scm
woodworking technology

IoT day

CQRS and IoT: A match made In Heaven

28/11/2019

Who Am I?

Carmine Ingaldi

Digital Services Team @ SCM Group SpA

Software Engineer
Interested in IoT, Microservice Architectures, DevOps, Big Data
Agile Believer
Bass Player & Former DJ
Neapolitan

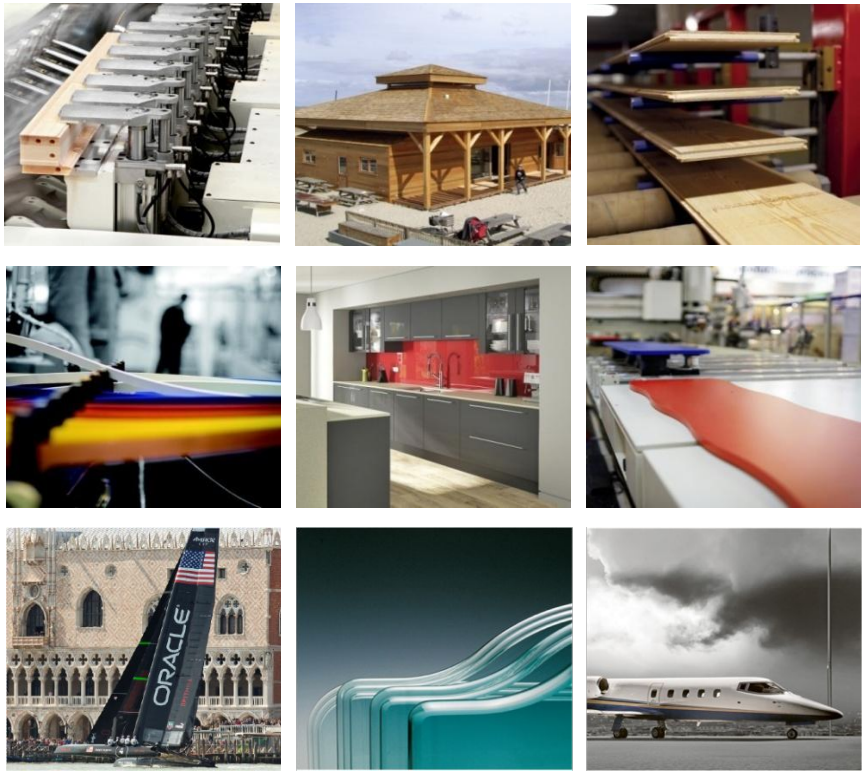
www.linkedin.com/in/carmine-ingaldi

https://github.com/cingaldi

scm
woodworking technology

What is SCM Group?

A technology world leader in processing a wide variety of materials: wood, plastic, glass, stone, metal and composites.



The Group companies, operating throughout the world, are reliable partners of leading companies in various market sectors, including:

Furniture

Construction

Automotive

Aerospace

Ship-building

Plastic processing industries

What is SCM Group?

Facts & Figures

about **700 M€** year revenue

3.600 people in Italy and other Countries

4 main production sites

5 continents selling and servicing

7% revenue in R&D investments

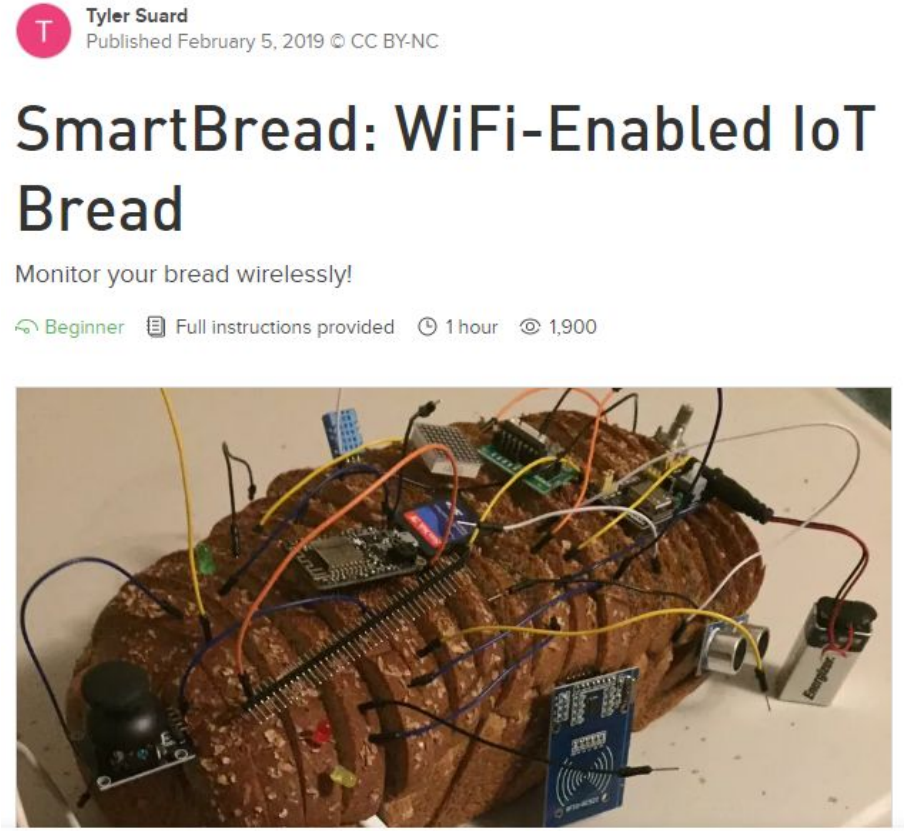
WE ARE HIRING!!

scm
woodworking technology

Abstract

Approccio “Broke”

- Sensorizza il tuo prodotto
- Mostra dati su una web dashboard
- Aspetta il successo
- BONUS: “predictive” a caso



Approccio “Woke”

- Colleziona informazioni
  - Sui tuoi prodotti in campo
  - Sul comportamento dei tuoi clienti
- Estrai conoscenza
- Migliora il tuo prodotto
- Migliora i tuoi processi
- Migliora l’esperienza d’uso del prodotto

How and Why?

How and Why?

CQRS - Definition

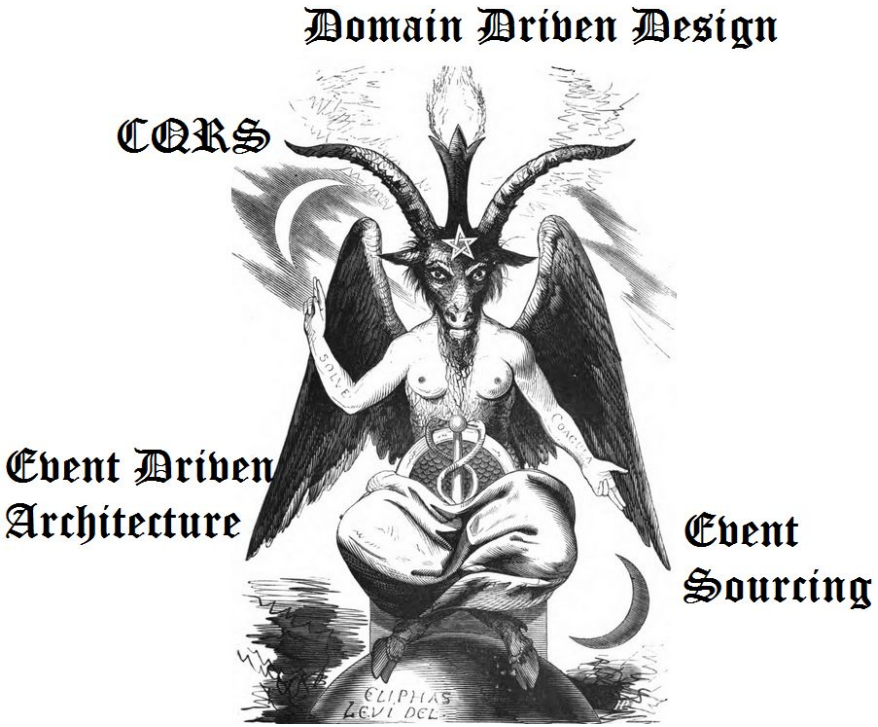
CQRS stands for Command Query Responsibility Segregation. [...]. At its heart is the notion that you can use a different model to update information than the model you use to read information. For some situations, this separation can be valuable, ***but beware that for most systems CQRS adds risky complexity.***

(M. Fowler)

https://martinfowler.com/bliki/CQRS.html

scm
woodworking technology

Is This Evil?


