Headless & API date | Liferay | mia Platform | Intesys

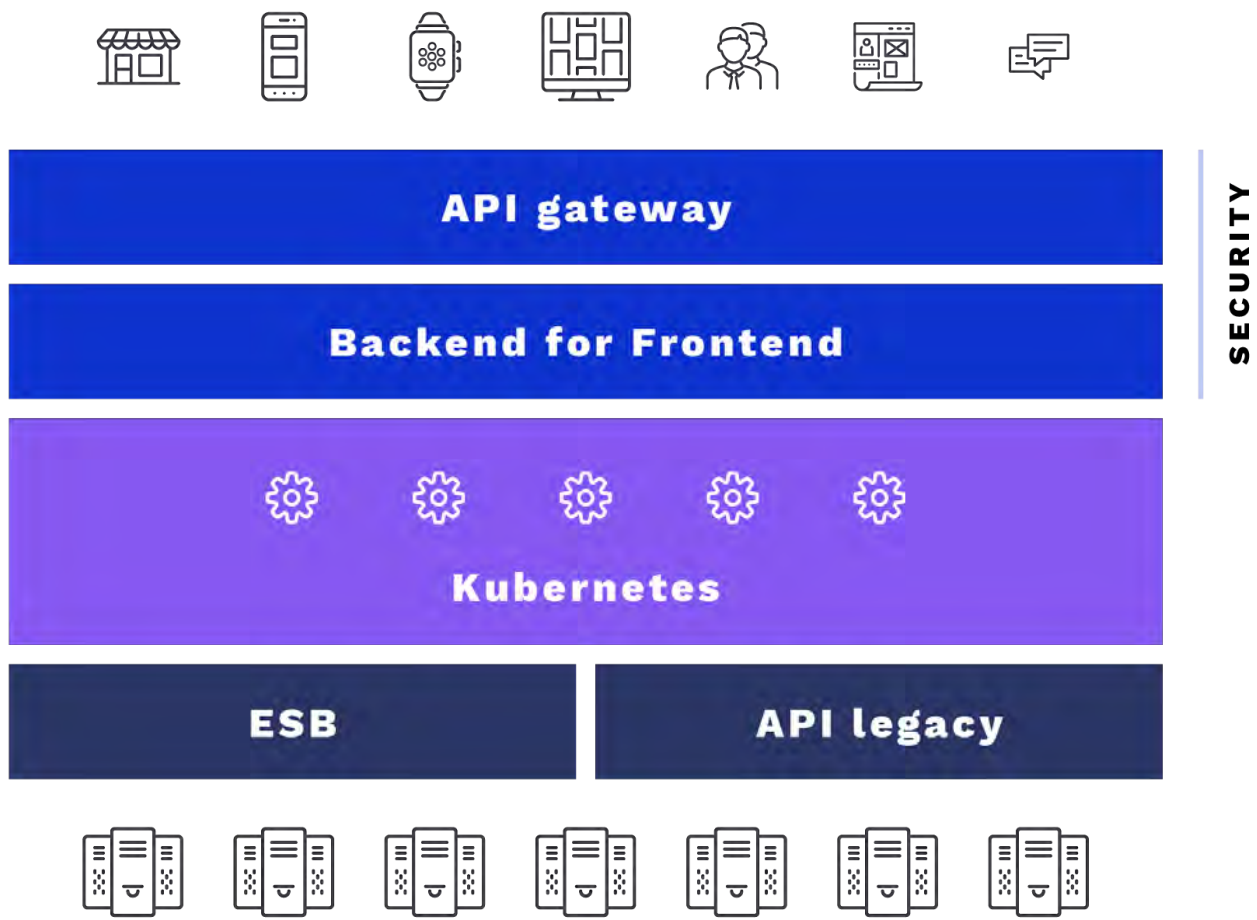
SESSIONE IT STRATEGY

System integration:
da Monolite a Microservizi e Architetture Event-driven

PAOLO QUAGLIA
API STRATEGIST & IT EXPERT, INTESYS

Agenda:

1. ARCHITETTURA MONOLITICA
2. ARCHITETTURA A MICROSERVIZI
3. TRANSIZIONE
4. EVENT-DRIVEN ARCHITECTURE
5. BENEFICI E CONCLUSIONI


