**📜 Manifesto Asgard.Abstraction**

**🎯 Obiettivo**

Definire un contratto agnostico per la pubblicazione e la ricezione di eventi, compatibile con diversi broker (RabbitMQ, Kafka, ecc.), senza introdurre concetti implementativi specifici.

**✅ Principi Fondanti**

**1. Broker-agnosticità**

* Nessun riferimento a concetti implementativi: exchange, routingKey, partitionKey, deliveryTag, topic, queue, consumer group.
* I messaggi devono essere trattati come **eventi logici**, non come comandi broker-specifici.

**2. Tipizzazione sicura**

* Il payload è un tipo serializzabile (POCO).
* L'involucro formale è sempre un CloudEvent, conforme allo standard [CloudEvents 1.0].

**3. Opzionalità tramite oggetto dedicato**

* Parametri come type, source, subject vanno incapsulati in una classe CloudEventOptions.
* Vietati overload con parametri scalari sparsi: **una sola entry point semantica**.

**4. Separazione dei contesti**

* Gli adapter (Asgard.RabbitMQ, Asgard.Kafka, ecc.) sono responsabili della mappatura tra CloudEvent e la topologia o semantica del broker.
* L’interfaccia dell’abstraction **non fa alcuna assunzione** su come questi dati vengano trasportati o serializzati.

**5. Estendibilità controllata**

* Ogni estensione va isolata nel proprio pacchetto implementativo (Asgard.RabbitMQ, ecc.).
* Campi opzionali o metadati si collocano nel dizionario Extensions del CloudEvent.

**👮‍♂️ Regole di validazione**

| **Tipo di violazione** | **Azione** |
| --- | --- |
| Parametri broker-specifici nell’abstraction | ❌ Bloccato |
| Retry/Nack nella parte astratta | ❌ Non ammesso |
| Naming orientato a topic, queue, ecc. | ❌ Naming errato |

**📁 Output attesi**

L’abstraction deve essere sufficiente a:

* Pubblicare e leggere eventi con semantica coerente.
* Essere compatibile con librerie implementative multi-broker.
* Garantire solidità delle interfacce anche in scenari enterprise distribuiti.