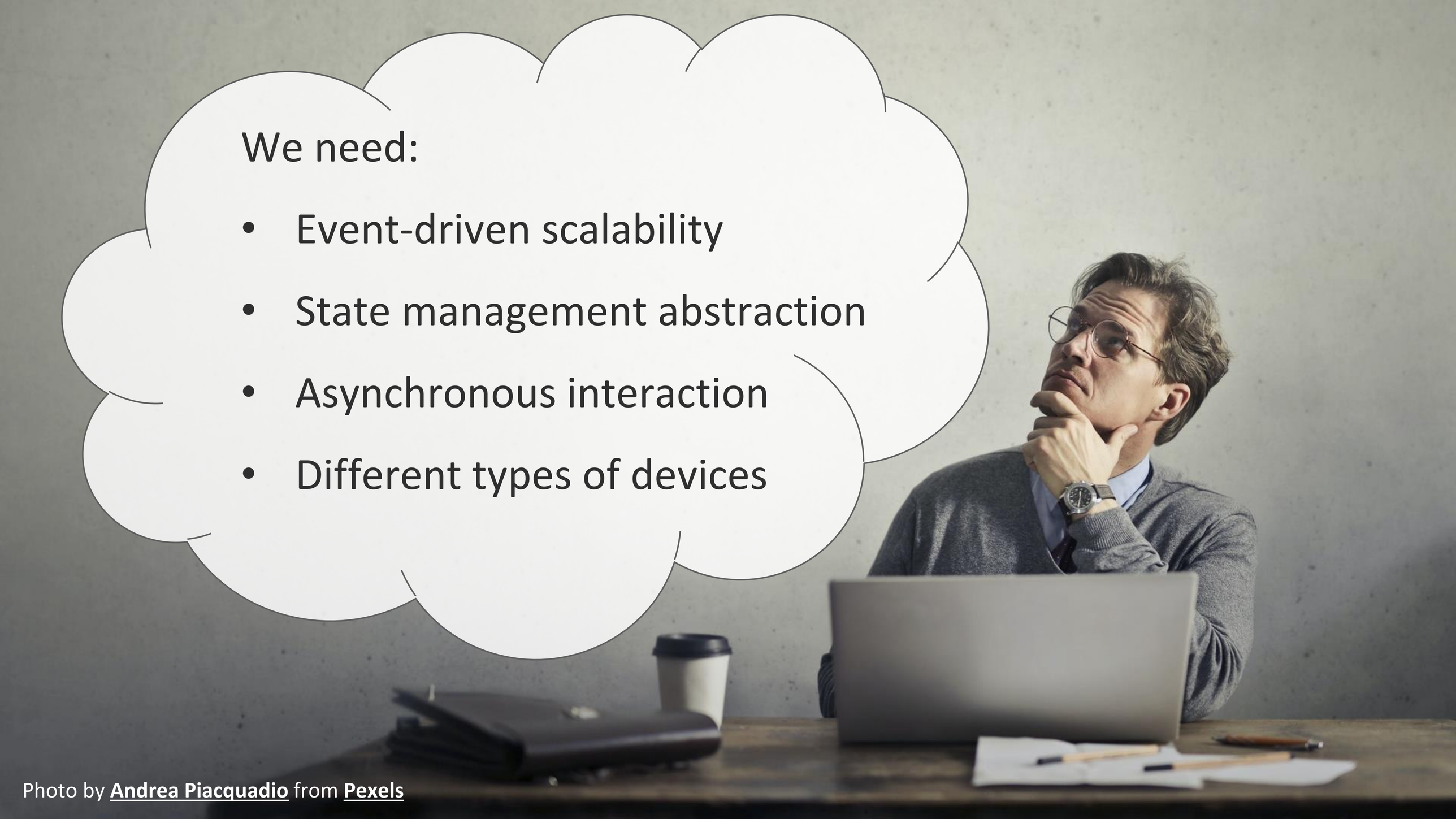


Photo by **Andrea Piacquadio** from **Pexels**

We need:

- ✓ Event-driven scalability → **Serverless**
- State management abstraction
- Asynchronous interaction
- Different types of devices