API gateway
Backend for Frontend
Kubernetes
ESB
API legacy
SECURITY

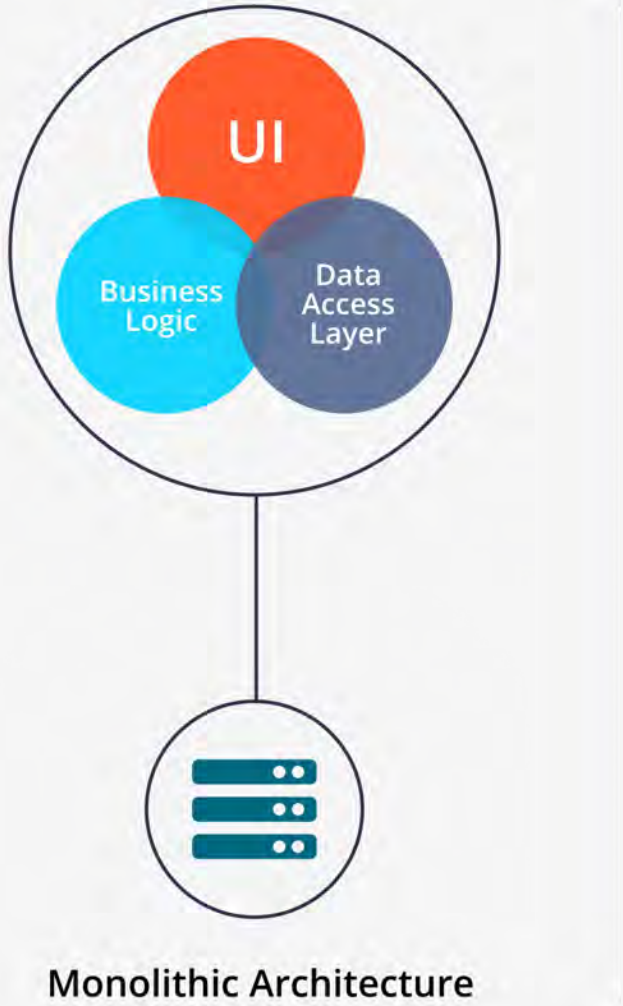
Architettura monolitica

Architettura monolitica

Tutte le funzionalità del sistema
sono racchiuse in un'unica entità.

Spesso la base di dati è unica
e centralizzata per tutte le funzioni.

Sistema monolitico è progettato
per essere autonomo.



Fonte: https://www.weave.works/blog/microservices-and-continuous-delivery/

Architettura monolitica - vantaggi

Semplici da sviluppare e da installare

Se vengono utilizzati standard come MVC gli sviluppatori lavorano con logiche full stack.

Di solito il deploy consiste in un unico artefatto che contiene tutta la applicazione.

Unica tecnologia

Un'unica tecnologia per lo sviluppo dei layer appiccativi.

Semplice testare

Test della intera applicazione in un unico passaggio.

Architettura monolitica – punti di attenzione

Difficili da evolvere e tempi di distribuzione lunghi

Ogni modifica impatta tutto il sistema.

Il rilascio riguarda l'intero sistema.

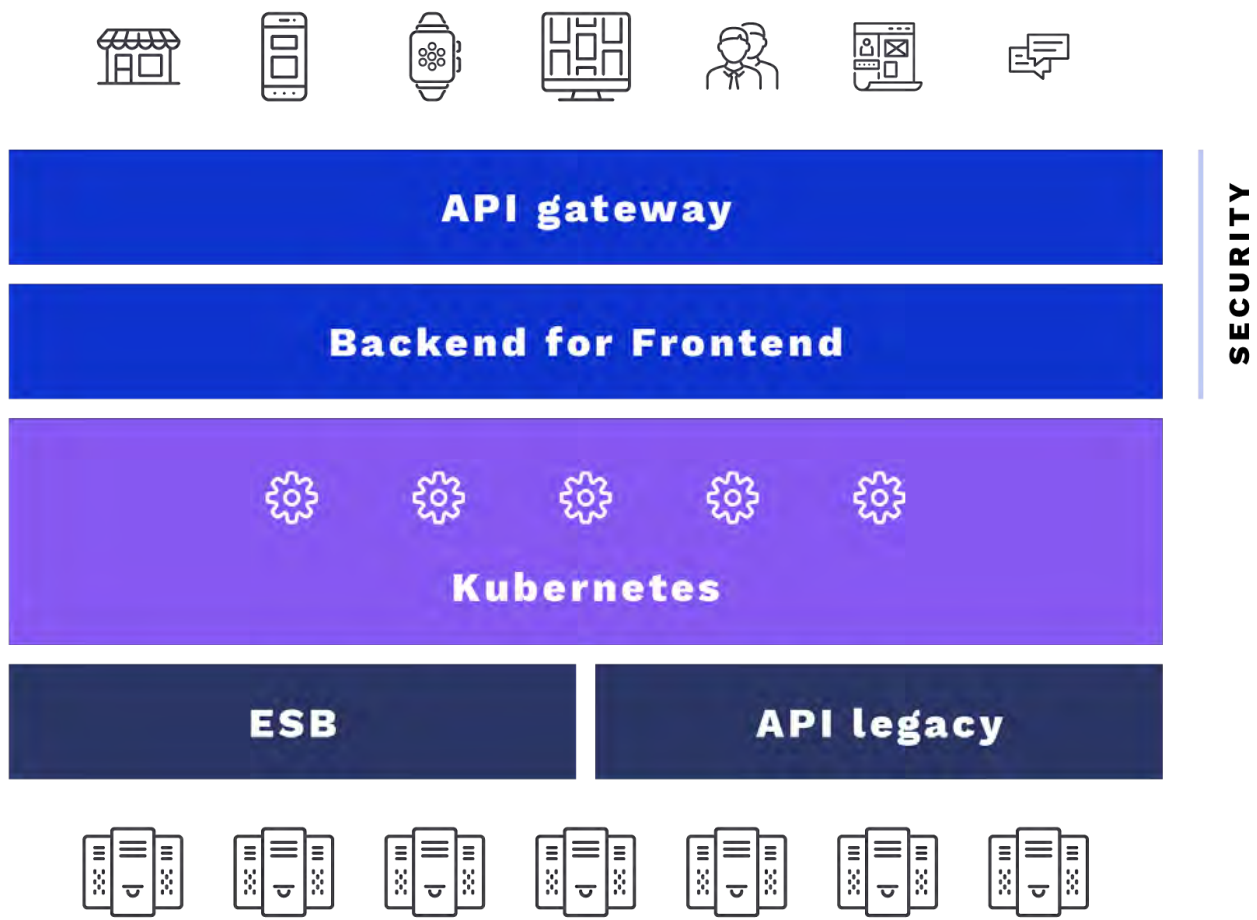
Difficili da scalare

Difficili da scalare.
Serve scalare tutto il sistema.

Complessità del codice e possibile obsolescenza

Se l'applicazione diventa molto complessa, capire il codice diventa molto arduo.