Domain Driven Design
CQRS
Event Driven
Architecture
Event
Sourcing
ELIPHAS
LEVI DEL

CRUD Model

un CRUD model è realizzato a partire dalla definizione di uno schema dei dati e decidendo quali metodi di accesso offrire al client su ogni **risorsa**

Le **mutazioni** e le **osservazioni** dello stato di un'applicazione hanno rappresentazioni tra loro compatibili

CRUD Model

```
POST /users

{
    "name" : "Carmine",
    "surname" : "Ingaldi",
    "email" : "carmine@ingaldi.com"
}
```

```
{
    success: true,
    result: {
        "id" : "123xyz456",
        "name" : "Carmine",
        "surname" : "Ingaldi",
        "email" : "carmine@ingaldi.com"
    }
}
```

```
GET /users/123xyz456

{
    "id" : "123xyz456",
    "name" : "Carmine",
    "surname" : "Ingaldi",
    "email" : "carmine@ingaldi.com"
}
```

scm
woodworking technology

CRUD Model

Nell’ IoT gli utenti sono umani e devices….parlano lingue differenti

Da CRUD a CQRS

- **C**reate
- **R**ead
- **U**pdate
- **D**elete

- **C**ommand
- **Q**uery
- **R**esponsibility
- **S**egregation

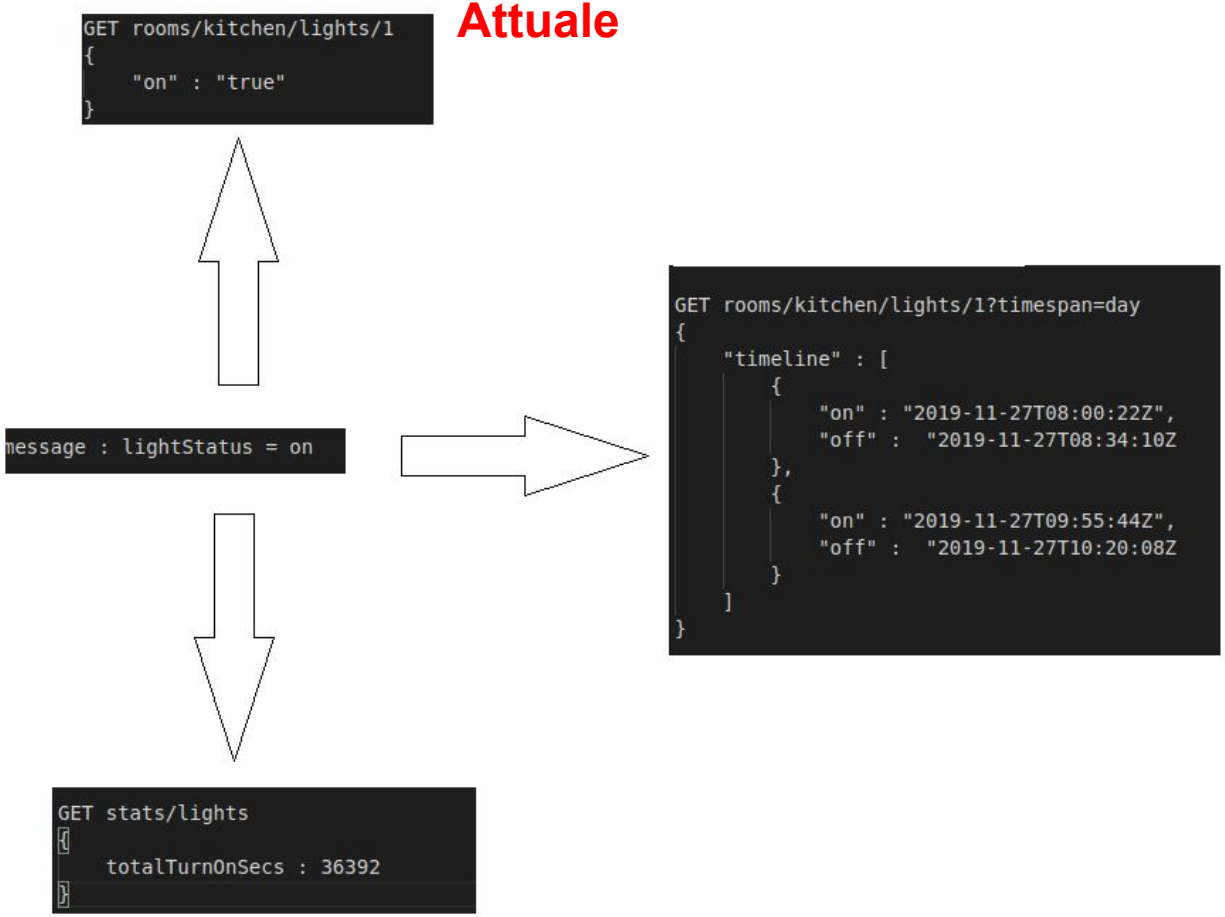
CQRS Model

Il dominio viene rappresentato ancora attraverso risorse intercorrelate, ma l'azione del client sullo stato del sistema è esplicitamente espressa in termini di mutazioni ed osservazioni dello stato

una mutazione -> più rappresentazioni

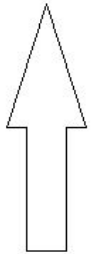
CQRS Model

Attuale

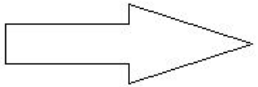


CQRS Model

```
GET rooms/kitchen/lights/1
{
    "on" : "true"
}
```

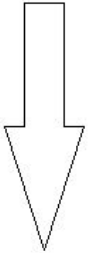


```
message : lightStatus = on
```



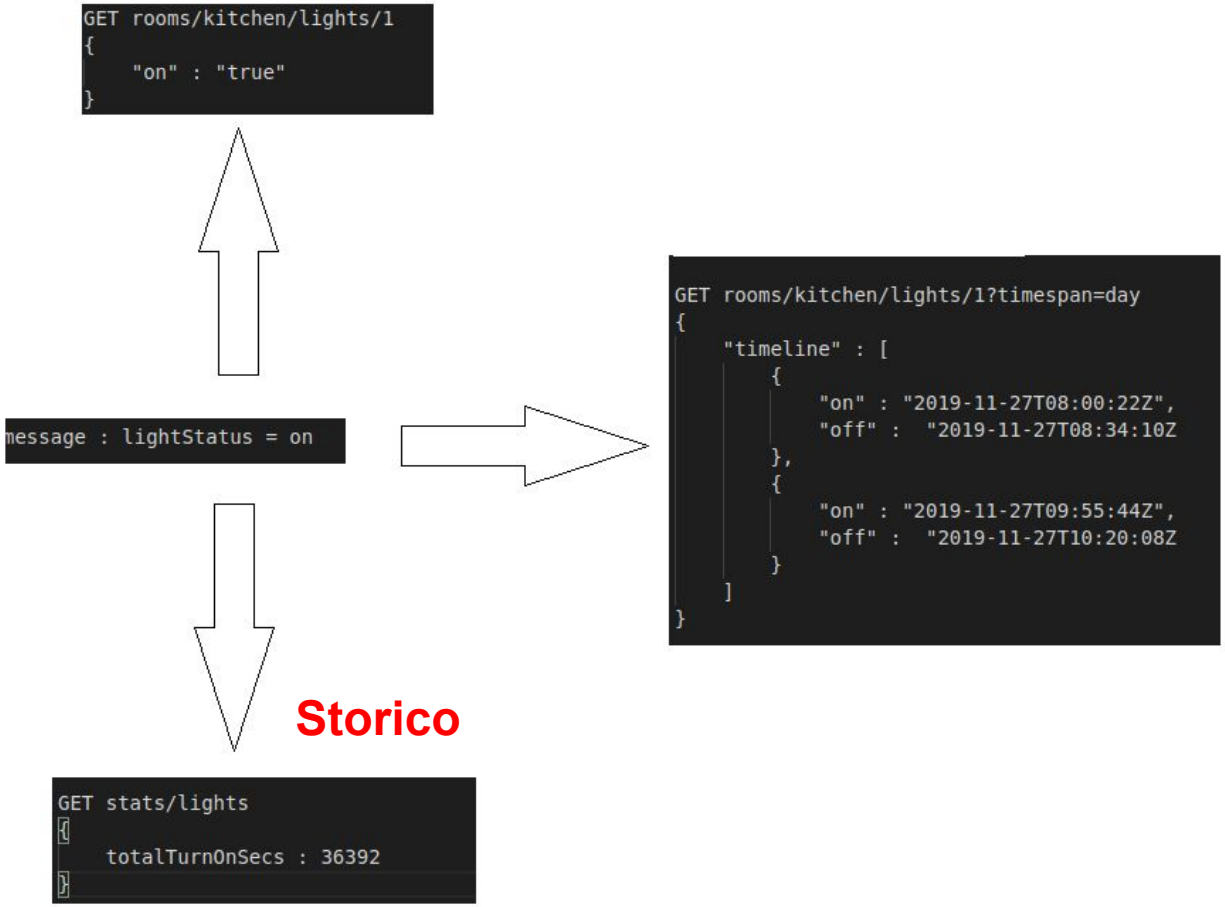
Trend

```
GET rooms/kitchen/lights/1?timespan=day
{
    "timeline" : [
        {
            "on" : "2019-11-27T08:00:22Z",
            "off" :  "2019-11-27T08:34:10Z
        },
        {
            "on" : "2019-11-27T09:55:44Z",
            "off" :  "2019-11-27T10:20:08Z
        }
    ]
}
```



```
GET stats/lights
{
    totalTurnOnSecs : 36392
}
```

CQRS Model


GET rooms/kitchen/lights/1
{
"on" : "true"
}
message : lightStatus = on
GET rooms/kitchen/lights/1?timespan=day
{
"timeline" : [
{
"on" : "2019-11-27T08:00:22Z",
"off" : "2019-11-27T08:34:10Z
},
{
"on" : "2019-11-27T09:55:44Z",
"off" : "2019-11-27T10:20:08Z
}
]
}
Storico
GET stats/lights
{
totalTurnOnSecs : 36392
}

