Azure messaging!
La comparación definitiva

Nacho Fanjul - @nfanjul
Carmen Checa - @cmcheca

¡Gracias!

Patrocinadores Locales

Colabora

Presentación

Carmen Checa

Software Developer

@cmcheca

adajs

Presentación

plain concepts

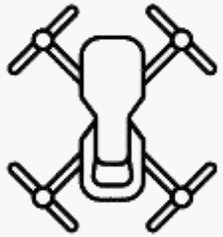
Nacho Fanjul

Software Developer

@ nfanjul@plainconcepts.com

@nfanjul

https://github.com/nfanjul



Cuando desarrollas en *Azure* siempre existen varios caminos…

Y aunque todos los caminos llevan a Roma…

Todos esos caminos no son los óptimos…

¿Qué necesitamos?

¿Cómo vamos a realizarlo?

¿De que vamos a hablar?

Mensaje != Evento

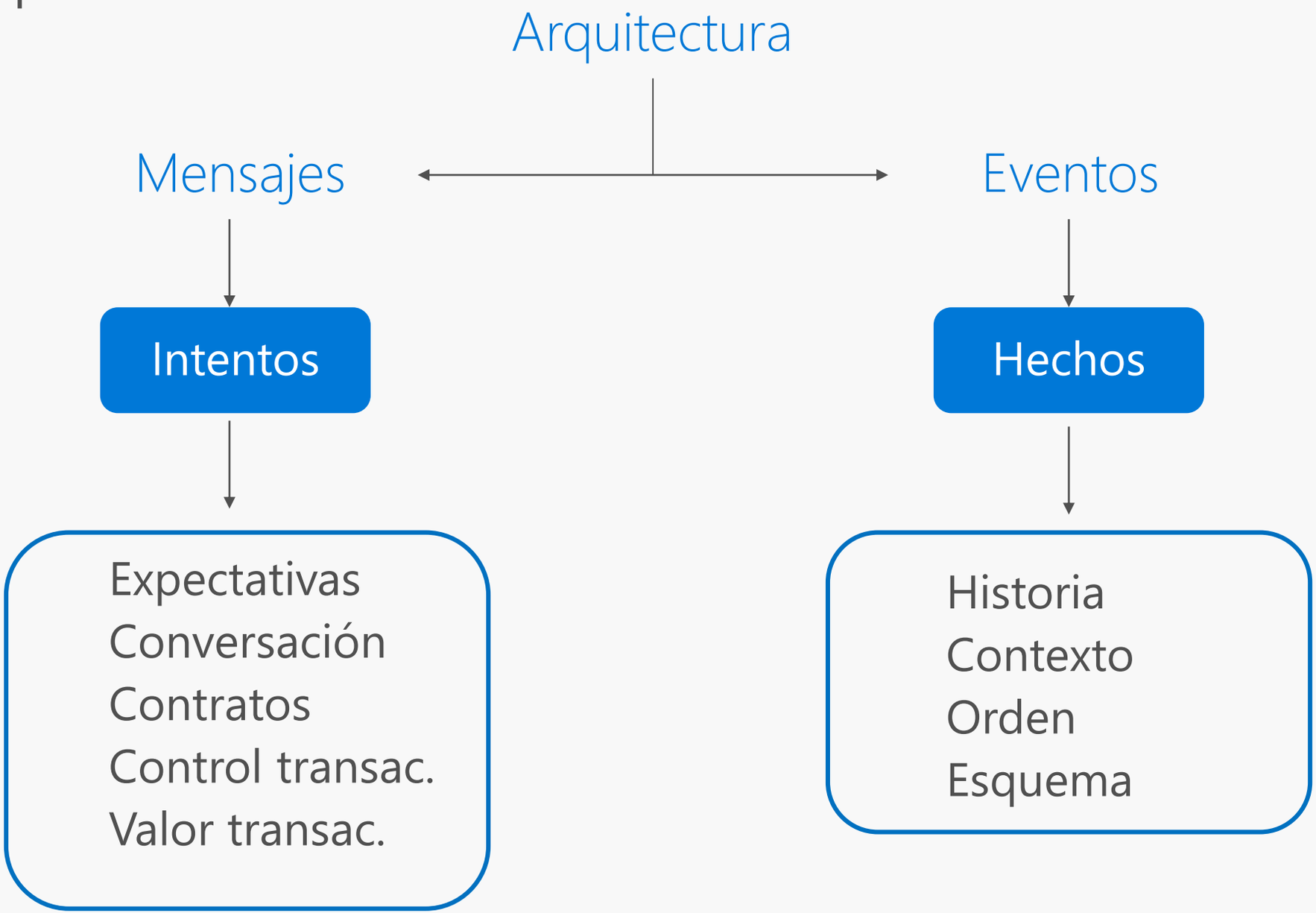
Mensaje != Evento != Evento

¿Mensaje? ¿Evento?

La **mensajería** es mantener una conversación con alguien.

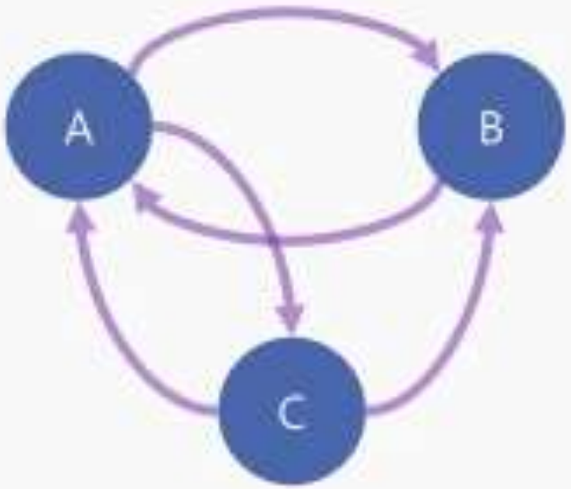
El **evento** es decirle a alguien (en una conversación) lo que acaba de pasar.