E può diventare difficile trovare risorse/know how sulla tecnologia/prodotto.


API gateway
Backend for Frontend
Kubernetes
ESB
API legacy
SECURITY

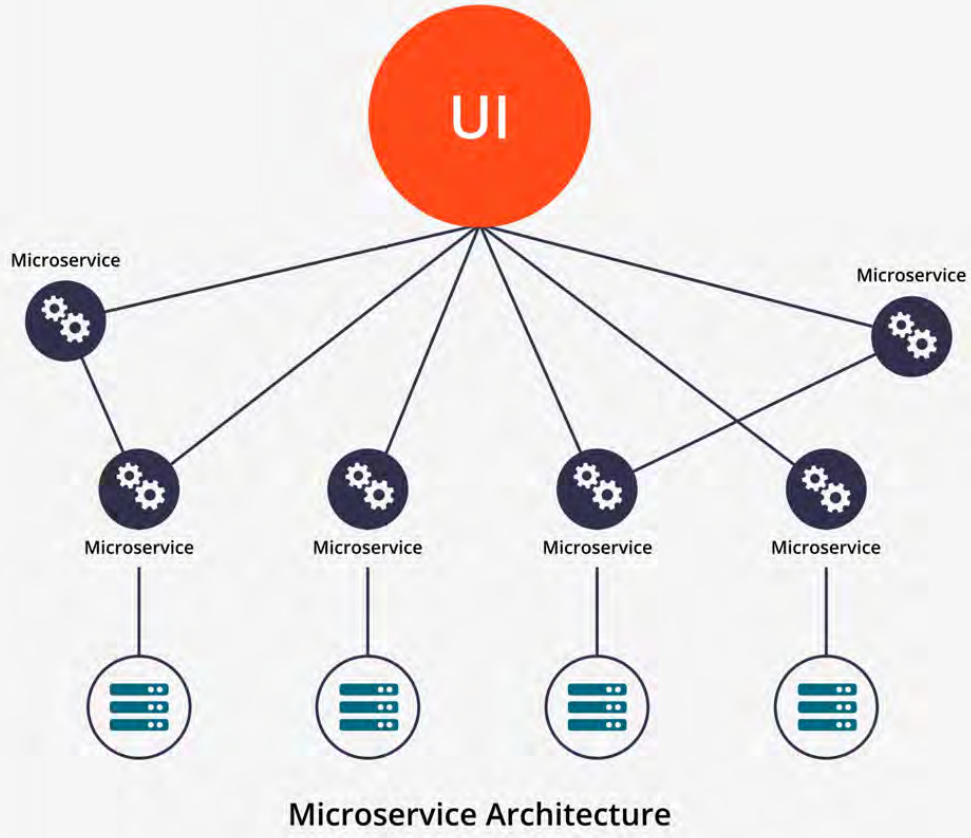
Architettura a microservizi

Architettura a microservizi

I microservizi sono un approccio per progettare e organizzare l'architettura software.

Ogni microservizio:

- indipendente
- piccole dimensioni
- comunica con gli altri tramite API ben definite
- propria base di dati



Fonte: https://www.weave.works/blog/microservices-and-continuous-delivery/

Architettura a microservizi - vantaggi

Maggiore indipendenza e resilienza

Grazie alla scomposizione in più servizi, un problema legato a uno di questi non genera impatti sugli altri.

Sviluppati da team diversi con **linguaggi diversi.**

Migliore scalabilità

Ad ogni servizio possono essere attribuite risorse proporzionali al carico di lavoro.

Permette cambiamenti e rilasci veloci

Gli interventi sulle applicazioni sono puntuali, più rapidi e i rilasci avvengono “a caldo”.

Architettura a Microservizi – Punti di attenzione

Controllo della coerenza dell'applicazione

La **progettazione** riveste un livello cruciale.

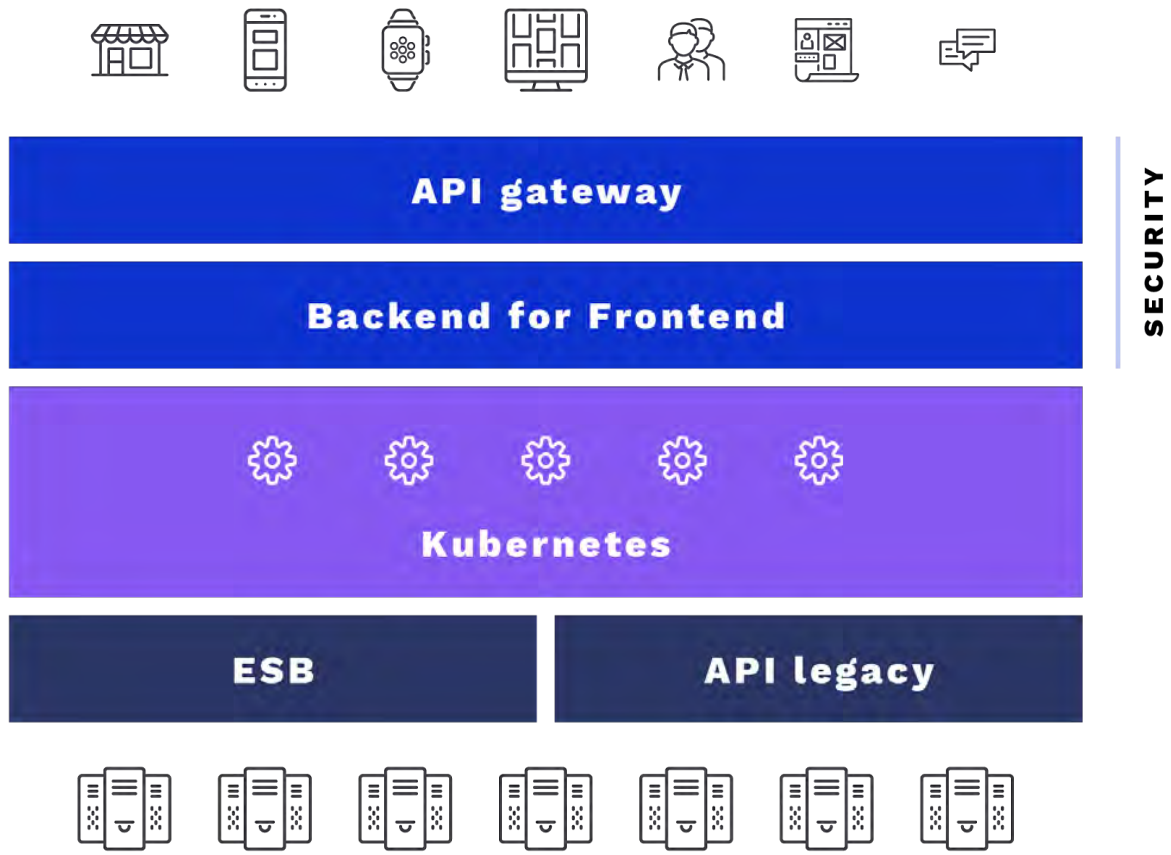
Richiedono un elevato livello di **automazione.**