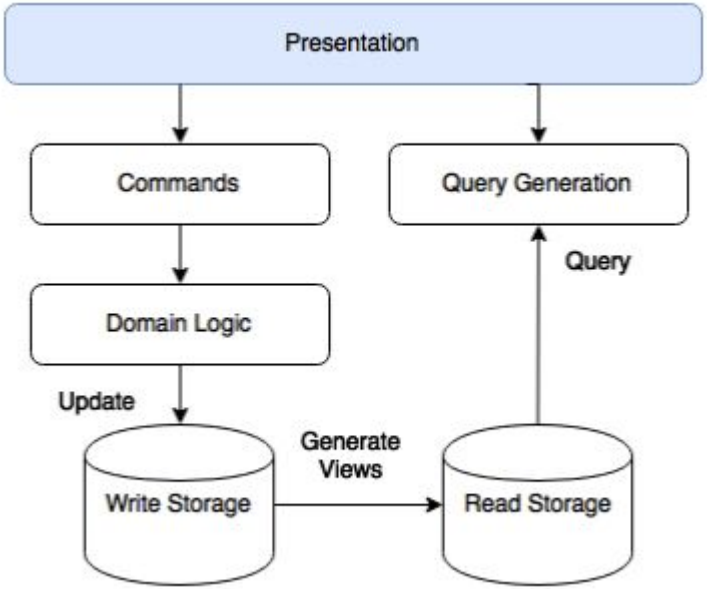
CQRS Model

Progettare l'applicazione in due tipi di componenti indipendenti:

- Componente di aggiornamento dello stato (write/command side)
- Componente di osservazione dello stato (read/query side)

Per ogni command side possono esistere più read sides (proiezioni)

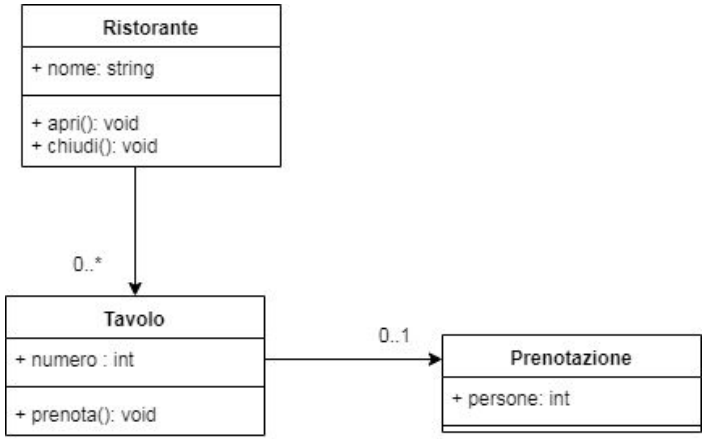
Il dominio è rappresentato in modi diversi a partire dallo stesso modello



Key Concepts

Domain Driven Design

Rappresentare la conoscenza del dominio di business attraverso un linguaggio condiviso che descrive concetti e comportamenti



Aggregates (CRUD-ish)

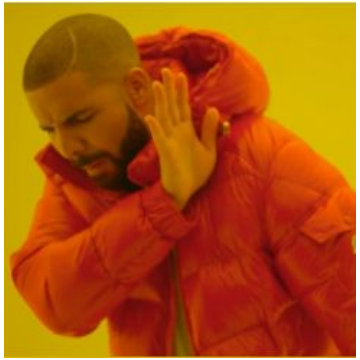
```
def addBlogPost( userId , title , text):

    user = userRepository.find(userId)

    posts = user.getPosts()
    posts.append(Post(title , text))
    user.setPosts(posts)

    userRepository.save(user)
```

Aggregates (CRUD-ish)



```
def addBlogPost( userId , title , text):

    user = userRepository.find(userId)

    posts = user.getPosts()
    posts.append(Post(title , text))
    user.setPosts(posts)

    userRepository.save(user)
```

Aggregates (CRUD-ish)

```
def addBlogPost( userId , title , text):

    user = userRepository.find(userId)

    user.addPost(title , text)

    userRepository.save(user)
```

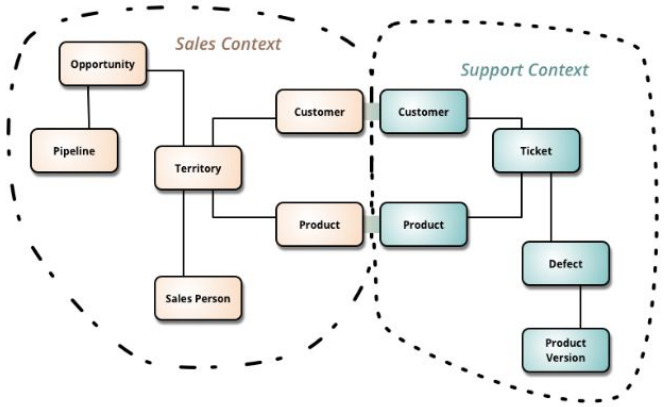
scm
woodworking technology

Aggregates

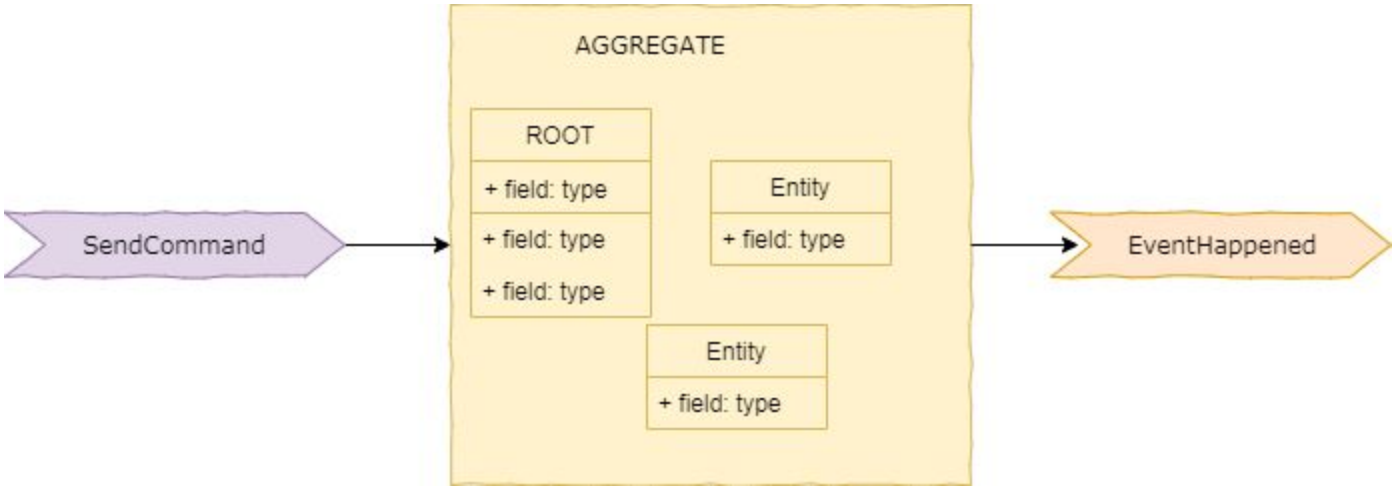
Le informazioni in un aggregato sono sempre consistenti.

La *aggregate root* è il concetto principale da cui dipendono gli altri. Cambiare lo stato dell'aggregato vuol dire effettuare un'operazione (**comando**) sulla aggregate root

Se l'aggregato diventa "distribuito" dobbiamo fare di più per garantire la consistenza!



From Command to Event



Un evento è la notifica che È AVVENUTO un cambiamento di stato in un sistema

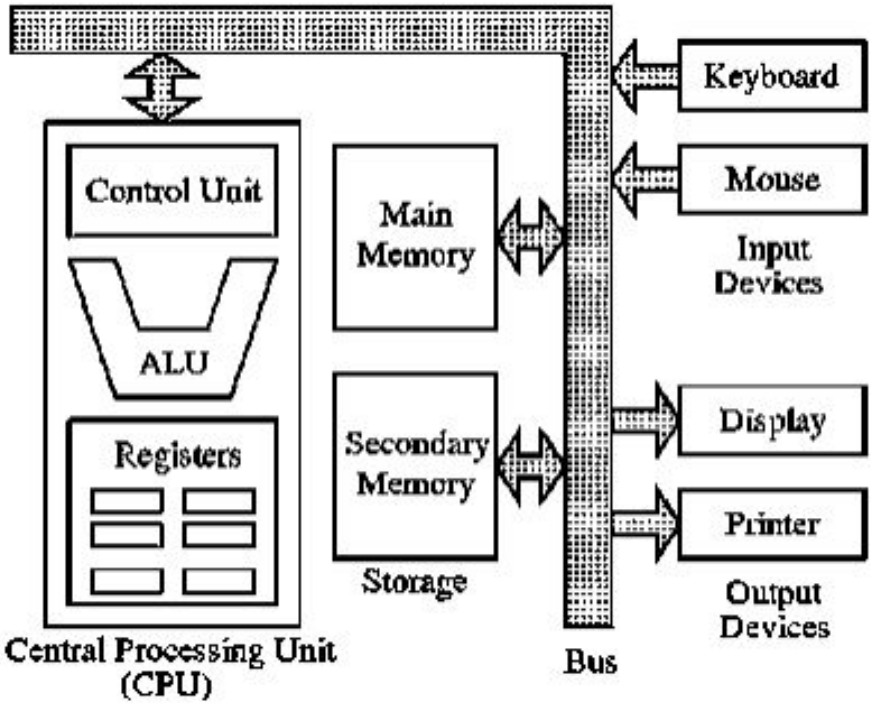
Event Driven Architecture

Progettare un’applicazione i cui componenti interagiscono tra loro non soddisfando richieste, ma reagendo ad eventi

Event Driven or Message Driven?