Conceptos

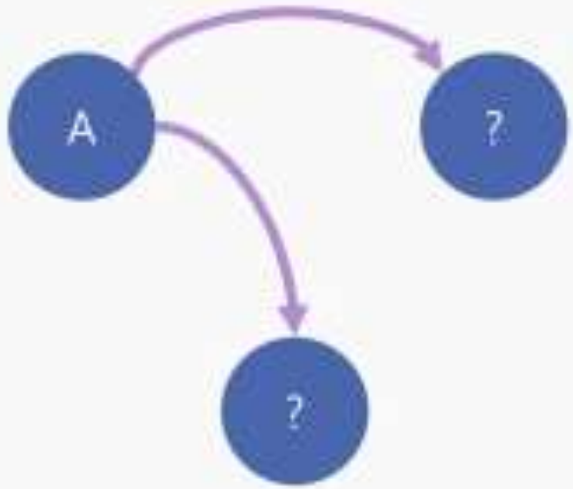

Arquitectura
Mensajes
Eventos
Intentos
Hechos
Expectativas
Conversación
Contratos
Control transac.
Valor transac.
Historia
Contexto
Orden
Esquema

¿Cómo se comunican?

Mensajes



Eventos

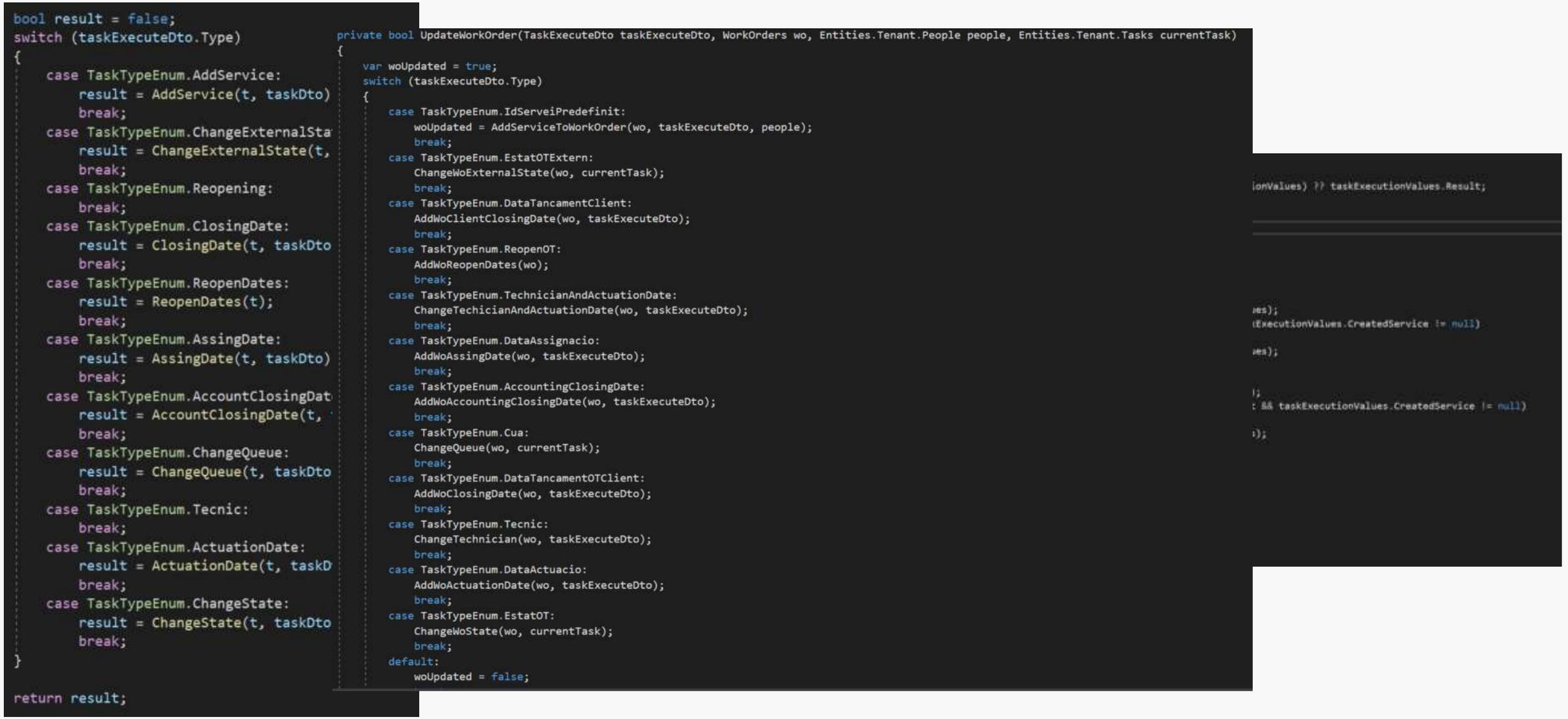


¿Empezamos?

Nuevo proyecto!!