Comunicazione complessa

Sistema distribuito che richiede bassi livelli di latenza e spesso sistemi di code di messaggi.

Test dei singoli servizi e dell'intero sistema

Quantità e varietà dei test.


API gateway
Backend for Frontend
Kubernetes
ESB
API legacy
SECURITY

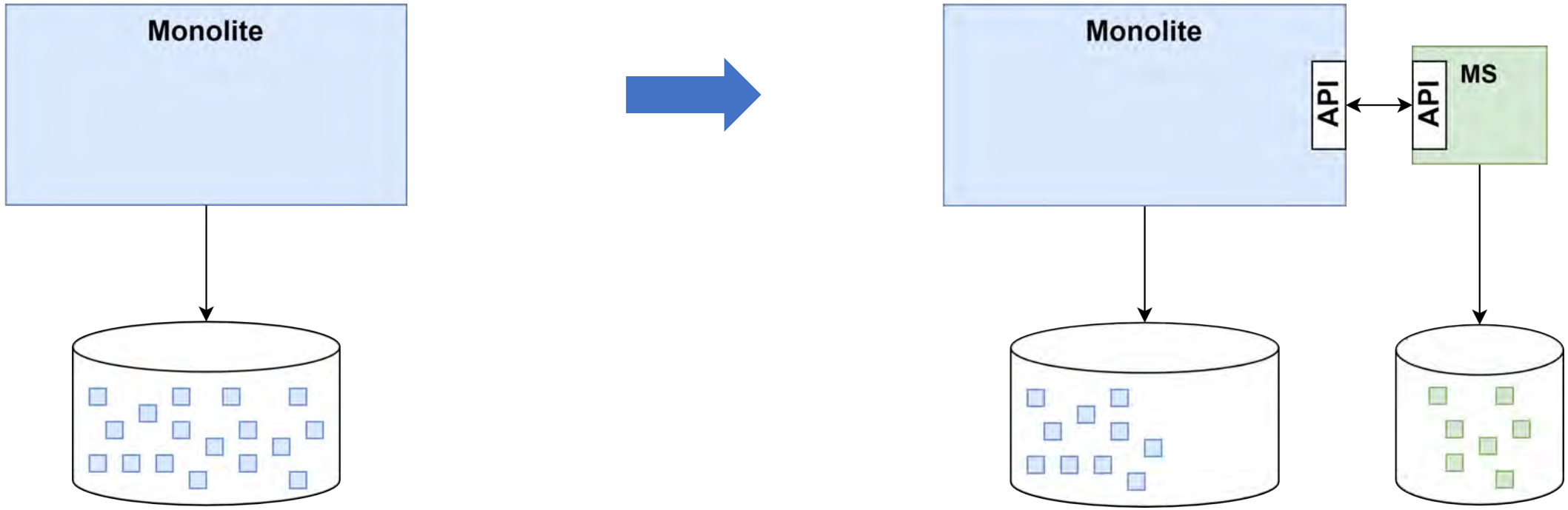
Headless & API date

SESSIONE IT STRATEGY

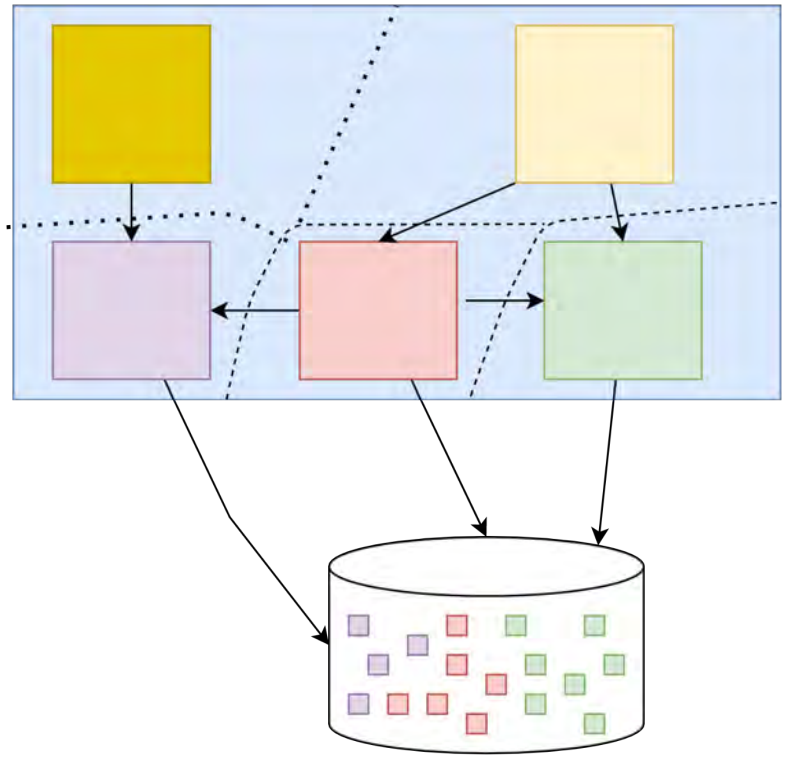
Transizione da una architettura monolitica a microservizi