Un’ applicazione IoT ha una natura Event Driven: La piattaforma raccoglie eventi nel mondo fisico ed orchestra servizi per fornire feedback e compiere azioni sul mondo fisico

You're Already into EDA



Event Driven or Message Driven?

Vantaggi

- Disaccoppiamento
- Sistemi Resilienti
- Consistenza
- Backpressure

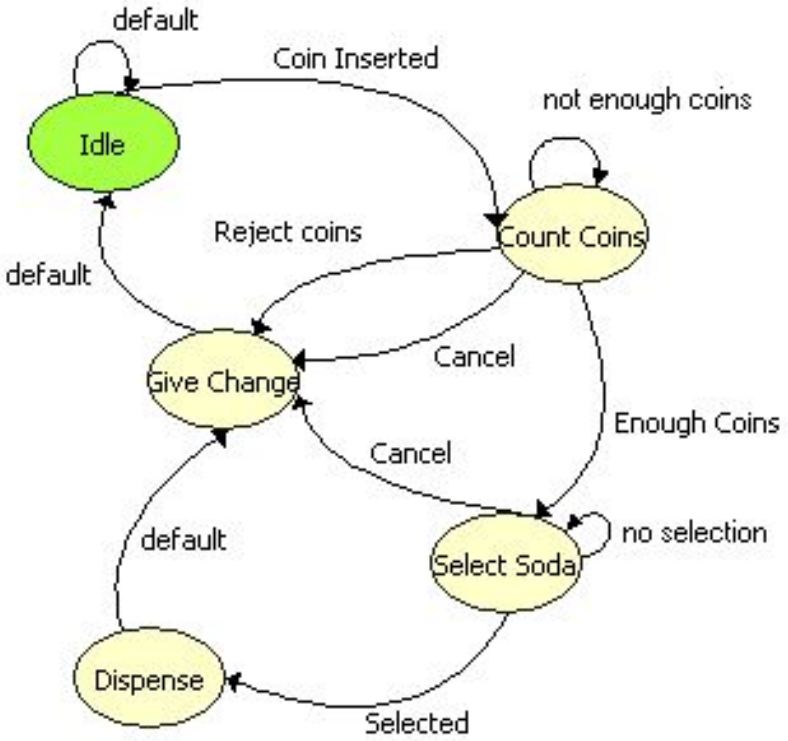
Svantaggi

- Spaghetti Flow
- “Single” Point of Failure
- Consistenza eventually
- Memory Intensive

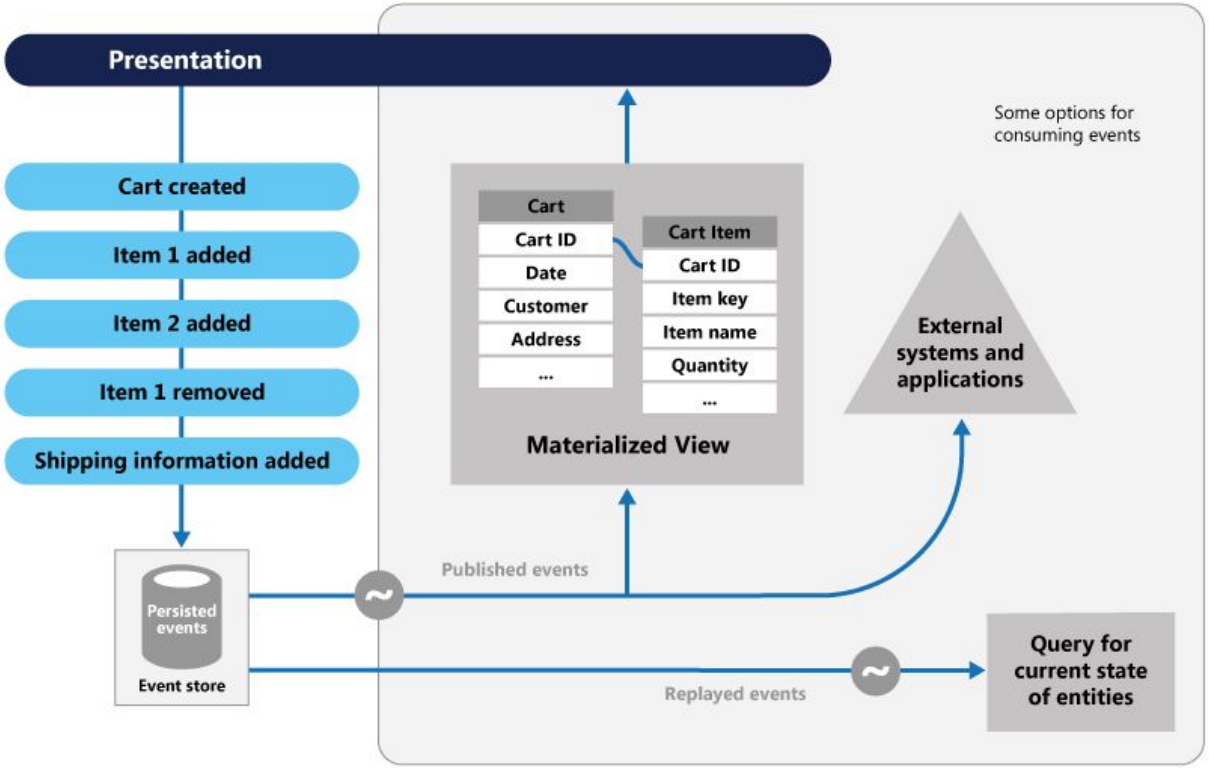
From Event Driven to Event Sourcing

Lo stato attuale di un'applicazione è il risultato di una sequenza di eventi

E se l'applicazione persistesse una sequenza di eventi anziché uno stato?



From Event Driven to Event Sourcing



https://docs.microsoft.com/it-it/azure/architecture/patterns/event-sourcing

Risks and Benefits of ES

Risks and Benefits of ES

Vantaggi

- Applicazioni Evolutive
- Audit Logging
- Consistenza
- Single Source of Truth
- Testabilità

Svantaggi

- Il DB cresce nel tempo
- Approccio ad-hoc per schema-changes
- Difficile comprendere lo stato delle informazioni persistite