Nuevo proyecto. Refactorización…

Lo que nos encontramos



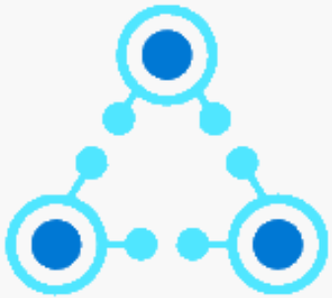
```
bool result = false;
switch (taskExecuteDto.Type)
{
    case TaskTypeEnum.AddService:
        result = AddService(t, taskDto)
        break;
    case TaskTypeEnum.ChangeExternalSta
        result = ChangeExternalState(t,
        break;
    case TaskTypeEnum.Reopening:
        break;
    case TaskTypeEnum.ClosingDate:
        result = ClosingDate(t, taskDto
        break;
    case TaskTypeEnum.ReopenDates:
        result = ReopenDates(t);
        break;
    case TaskTypeEnum.AssingDate:
        result = AssingDate(t, taskDto)
        break;
    case TaskTypeEnum.AccountClosingDat
        result = AccountClosingDate(t,
        break;
    case TaskTypeEnum.ChangeQueue:
        result = ChangeQueue(t, taskDto
        break;
    case TaskTypeEnum.Tecnic:
        break;
    case TaskTypeEnum.ActuationDate:
        result = ActuationDate(t, taskD
        break;
    case TaskTypeEnum.ChangeState:
        result = ChangeState(t, taskDto
        break;
}

return result;
```

```
private bool UpdateWorkOrder(TaskExecuteDto taskExecuteDto, WorkOrders wo, Entities.Tenant.People people, Entities.Tenant.Tasks currentTask)
{
    var woUpdated = true;
    switch (taskExecuteDto.Type)
    {
        case TaskTypeEnum.IdServeiPredefinit:
            woUpdated = AddServiceToWorkOrder(wo, taskExecuteDto, people);
            break;
        case TaskTypeEnum.EstatOTExtern:
            ChangeWoExternalState(wo, currentTask);
            break;
        case TaskTypeEnum.DataTancamentClient:
            AddWoClientClosingDate(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.ReopenOT:
            AddWoReopenDates(wo);
            break;
        case TaskTypeEnum.TechnicianAndActuationDate:
            ChangeTechicianAndActuationDate(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.DataAssignacio:
            AddWoAssingDate(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.AccountingClosingDate:
            AddWoAccountingClosingDate(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.Cua:
            ChangeQueue(wo, currentTask);
            break;
        case TaskTypeEnum.DataTancamentOTClient:
            AddWoClosingDate(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.Tecnic:
            ChangeTechnician(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.DataActuacio:
            AddWoActuationDate(wo, taskExecuteDto);
            break;
        case TaskTypeEnum.EstatOT:
            ChangeWoState(wo, currentTask);
            break;
        default:
            woUpdated = false;
```

Service Bus

Service Bus

- Servicio de mensajería
  - Envío de información entre aplicaciones y servicios
  - Asincronía por excelencia
  - Mensajería estructurada de tipo FIFO
  - Publicación-suscripción

