



# Creare applicazioni REALI con Microsoft Orleans

30 Settembre 2023

*Innova Saturday 2023 - Pordenone*



# SPONSORS



# Anatomia di un'applicazione REALE

- Microservizi
- Swagger & OpenAPI client generator
- Layered architecture
- Serve una cache
- Mini database per microservizio
  - Diversi database da gestire
  - ORM
  - Migration
- Dashboard per monitoring
- Gestione delle transazioni aka Sagas
- Una manciata di job non manca mai
- Message bus
- Ci serve un DevOp per il deploy
- Oppure Event sourcing (ma qui è tutta un'altra cosa... serve altro)



# Dipendenze di un'applicazione REALE

## Microservizi

Project Tye



OpenAPITools/openapi-generator

OpenAPI Generator allows generation of API client libraries (SDK generation), server stubs, documentation and configuration automatically given an OpenAPI Spec...



## Dashboard



## Transazioni

OpenSleigh



## Message bus

RabbitMQ



## Caching



redis



## Librerie



MediatR

MiniProfiler/dotnet

A simple but effective mini-profiler for ASP.NET (and Core) websites

auto<X>mapper

QUARTZ

Hangfire

An easy way to perform background processing in .NET and .NET Core applications. No Windows Service or separate process required. Backed by persistent storage. Open and free for commercial use.



## DBs



mongoDB

FluentMigrator



# Dipendenenze di un

## Microservizi

### Project Tyke



OpenAPITools/openapi-generator

OpenAPI Generator allows generation of API client libraries (SDK generation), server stubs, documentation and configuration automatically given an OpenAPI Spec...



## Caching



redis



Milan Jovanović @mjovanovic... · 9h  
Here are 13 excellent libraries I use in my Microservices:

1. EF Core
2. Dapper
3. MediatR
4. Refit
5. Polly
6. Scrutor
7. MassTransit
8. FluentValidation
9. Hangfire
10. Testcontainers
11. FluentAssertions
12. NetArchTest.Rules
13. StackExchange.Redis

18

57

353

28,8K



age bus

RabbitMQ



DBs



goDB

FluentMigrator



Microsoft SQL Server



PostgreSQL



EVENT STORE



Marty, torniamo nel 2010

2010



2010

La regina Elisabetta...



2010



15 Aprile

Eyjafjallajokull inizia a eruttare



2010



11 Giugno 2010

L'Italia campione del mondo in carica affronta  
il Paraguay nella prima partita del girone.



2010



28 Novembre 2010

Esplode il caso Wikileaks



2010



Novembre 2010

Si comincia a parlare di Orleans





# Orleans: A Framework for Cloud Computing

Sergey Bykov, Alan Geller, Gabriel Kliot, Jim Larus, Ravi Pandya, [Jorgen Thelin](#)

MSR-TR-2010-159 | November 2010

Superseded by SOCC '12 publication. Please read and cite that publication.

Download BibTex

[View Publication](#)

## Groups

[Systems Research Group - Redmond](#)

## Projects

[Orleans - Virtual Actors](#)

## Research Areas

[Programming languages and software engineering](#)

[Systems and networking](#)



[Research](#)[Our research](#) ▾[Programs & events](#) ▾[Blogs & podcasts](#) ▾[About](#) ▾[Sign up: Research Newsletter](#)[All Microsoft](#) ▾[Search](#)

# Orleans: A Framework for Cloud Computing

Sergey Bykov, Alan Geller, Gabriel Kliot, Jim Larus, Ravi Pandya, [Jorgen Thelin](#)

MSR-TR-2010-159 | November 2010

Superseded by SOCC '12 publication. Please read and cite that publication.

 [Download BibTex](#)[View Publication](#)

## Groups

[Systems Research Group - Redmond](#)

## Projects

[Orleans - Virtual Actors](#)

## Research Areas

[Programming languages and software engineering](#)

[Systems and networking](#)

# Orleans

## A Framework for Cloud Computing



[Research](#)[Our research](#) ▾[Programs & events](#) ▾[Blogs & podcasts](#) ▾[About](#) ▾[Sign up: Research Newsletter](#)[All Microsoft](#) ▾[Search](#)

# Orleans: A Framework for Cloud Computing

Sergey Bykov, Alan Geller, Gabriel Kliot, Jim Larus, Ravi Pandya, [Jorgen Thelin](#)

MSR-TR-2010-159 | November 2010

Superseded by SOCC '12 publication. Please read and cite that publication.