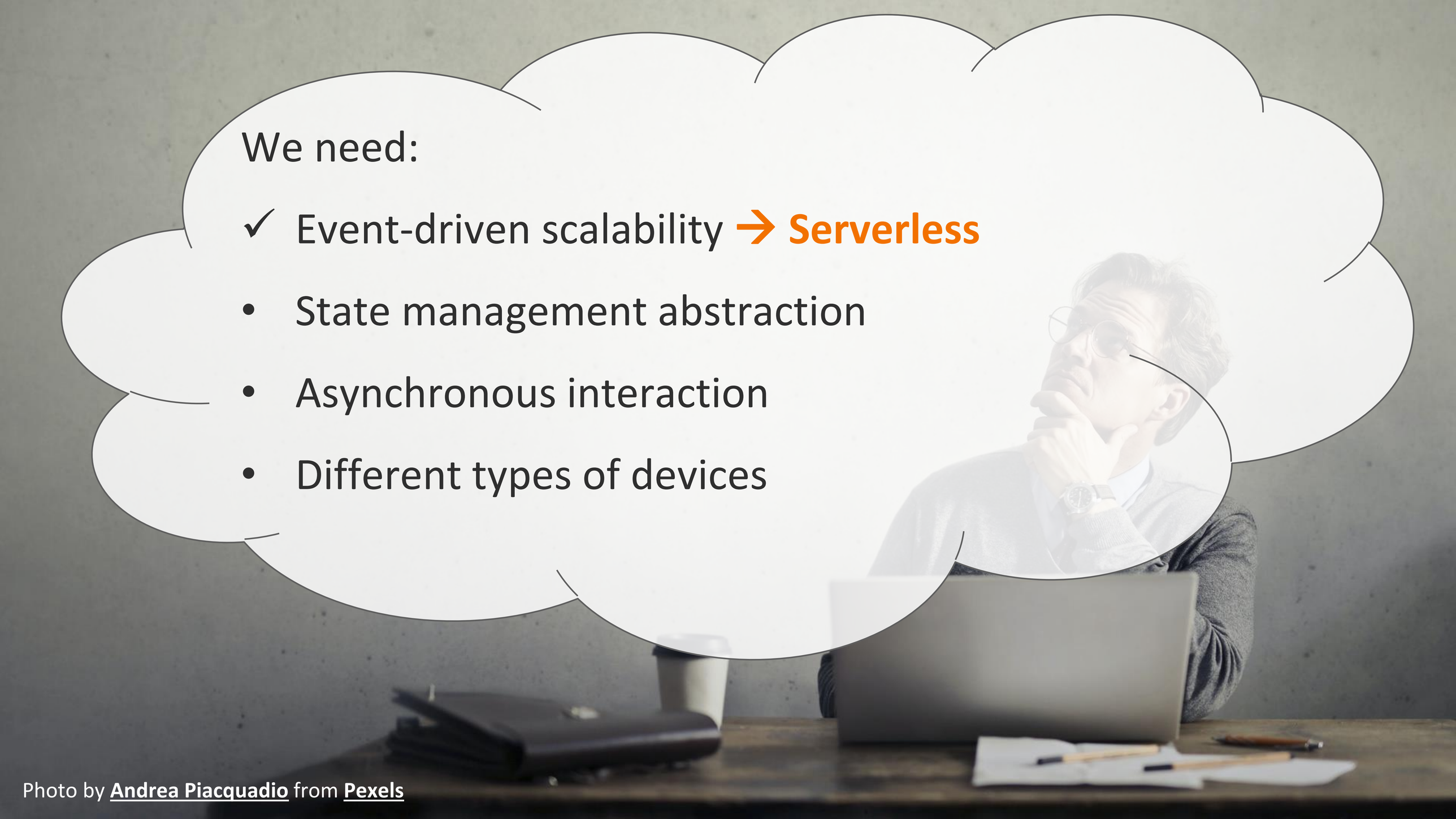


Photo by **Andrea Piacquadio** from **Pexels**

We need:

- ✓ Event-driven scalability → **Serverless**
- ✓ State management abstraction → **Durable Entities**
- Asynchronous interaction
- Different types of devices

We need:

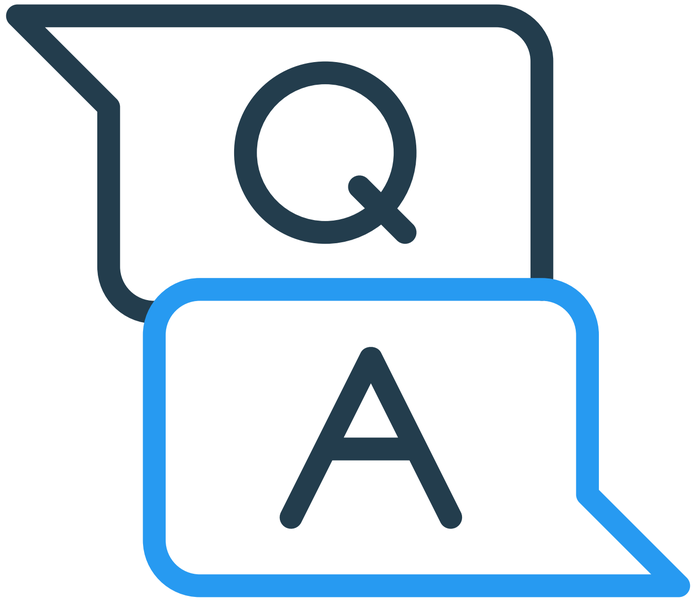
- ✓ Event-driven scalability → **Serverless**
- ✓ State management abstraction → **Durable Entities**
- ✓ Asynchronous interaction → **Signaling**
- Different types of devices

We need:

- ✓ Event-driven scalability → **Serverless**
- ✓ State management abstraction → **Durable Entities**
- ✓ Asynchronous interaction → **Signaling**
- ✓ Different types of devices → **Entity Interface**

http://bit.ly/MasteringServerless

Thanks for your attention!!!!



Connect with me on LinkedIn

linkedin.com/in/massimobonanni/

Massimo Bonanni

Azure Technical Trainer

massimo.bonanni@microsoft.com

@massimobonanni

#GLOBALAZURE

REFERENCES

Azure Functions Documentation
https://docs.microsoft.com/en-US/azure/azure-functions/

Durable Functions overview
https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-overview?tabs=csharp

Developer's guide to durable entities in .NET
https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-dotnet-entities

Entity Functions
https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-entities?tabs=csharp

Durable Task Framework
https://github.com/Azure/durabletask

GitHub ServerlessIoT Demo
https://github.com/massimobonanni/ServerlessIoT

Photo by Ashim D'Silva on Unsplash