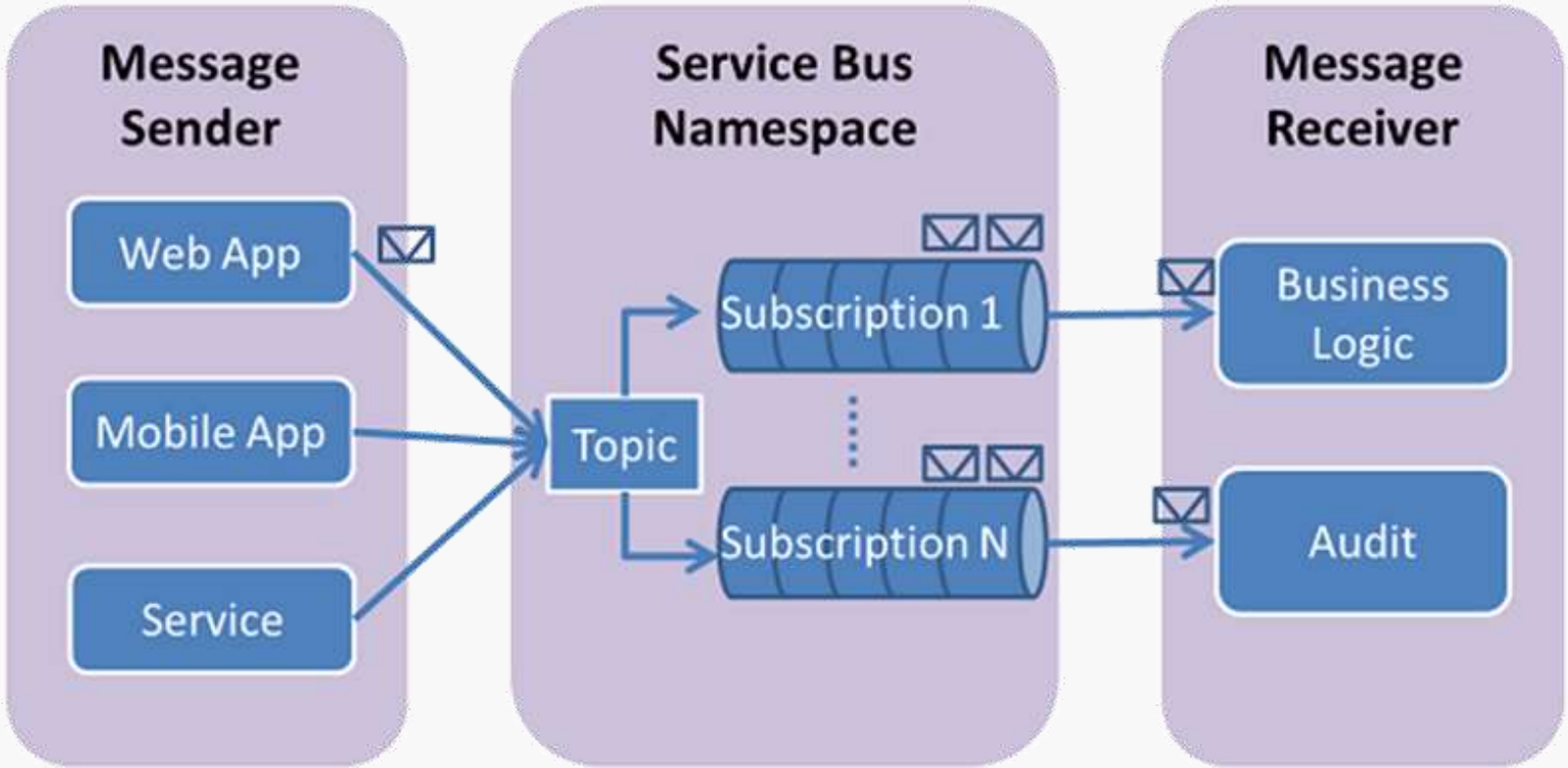


- Grandes posibilidades
  - Mensajería
  - Desacoplamiento de aplicaciones
  - Flujos de trabajo

¿Qué nos aporta?

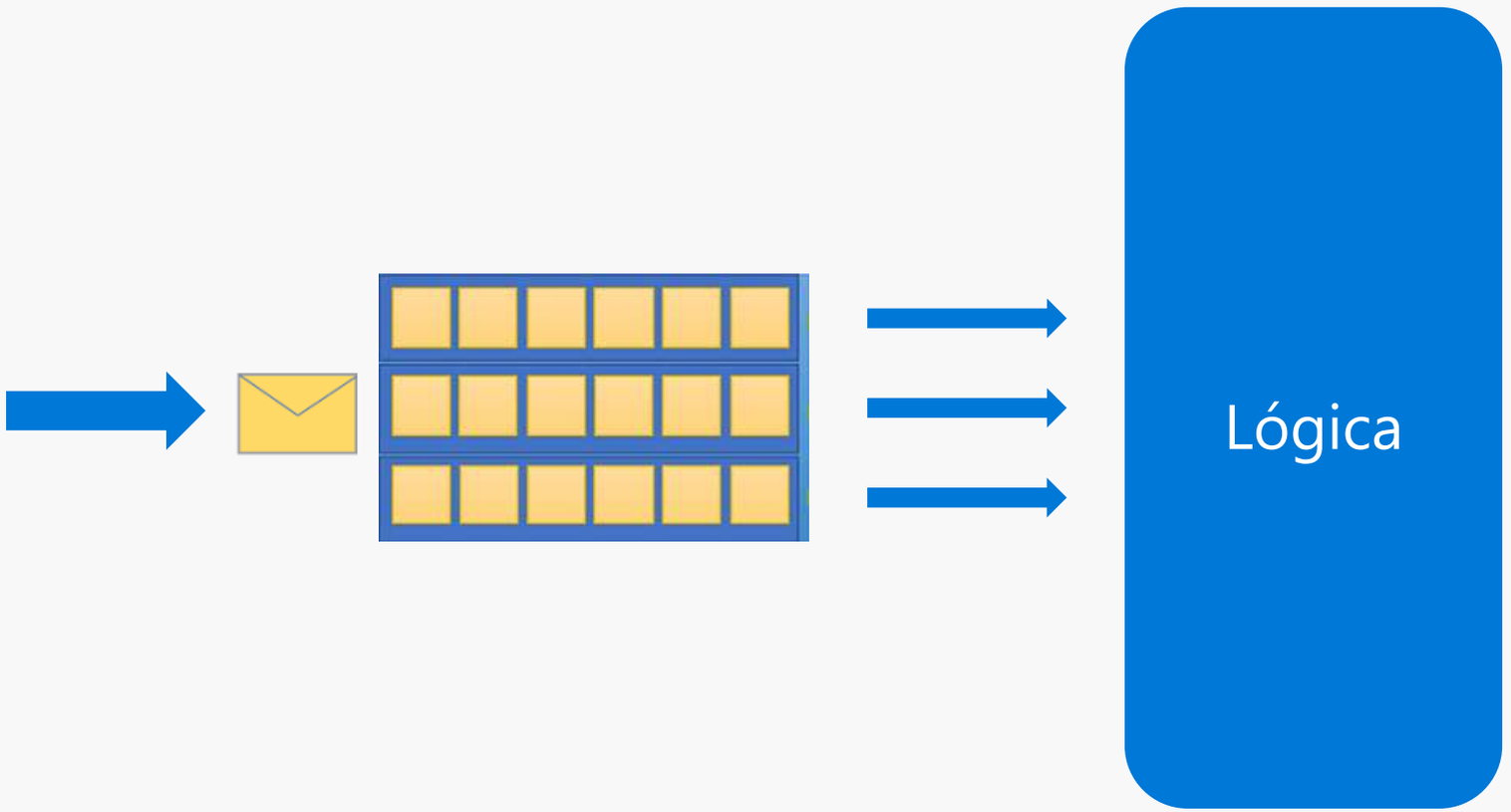
- Asincronía
- Procesamiento por lotes
- Transacciones
- Colas de mensajes fallidos
- Filtrado, y detección de duplicados
- Al menos una entrega
- Entrega en orden opcional

Escenarios


Message Sender
Web App
Mobile App
Service
Service Bus Namespace
Topic
Subscription 1
Subscription N
Message Receiver
Business Logic
Audit