 [Download BibTex](#)[View Publication](#)

## Groups

[Systems Research Group - Redmond](#)

## Projects

[Orleans - Virtual Actors](#)

## Research Areas

[Programming languages and software engineering](#)

[Systems and networking](#)

# Orleans

## A Framework for Cloud Computing

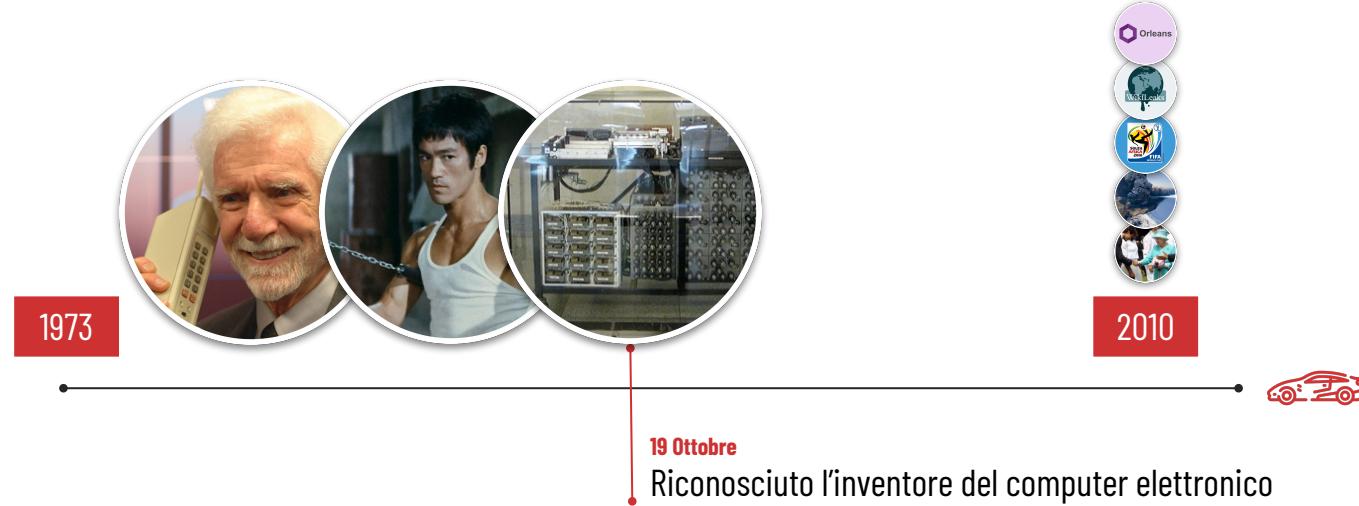


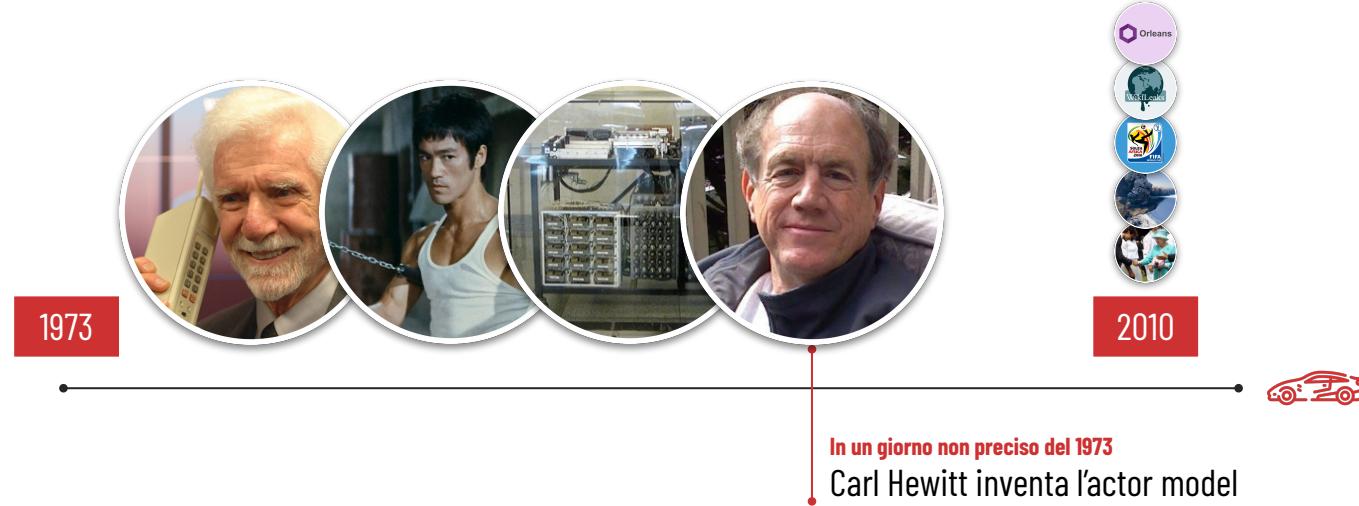


Marty, serve andare più indietro: nel 1973









[https://www.youtube.com/watch?v=7erJ1DV\\_Tlo&ab\\_channel=jasonofthe133t](https://www.youtube.com/watch?v=7erJ1DV_Tlo&ab_channel=jasonofthe133t)



# Carl Hewitt ci dice che un **attore**...



E' l'**unità fondamentale** di calcolo, e...

- **processa** i dati che riceve
- **persiste** ciò che dev'essere salvato (lo stato)
- **comunica** tramite messaggi



Da solo non serve a niente, ma diventa interessante all'interno di un **sistema**



Processa un messaggio alla volta

- risolve i problemi di **concorrenza**
- semplifica il **debug**



# Actor Model - Definizione

“

The **actor model** in computer science is a mathematical model of concurrent computation that treats actor as the universal primitive of concurrent computation.

In response to a message it receives, an actor can: make local decisions, create more actors, send more messages, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats actor as the universal primitive of concurrent computation.

In response to a message it receives, an actor can: make local decisions, create more actors, send more messages, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

In response to a message it receives, an actor can: make local decisions, create more actors, send more messages, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

**In response to a message it receives**, an actor can: make local decisions, create more actors, send more messages, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

In response to a message it receives, an actor can: **make local decisions**, create more actors, send more messages, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