Interacting With the Outside

Interacting With the Outside

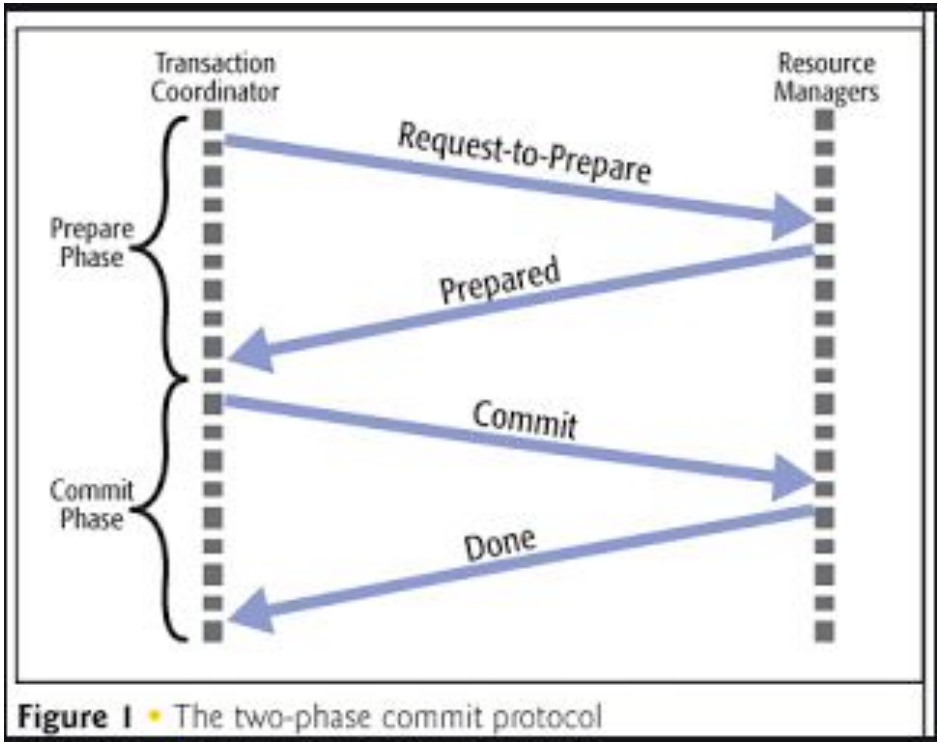


Figure 1 • The two-phase commit protocol

Interacting With the Outside

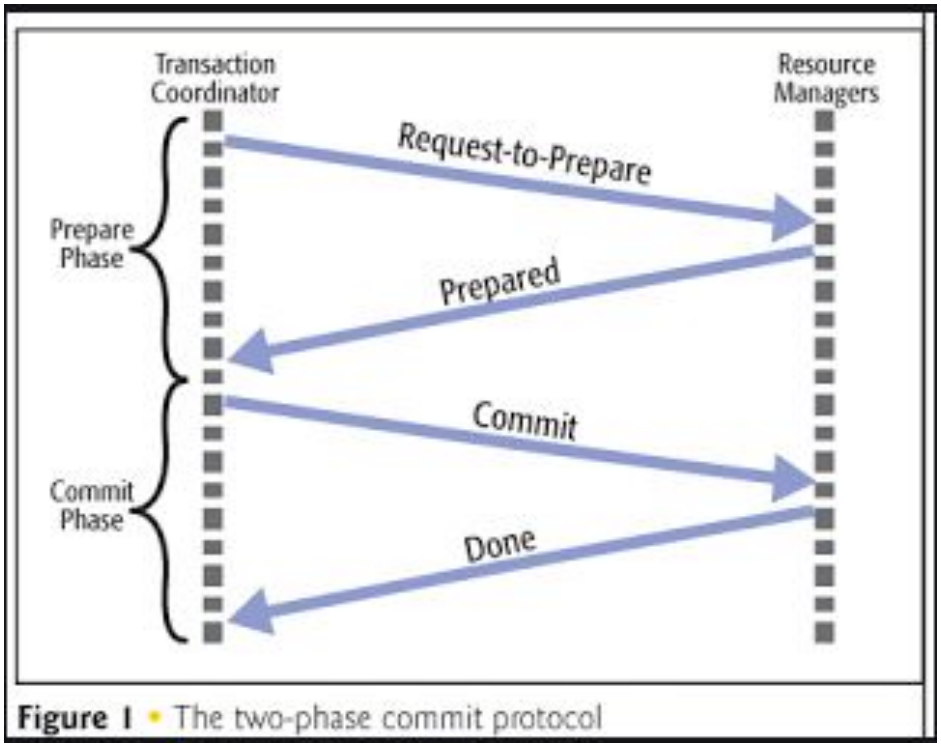
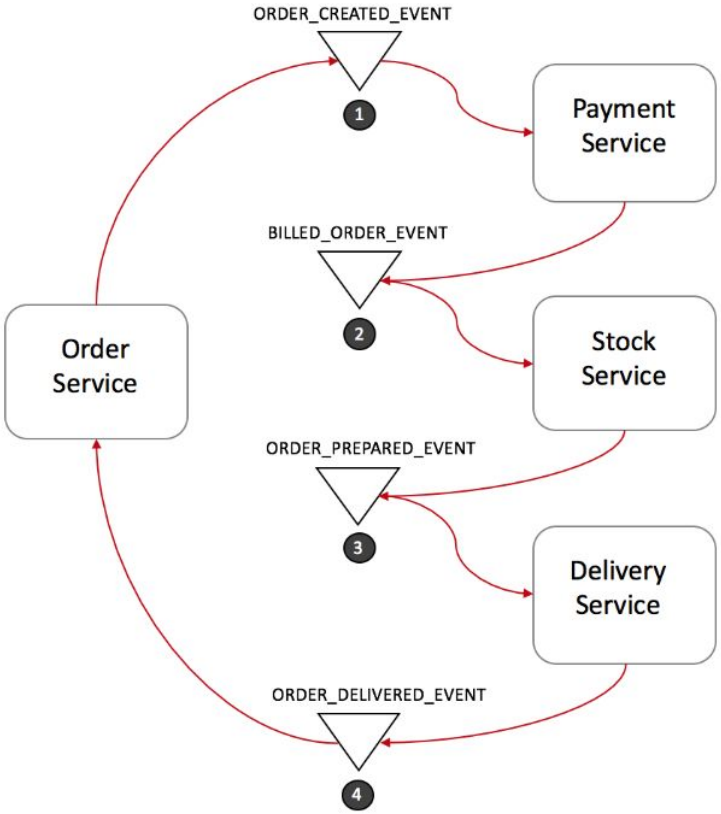
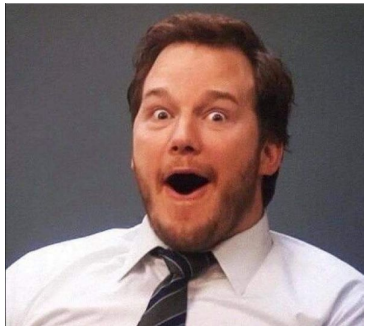


Figure 1 • The two-phase commit protocol

Interacting With the Outside





Interacting With the Outside

- **A**tomic
- **C**onsistent
- **I**solated
- **D**urable

- **B**asically
- **A**vailable
- **S**oft state
- **E**ventually consistent

Sagas / Event Processors

Process Manager : Gestisce transazioni distribuite complesse garantendo consistenza (eventually) e uno stato persistente

Saga: Gestisce una *Long Term Transaction* non atomica: Per ogni comando esiste un'operazione correttiva che viene applicata in caso di errore

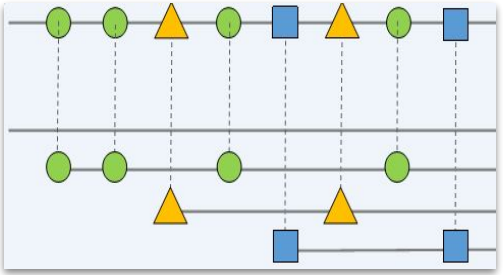
Sagas and Event Processing

Saga

- Object Oriented Design
- Manages processes
- Addresses concurrency

Stream Processing (CEP)

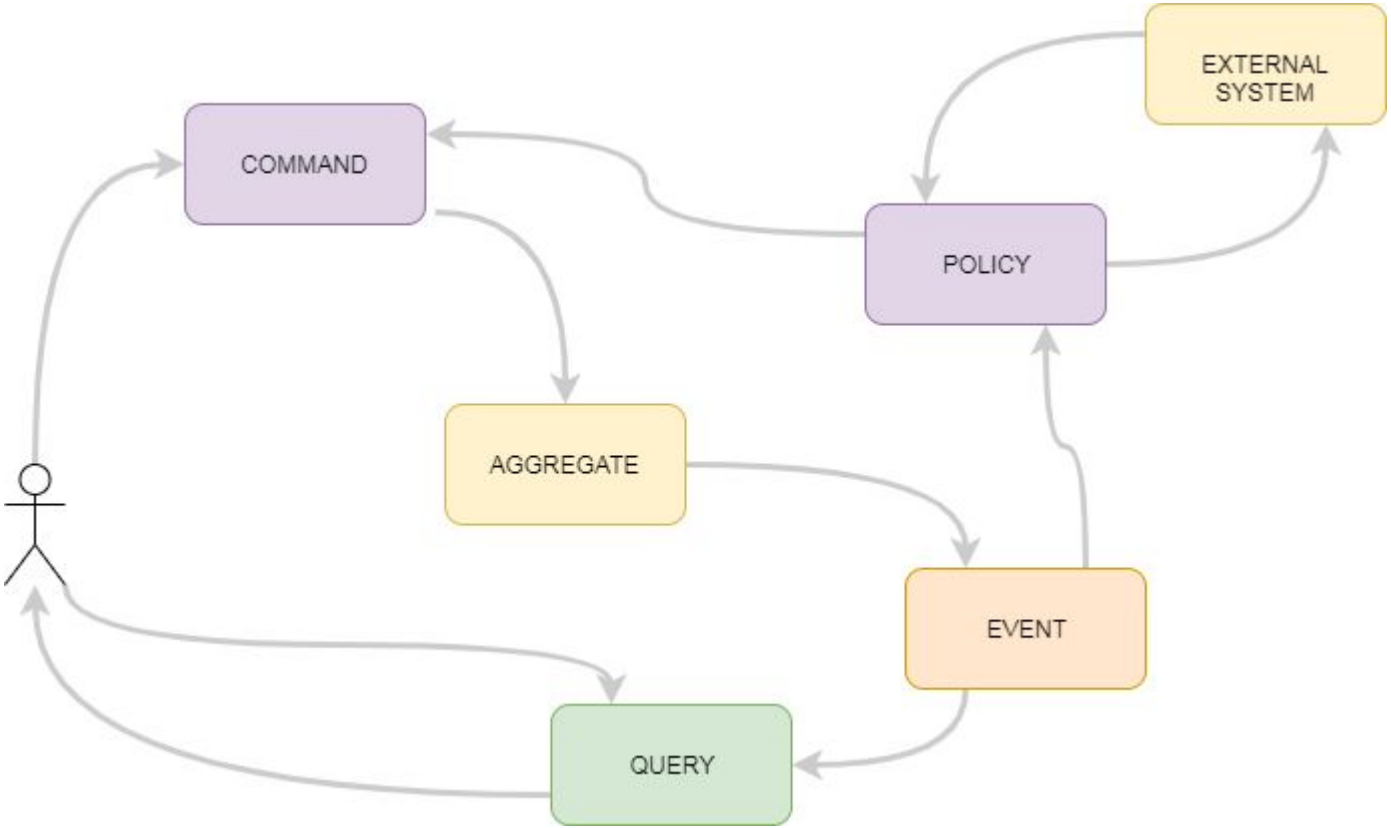
- Functional Definition
- Applies operators
- Addresses parallelism