Evolución ARQ

```
bool result = false;
switch (taskExecuteDto.Type)
{
    case TaskTypeEnum.AddService:
        result = AddService(t, taskDto);
        break;
    case TaskTypeEnum.ChangeExternalState:
        result = ChangeExternalState(t, taskDto);
        break;
    case TaskTypeEnum.Reopening:
        break;
    case TaskTypeEnum.ClosingDate:
        result = ClosingDate(t, taskDto);
        break;
    case TaskTypeEnum.ReopenDates:
        result = ReopenDates(t);
        break;
    case TaskTypeEnum.AssingDate:
        result = AssingDate(t, taskDto);
        break;
    case TaskTypeEnum.AccountClosingDate:
        result = AccountClosingDate(t, taskDto);
        break;
    case TaskTypeEnum.ChangeQueue:
        result = ChangeQueue(t, taskDto);
        break;
    case TaskTypeEnum.Tecnic:
        break;
    case TaskTypeEnum.ActuationDate:
        result = ActuationDate(t, taskDto);
        break;
    case TaskTypeEnum.ChangeState:
        result = ChangeState(t, taskDto);
        break;
}

return result;
```



Event Grid

Event Grid

- Simplificación del consumo de eventos (PUB - SUB)
  - Reacción a eventos (casi tiempo real)
  - Fácil implementación de ARQ basadas en eventos

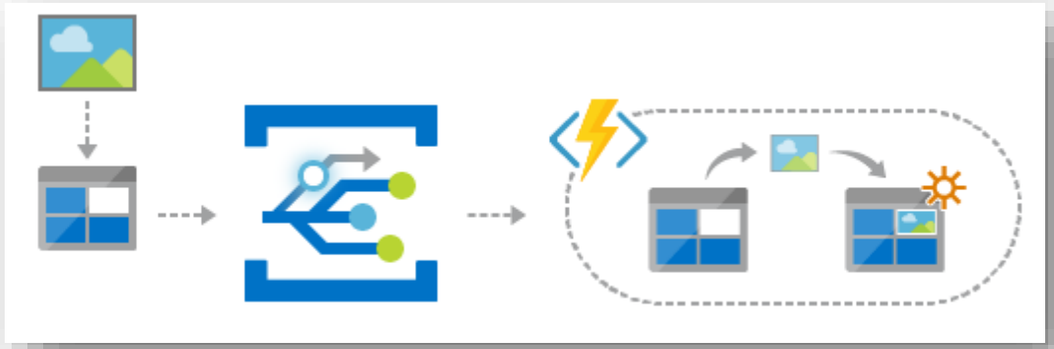
- Aplicaciones confiables en cloud
  - Escalable y alta disponibilidad
  - Grandes posibilidades con *serverless*
  - Automatización

¿Qué nos aporta?

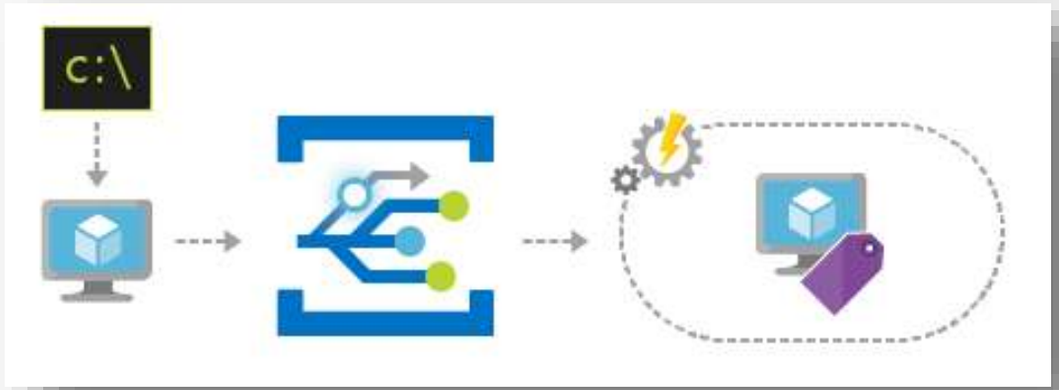
- Siempre disponible
- CASI tiempo real
- Escalable
- Plataforma agnóstica (WebHook)
- Lenguaje agnóstico (Protocolo HTTP)