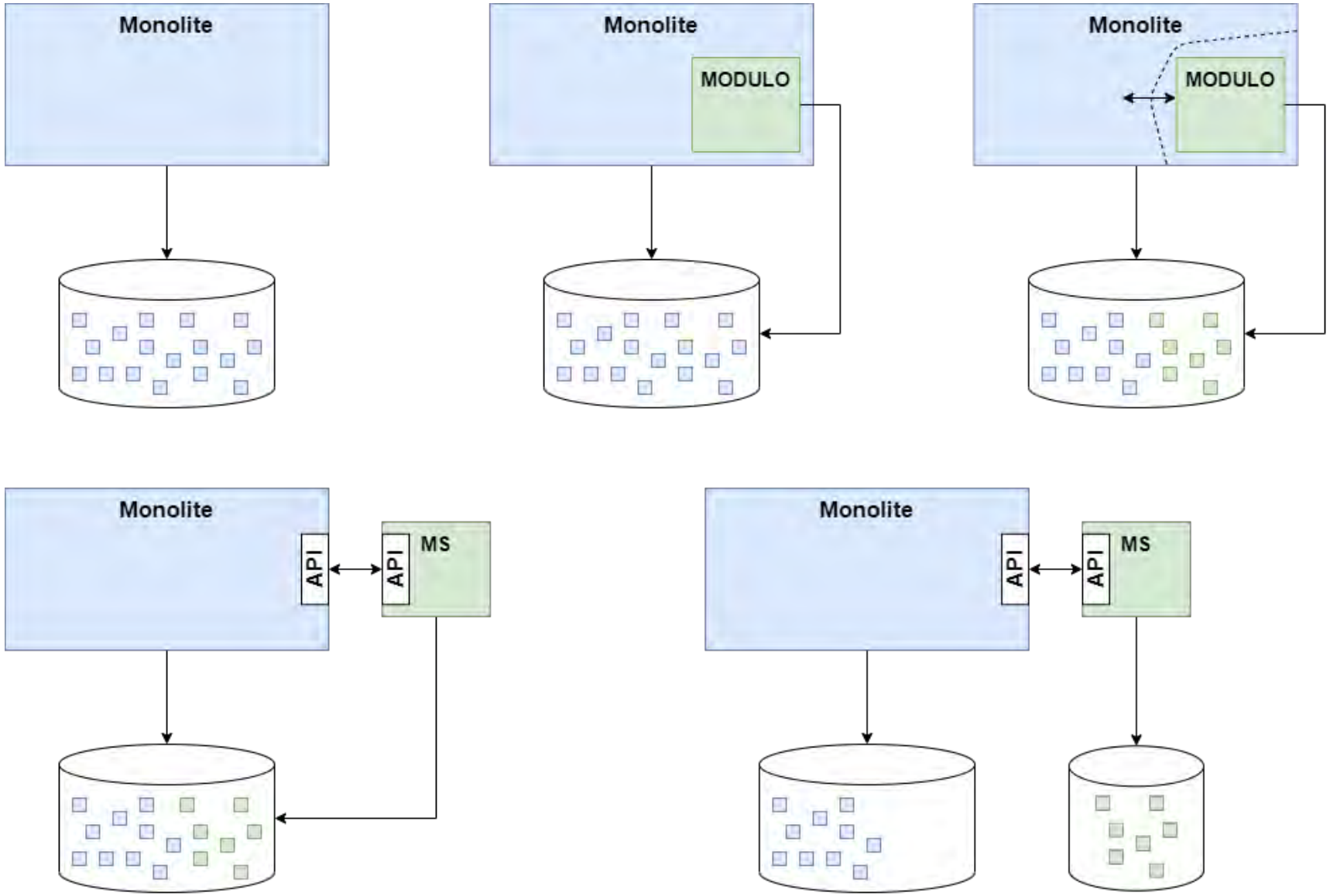
Intesys.it

Obiettivo



First step: analisi dipendenze e refactoring




Monolite
Monolite
MODULO
Monolite
MODULO
Monolite
API
API
MS
Monolite
API
API
MS

Considerazioni

Transizione richiede tempo

L’analisi, le operazioni di refactoring e il test.