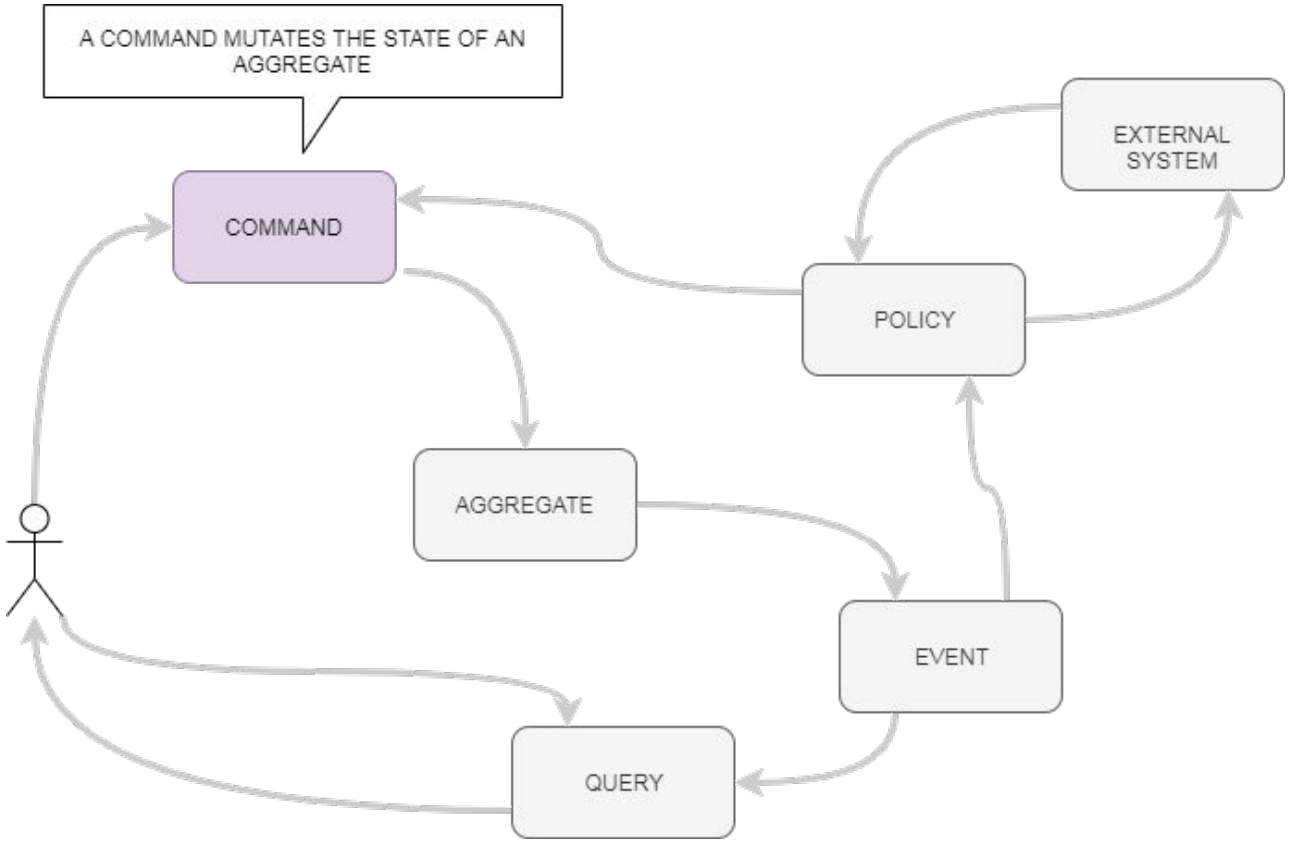
https://www.confluent.io/blog/event-sourcing-cqrs-stream-processing-apache-kafka-whats-connection/

Summing Up!