Escenarios

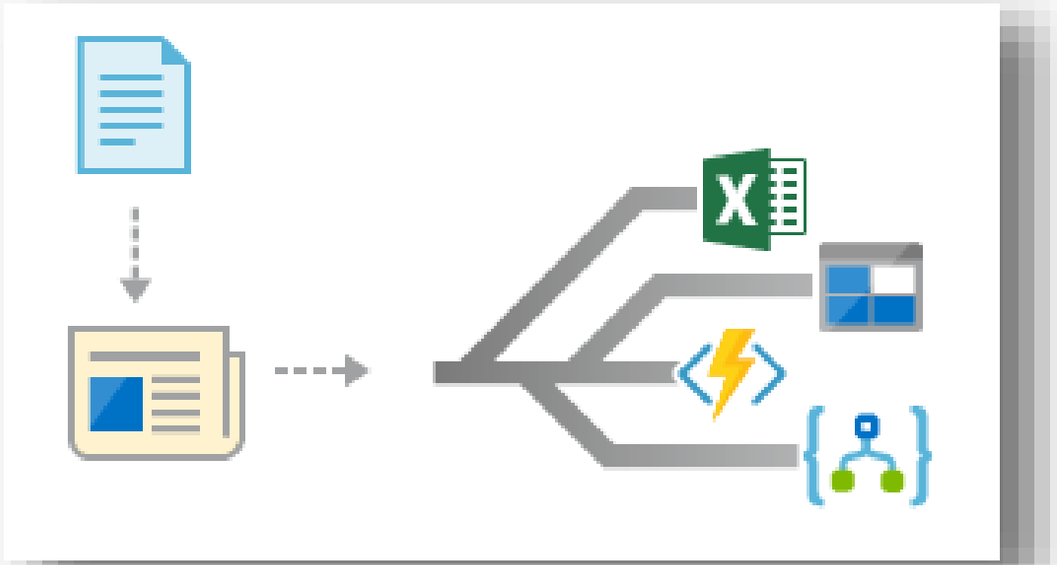
Arquitectura serveless



Automatización de procesos



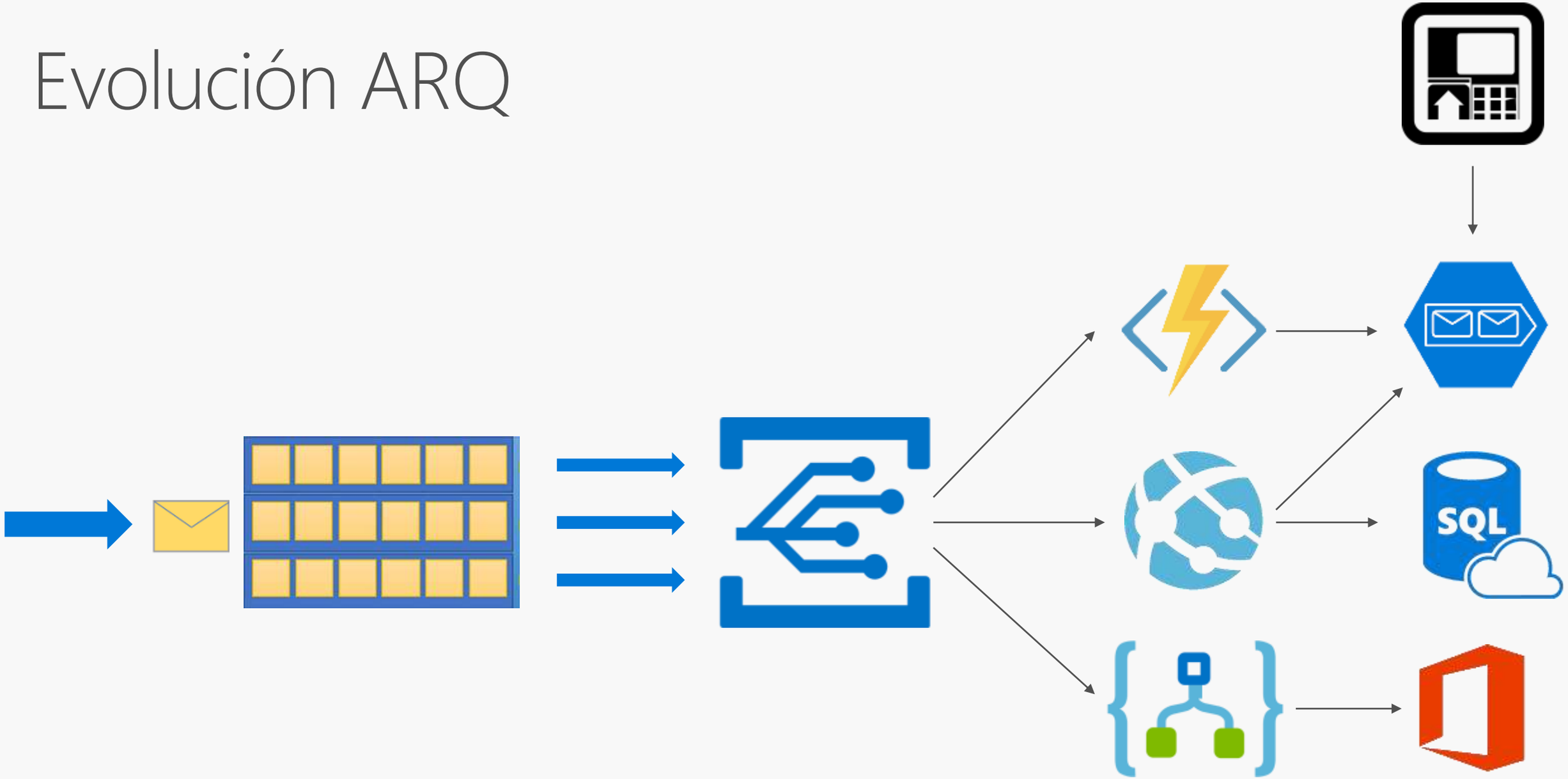
Integración



Evolución ARQ

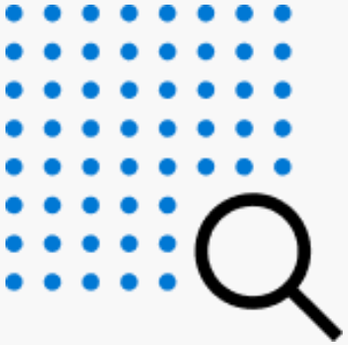
```
bool result = false;
switch (taskExecuteDto.Type)
{
    case TaskTypeEnum.AddService:
        result = AddService(t, taskDto);
        break;
    case TaskTypeEnum.ChangeExternalState:
        result = ChangeExternalState(t, taskDto);
        break;
    case TaskTypeEnum.Reopening:
        break;
    case TaskTypeEnum.ClosingDate:
        result = ClosingDate(t, taskDto);
        break;
    case TaskTypeEnum.ReopenDates:
        result = ReopenDates(t);
        break;
    case TaskTypeEnum.AssingDate:
        result = AssingDate(t, taskDto);
        break;
    case TaskTypeEnum.AccountClosingDate:
        result = AccountClosingDate(t, taskDto);
        break;
    case TaskTypeEnum.ChangeQueue:
        result = ChangeQueue(t, taskDto);
        break;
    case TaskTypeEnum.Tecnic:
        break;
    case TaskTypeEnum.ActuationDate:
        result = ActuationDate(t, taskDto);
        break;
    case TaskTypeEnum.ChangeState:
        result = ChangeState(t, taskDto);
        break;
}

return result;
```