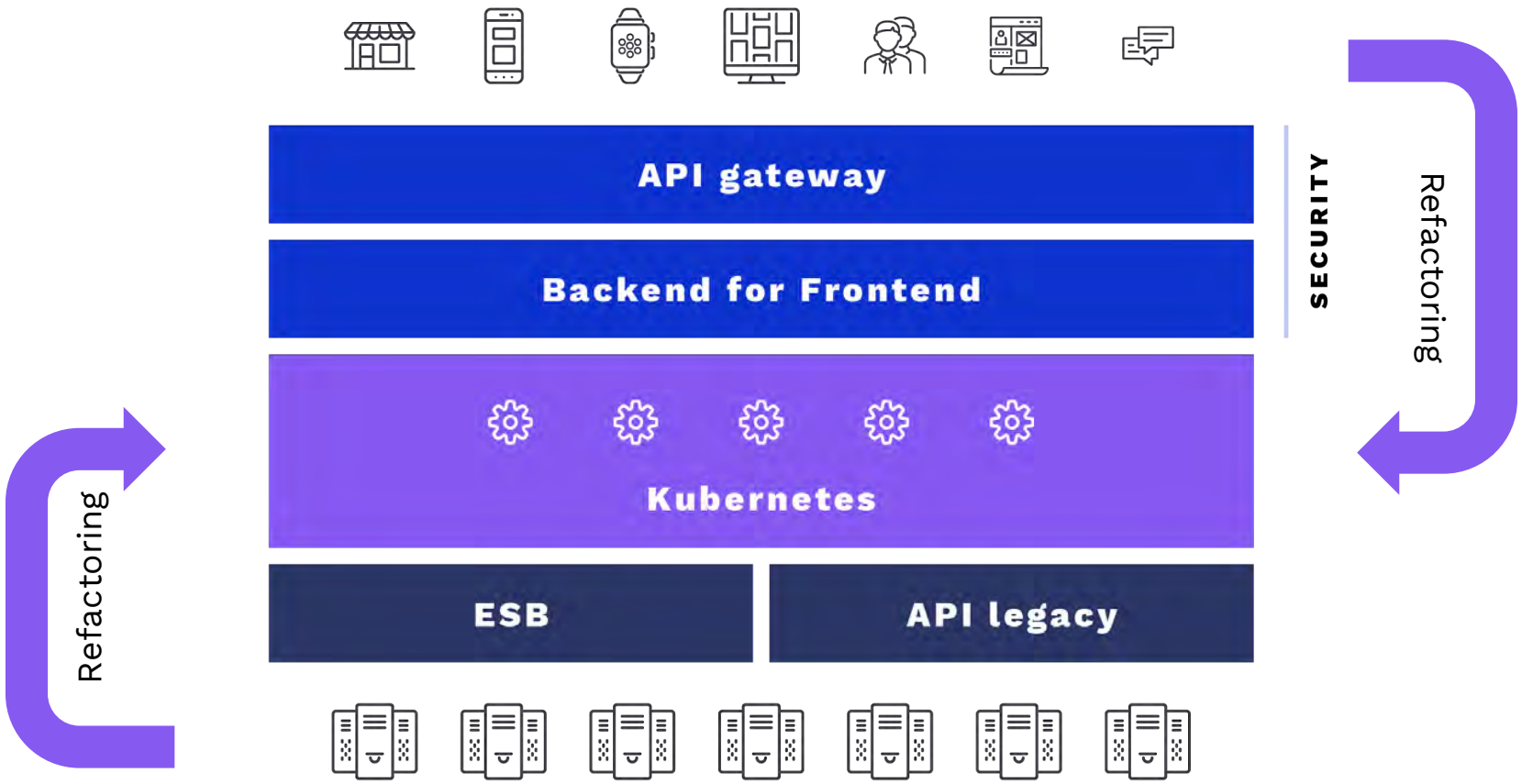
Transizione può essere fatta per gradi

È possibile procedere con un servizio alla volta.

L’alternativa è la riscrittura globale

Revisione globale della intera applicazione.

In questo scenario vanno portati tutti i dati di produzione sul nuovo sistema.


API gateway
Backend for Frontend
Kubernetes
ESB
API legacy
SECURITY
Refactoring
Refactoring

Event-driven Architecture

Cosa è una Event-driven Architecture

Event-driven Architecture

Modello di progettazione dove l'acquisizione, la comunicazione, l'elaborazione e la persistenza degli eventi costituiscono i pilastri portanti della soluzione

Concetto di evento

Avvenimento, o variazione significativa nello stato del nostro sistema.