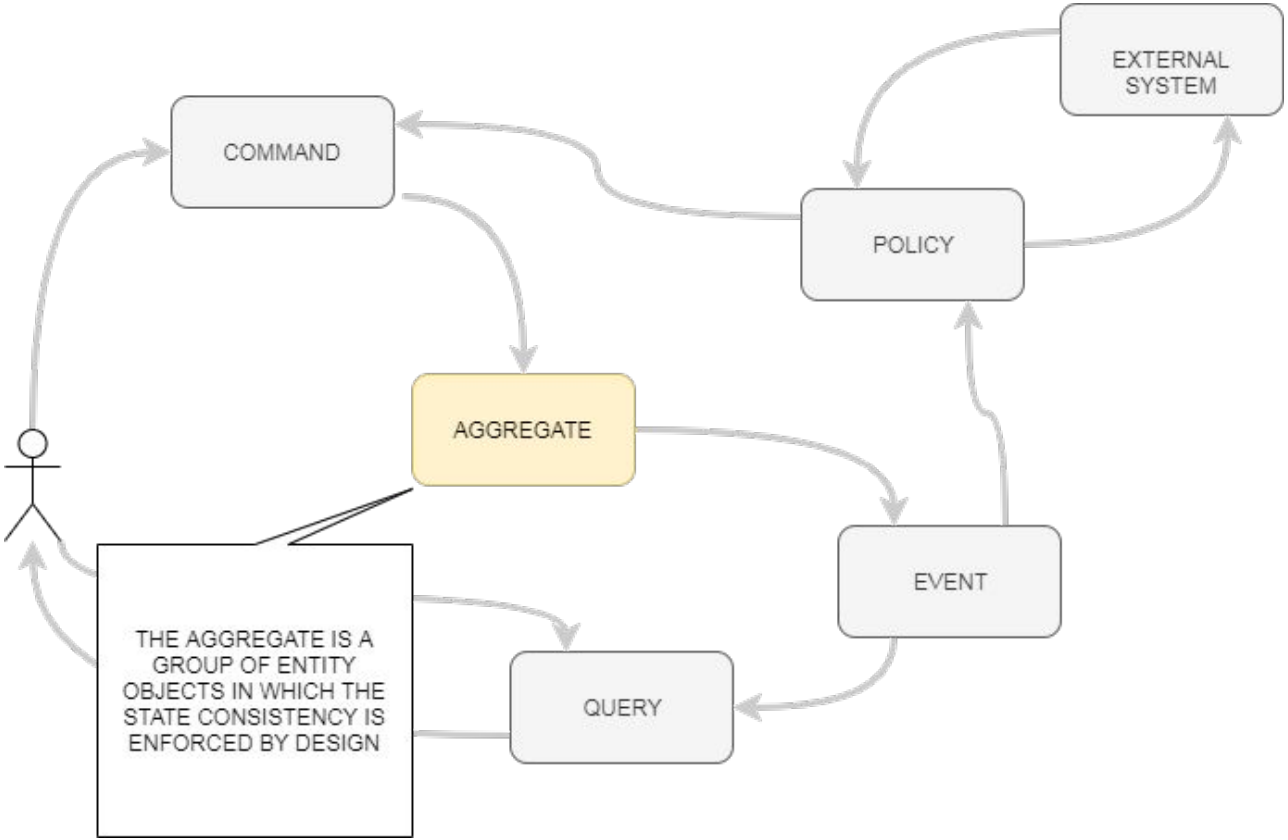
A New Reference Architecture



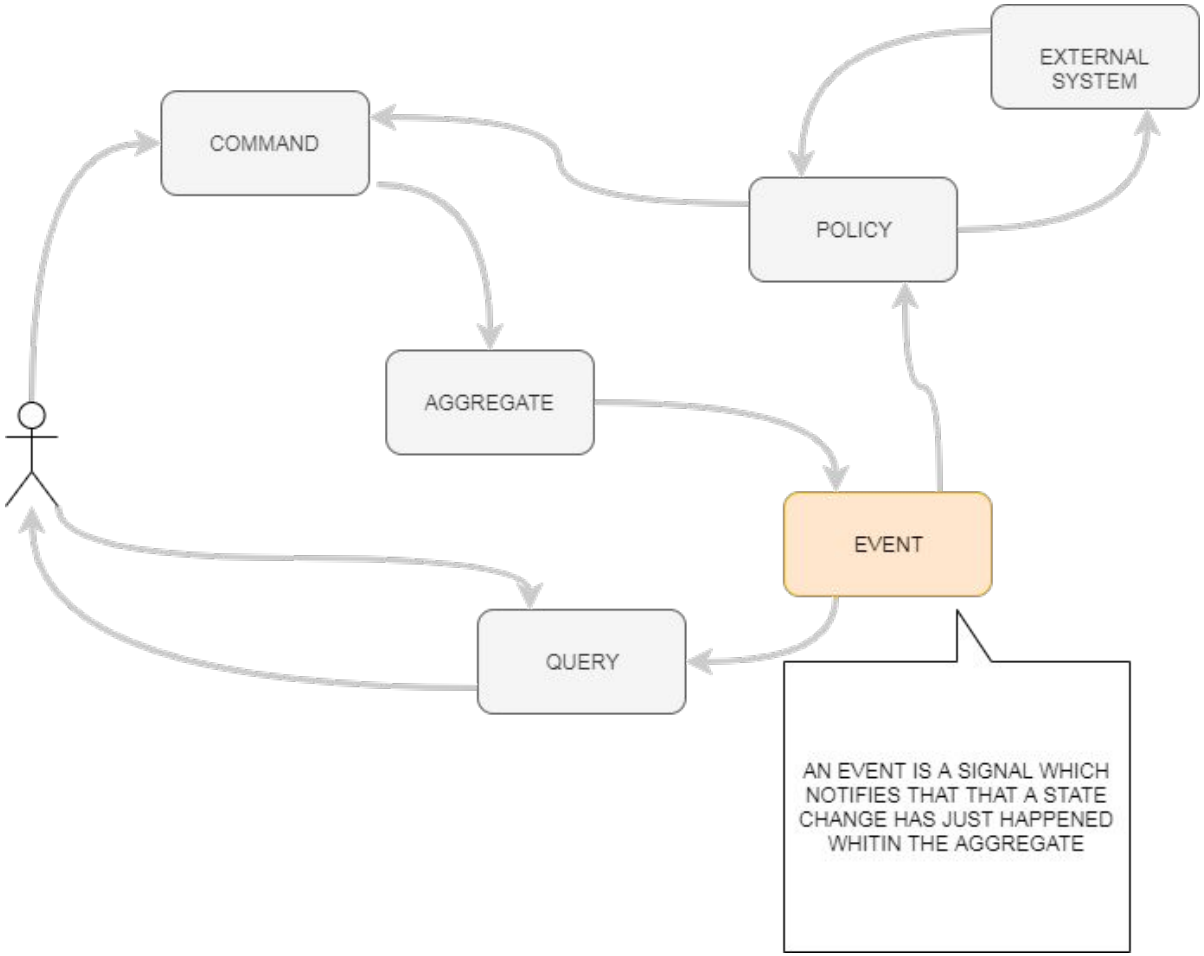
A New Reference Architecture



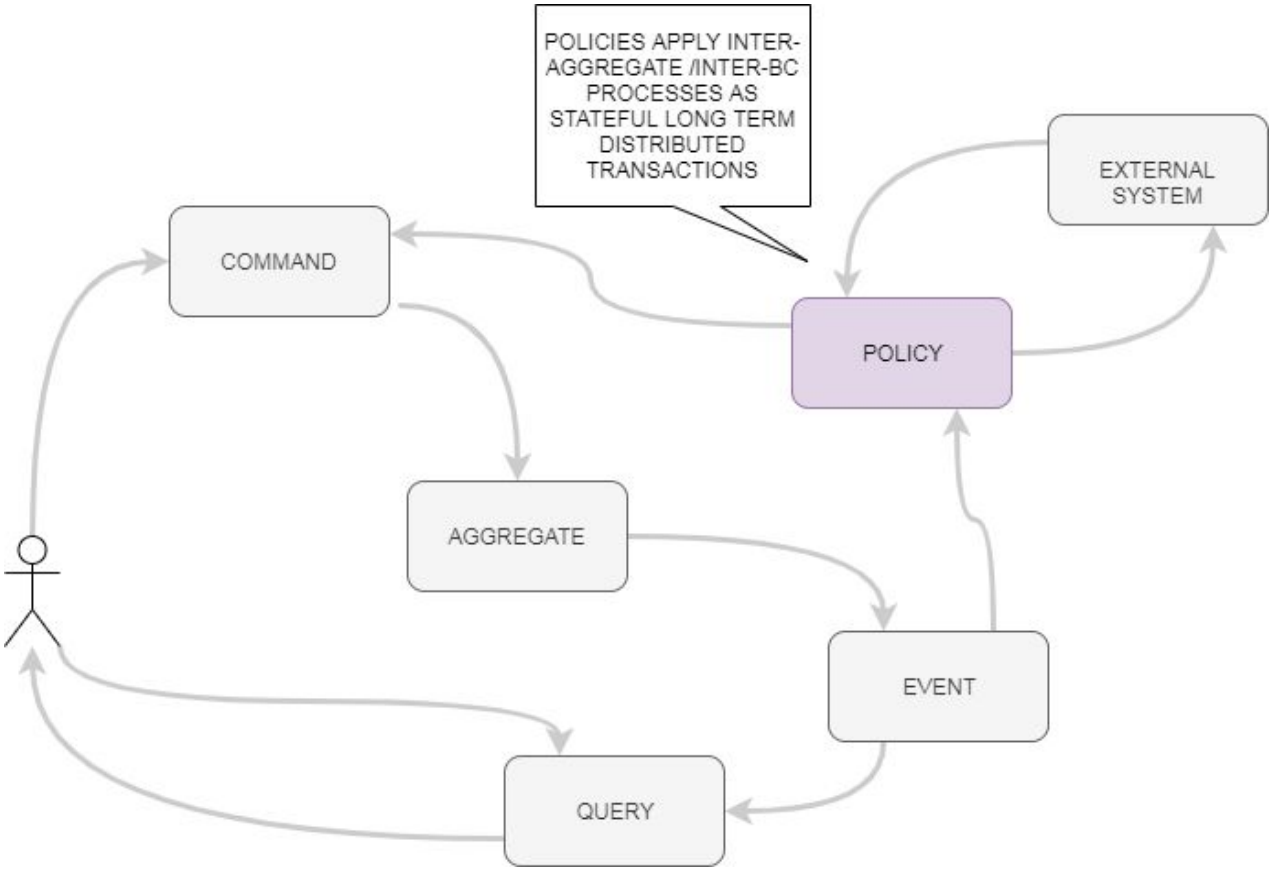
A New Reference Architecture



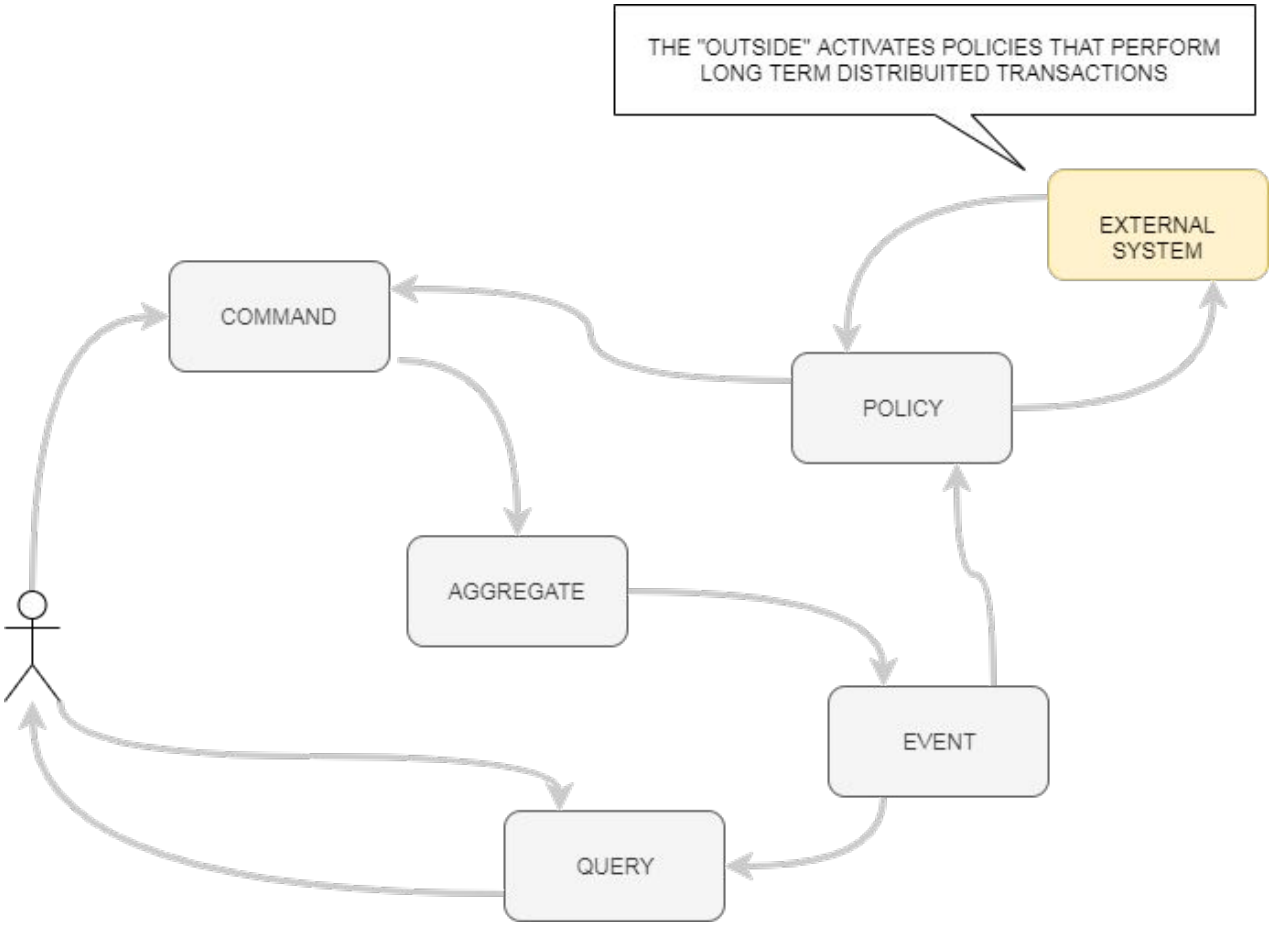
A New Reference Architecture



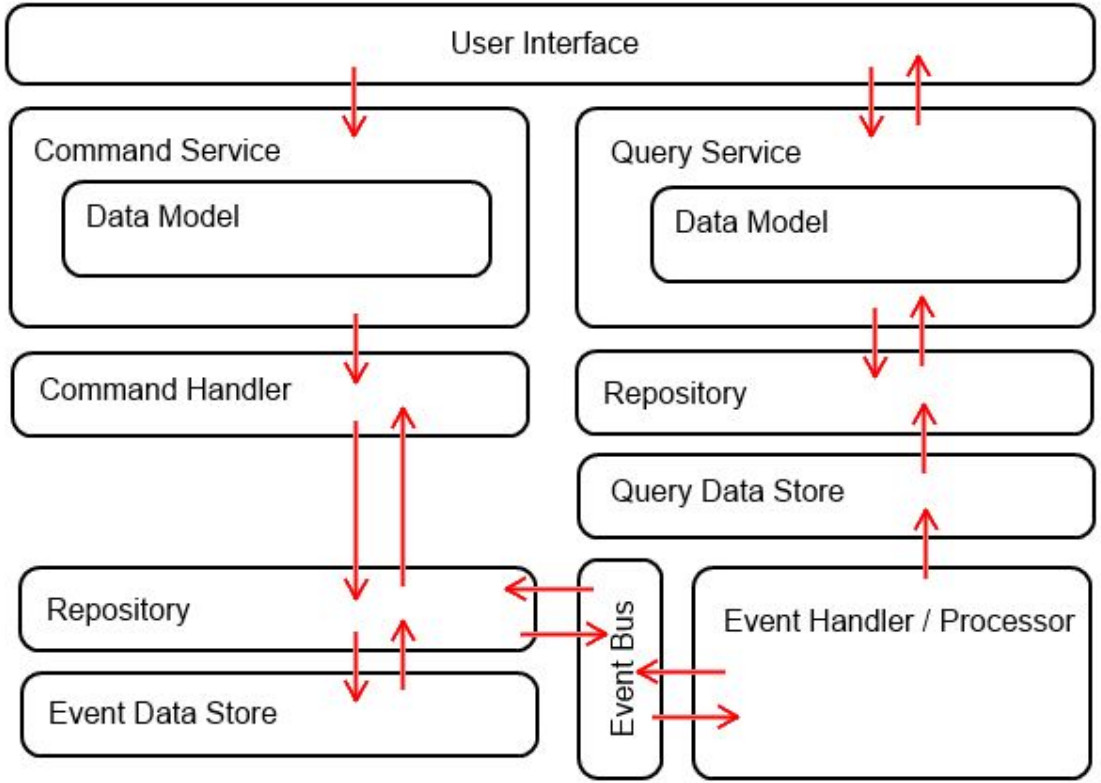
A New Reference Architecture



A New Reference Architecture



A New Reference Architecture



SmartLight App

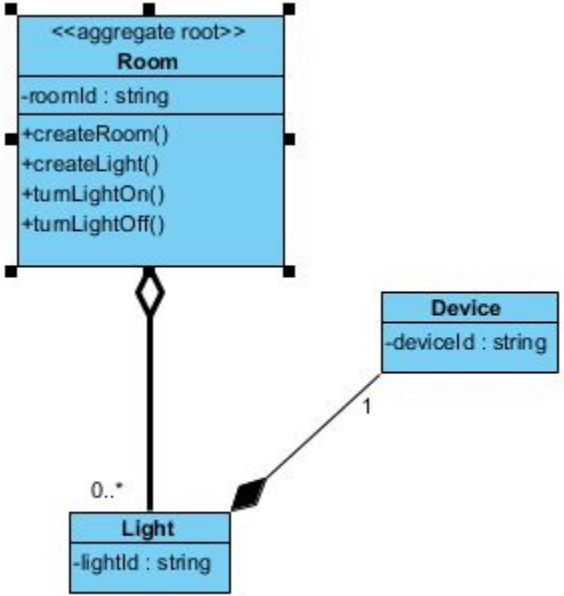
Da utente smartlight, voglio accendere e spegnere una luce in una stanza tramite API, così posso gestire l'illuminazione anche da remoto

Da utente smartlight, voglio ricevere lo stato delle luci in una stanza, così posso controllare l'illuminazione della mia casa

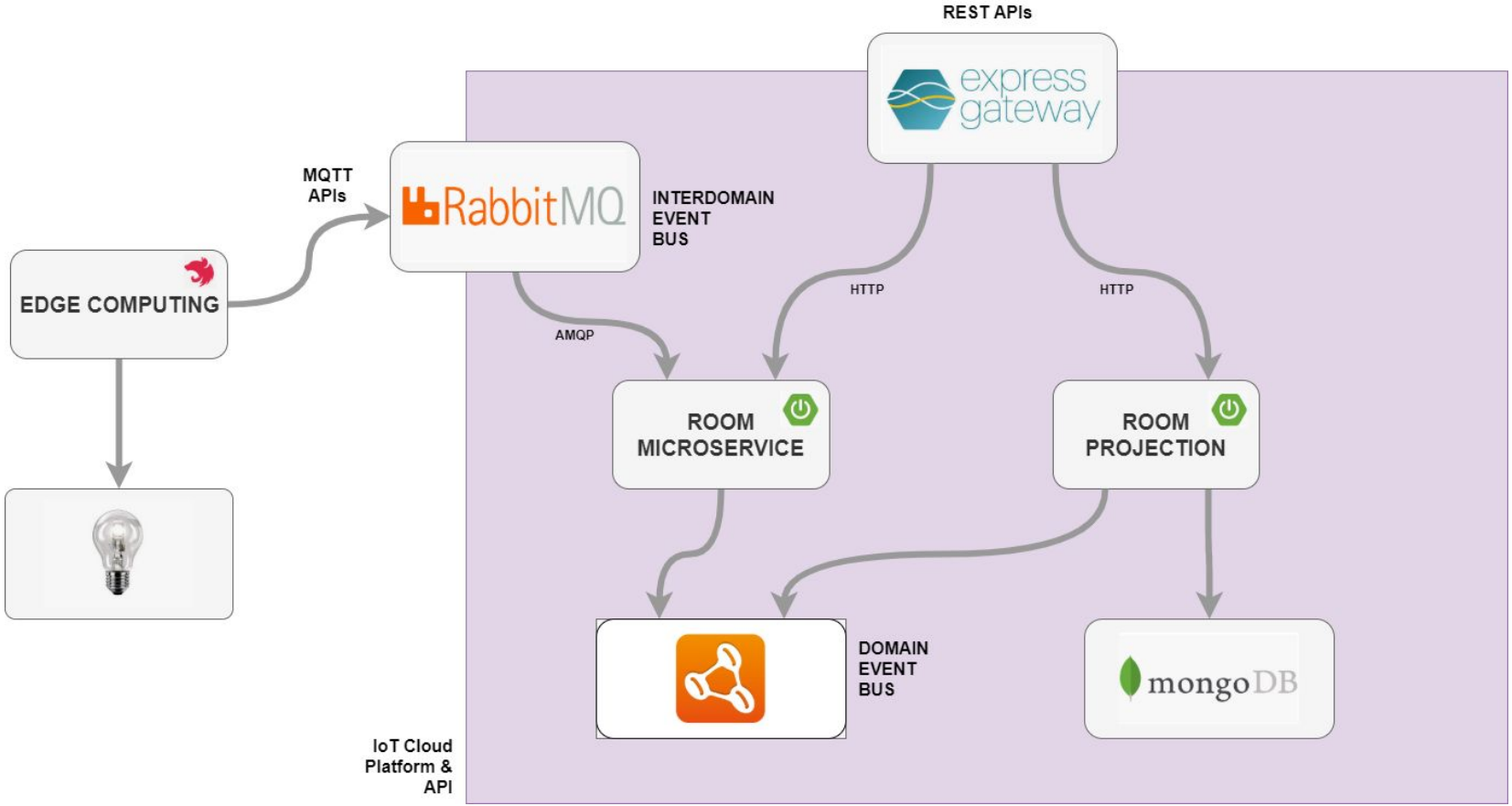
BONUS

Da utente smartlight, voglio conoscere il tempo totale di accensione delle luci in una stanza, così posso tenere sotto controllo il consumo energetico

SmartLight App



SmartLight App



Conclusions

Conclusions

Questo approccio è adatto quando

- Le informazioni contano più dei dati
- E’ forte il legame tra la realtà fisica e il modello digitalizzato
- L’applicazione reagisce ad eventi
- L’applicazione gestisce interazioni complesse tra sistemi indipendenti
- Il prodotto evolve rapidamente
- Le condizioni di carico possono variare rapidamente

Conclusions

Rischi:

- Perdere il controllo del comportamento globale del sistema
- Complessità di gestione dell'infrastruttura
- Tecnologie non *mainstream*
- Rende semplici cose complesse, ma rende complesse le cose semplici

Conclusions

CQRS (& friends) <3 IoT

- Trasforma eventi in insight
- Adatto a sistemi distribuiti
- Pensa in maniera event-driven
- Pensa in maniera asincrona, reattiva

?

Thank you for Your Time!

scm
woodworking technology

P.S.

WE ARE HIRING!!

scmgroup.com/careers