Evolución ARQ



Event Hub

Event Hub



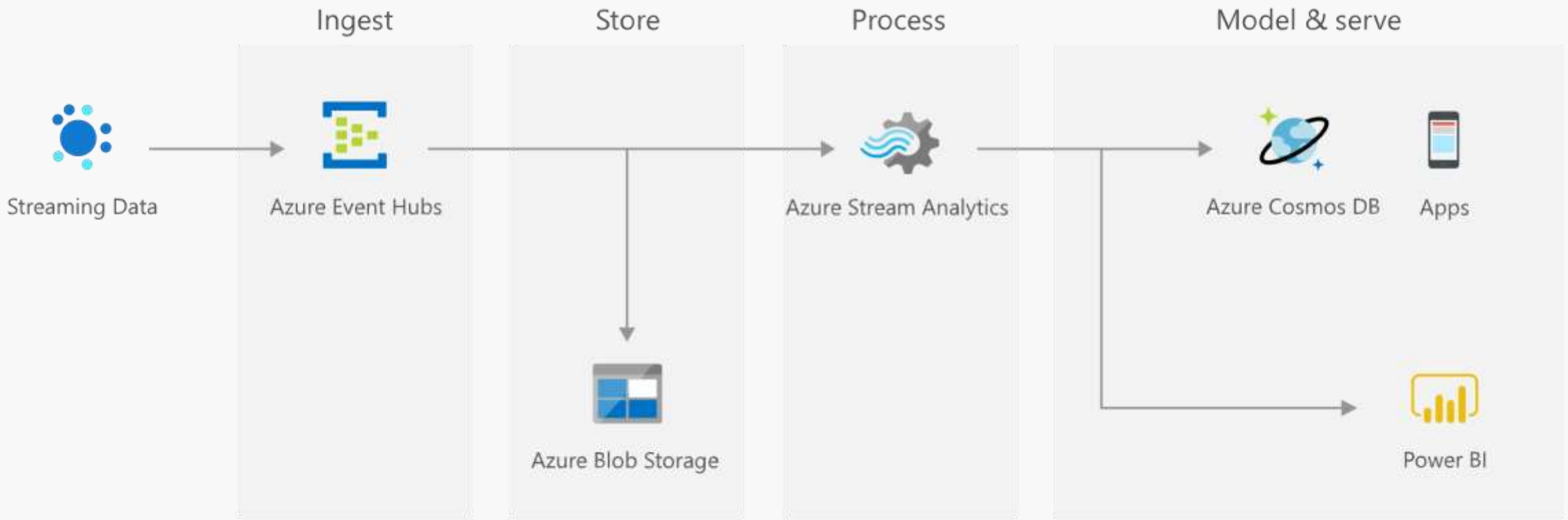
- Canalizador de datos
  - Streaming
  - Ingesta masiva de datos
  - Procesamiento de millones de datos por segundo

- Confianza y posibilidades
  - Detección de anomalías (fraude/valores atípicos)
  - Paneles en vivo
  - Procesamiento de transacciones
  - Telemetrías

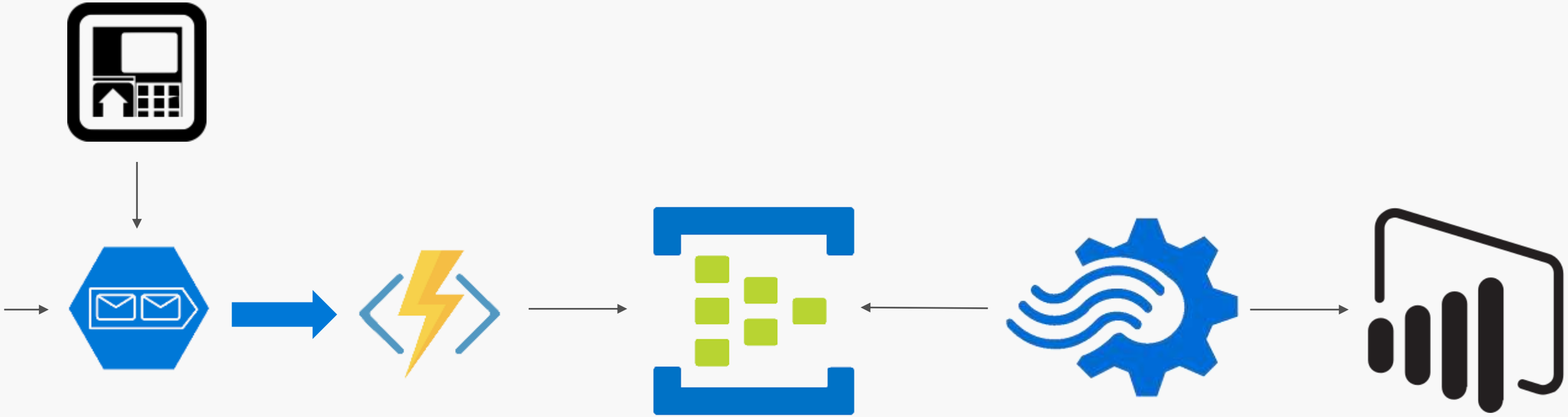
¿Qué nos aporta?