Concetto di Producer

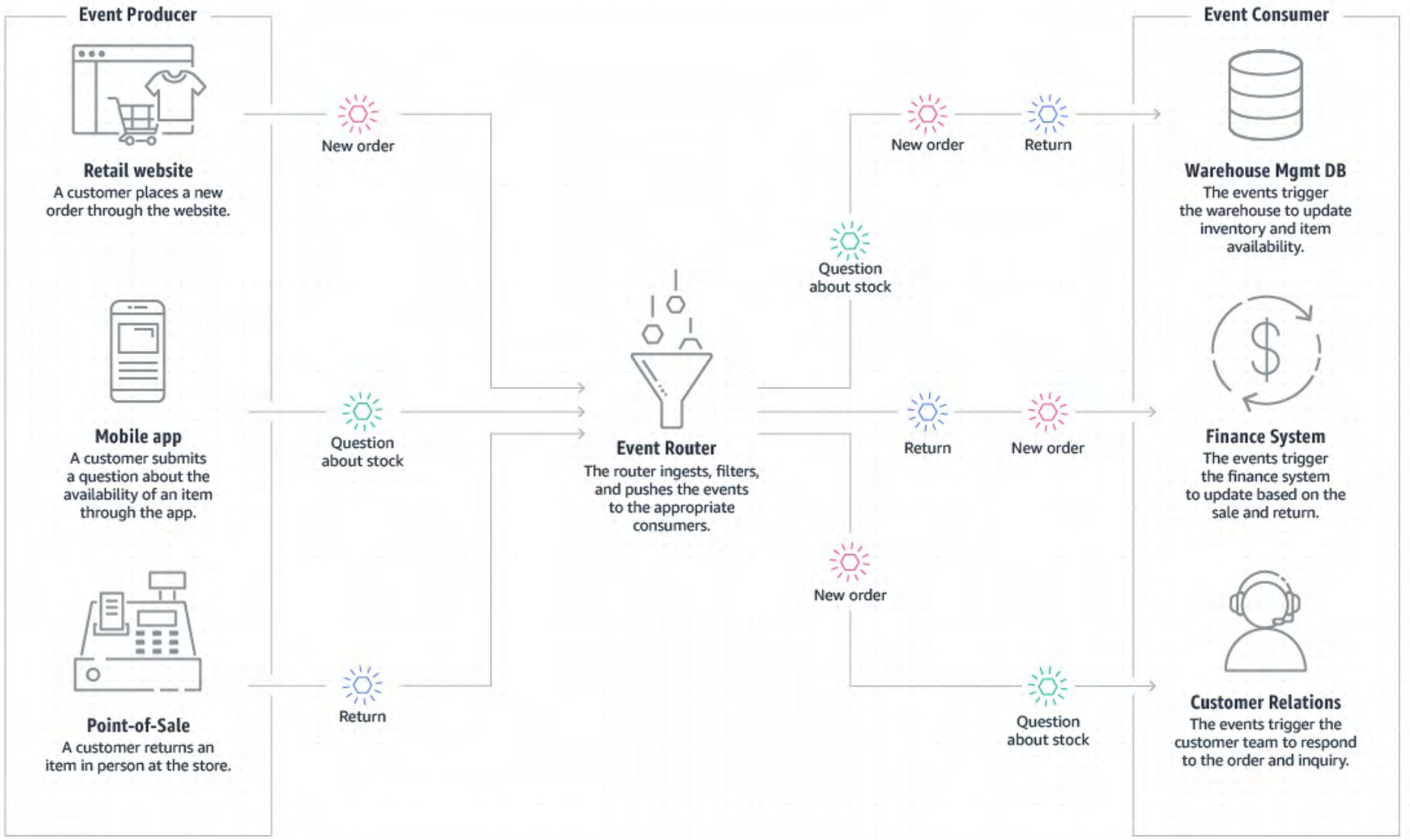
Rileva o percepisce un evento e lo rappresenta come messaggio.

Non conosce il consumer né l'esito dell'evento stesso

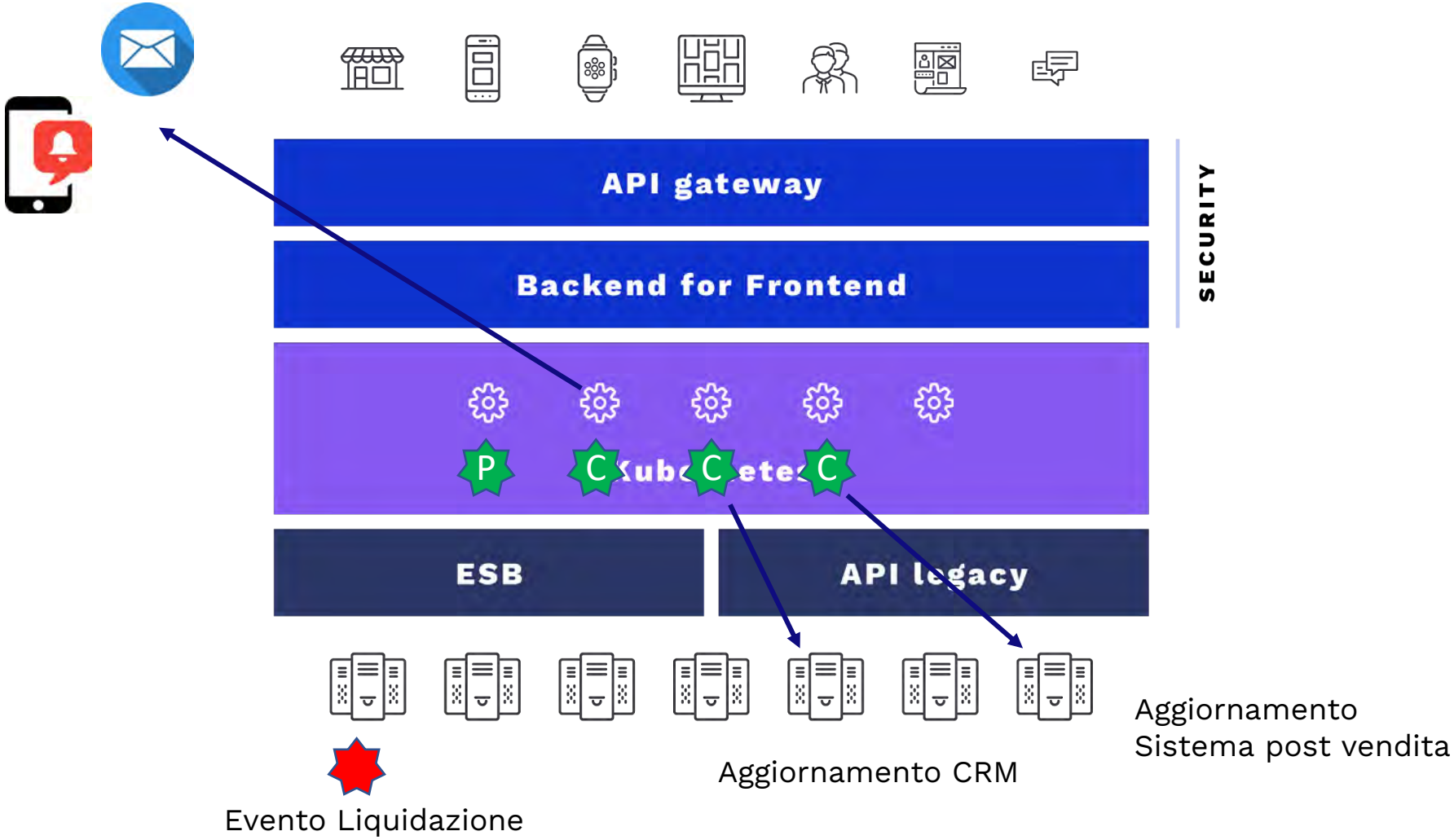
Concetto di Consumer

Elabora l'evento e ne produce i risultati.