- Baja latencia
- Capacidad de recibir y procesar millones de eventos por segundo
- Al menos una entrega
- Solución streaming sin servidor

Escenarios


Ingest
Store
Process
Model & serve
Streaming Data
Azure Event Hubs
Azure Blob Storage
Azure Stream Analytics
Azure Cosmos DB
Apps
Power BI

Evolución ARQ



Conclusiones

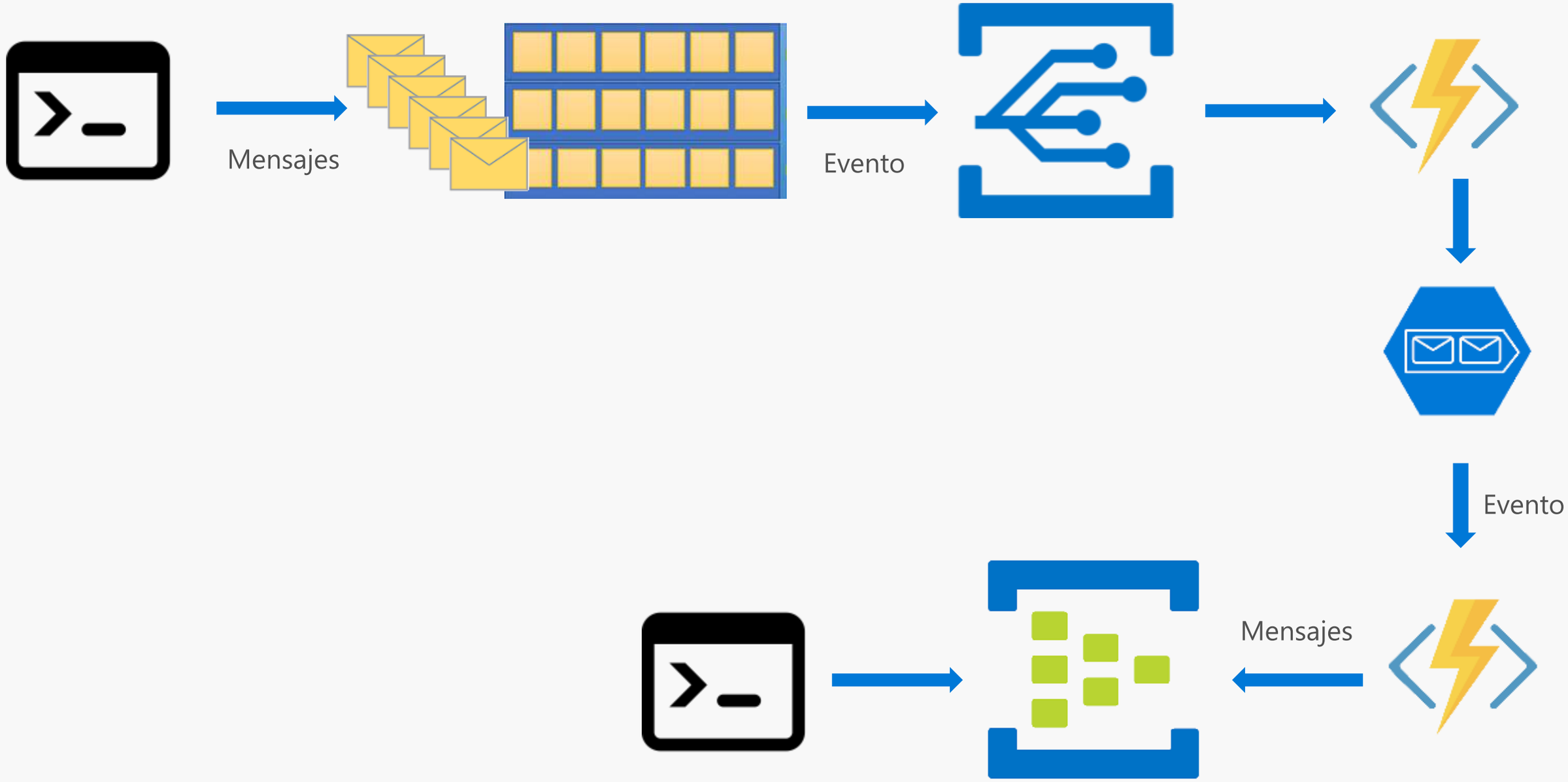
Resumen

Servicio	Proposito	Tipo	Uso
Event Grid	Prog. reactiva	Evento - distribución	Reacción a cambios
Event Hub	Big data	Evento - streaming	Telemetría y streaming
Service Bus	Mensajería empresarial	Mensaje	Procesamiento y transacciones

Cuando usar cada uno

Servicio	Proposito
Event Grid	• Desarrollo 100% desacoplado. • Reaccionar a algo que ha ocurrido.
Event Hub	• Cuando necesitemos alta capacidad de procesamiento • Streaming de datos
Service Bus	• Transaccionabilidad • Alta fiabilidad (no se puede perder ni una transacción)

Demo Time!


Mensajes
Evento
Evento
Mensajes

Q&A

Nacho Fanjul- @nfanjul
Carmen Checa- @cmcheca

2019

Global Azure BOOTCAMP