Non conosce il Producer



Fonte: https://aws.amazon.com/it/event-driven-architecture/


API gateway
Backend for Frontend
SECURITY
P
C
C
C
ESB
API legacy
Evento Liquidazione
Aggiornamento CRM
Aggiornamento
Sistema post vendita

Event driven architecture - considerazioni

Separazione tra Producer e Consumer

Disaccoppiamento logico e fisico.

Elaborazione asincrona e resiliente

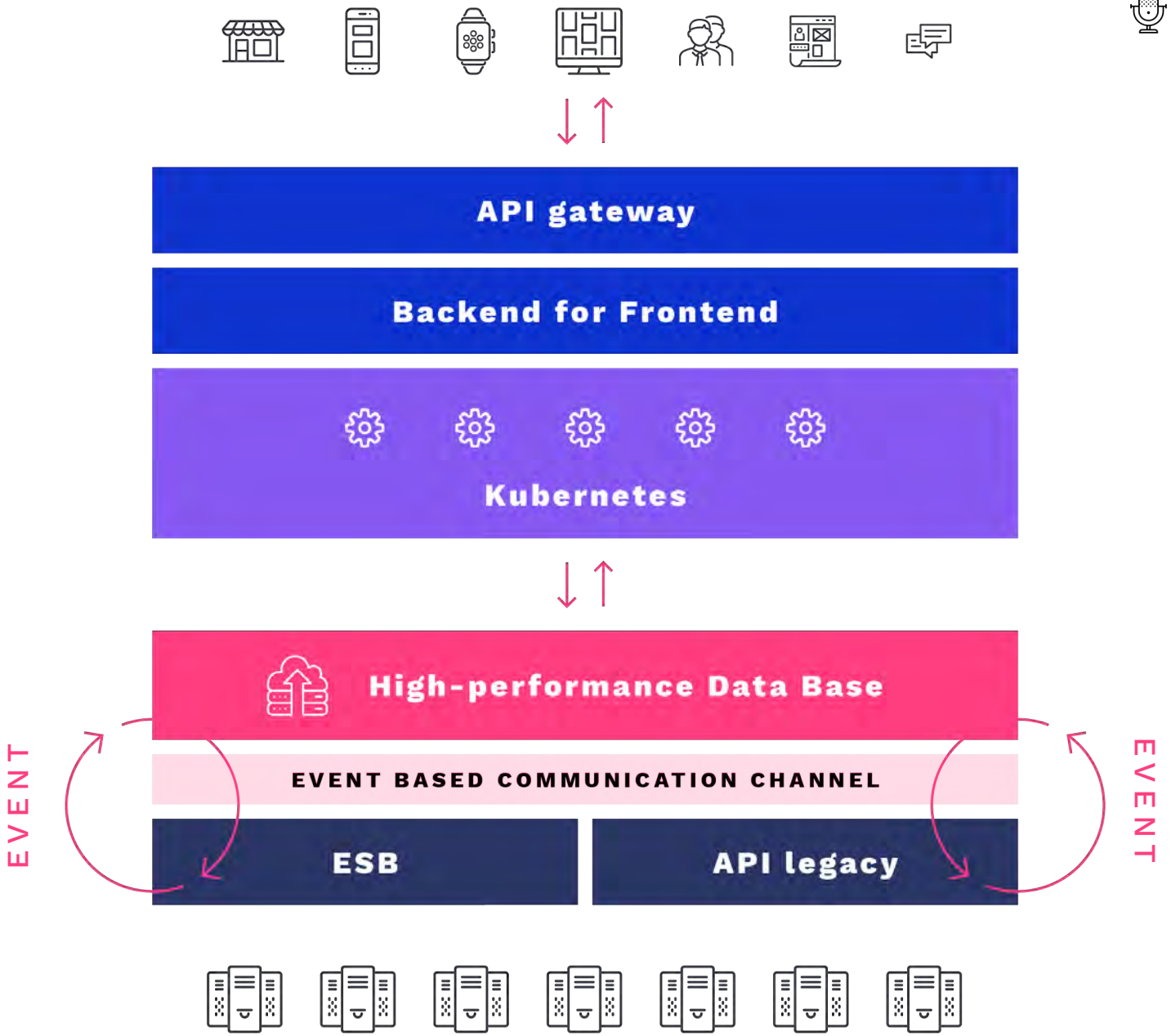
Il malfunzionamento di una applicazione non compromette il funzionamento generale.

Favorisce parallelismo tramite architetture distribuite e microservizi

Aumentando le performance e la possibilità di revisione architetturale.

Possibile agire per step incrementali

Conclusioni


API gateway
Backend for Frontend
Kubernetes
High-performance Data Base
EVENT BASED COMMUNICATION CHANNEL
ESB
API legacy
EVENT
EVENT

Headless & API date

Liferay | mia Platform | Intesys

Grazie

Paolo Quaglia

API STRATEGIST & IT EXPERT, INTESYS

paolo.quaglia@intesys.it