In response to a message it receives, an actor can: **make local decisions, create more actors**, send more messages, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

In response to a message it receives, an actor can: **make local decisions, create more actors, send more messages**, and determine how to respond to the next message received.

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

**In response to a message it receives**, an actor can: **make local decisions, create more actors, send more messages, and determine how to respond to the next message received.**

Actors may modify their own private state, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

**In response to a message it receives**, an actor can: **make local decisions, create more actors, send more messages, and determine how to respond to the next message received.**

Actors **may modify their own private state**, but can only affect each other indirectly through messaging (removing the need for lock-based synchronization).

Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

”



# Actor Model - Definizione

“

The **actor model** in computer science is a **mathematical model** of concurrent computation that treats **actor as the universal primitive of concurrent computation**.

In response to a message it receives, an actor can: **make local decisions, create more actors, send more messages, and determine how to respond to the next message received.**

Actors **may modify their own private state**, but can only affect each other indirectly through messaging (**removing** the need for **lock-based synchronization**).

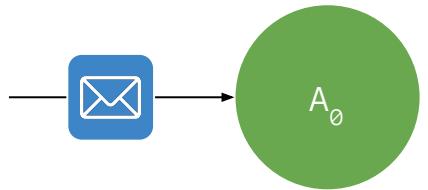
Fonte: [https://en.wikipedia.org/wiki/Actor\\_model](https://en.wikipedia.org/wiki/Actor_model)

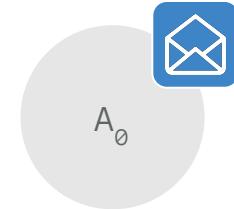
”



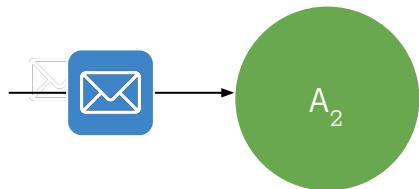


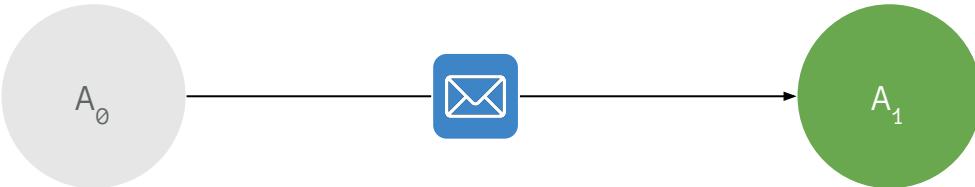
Everything is an Actor!

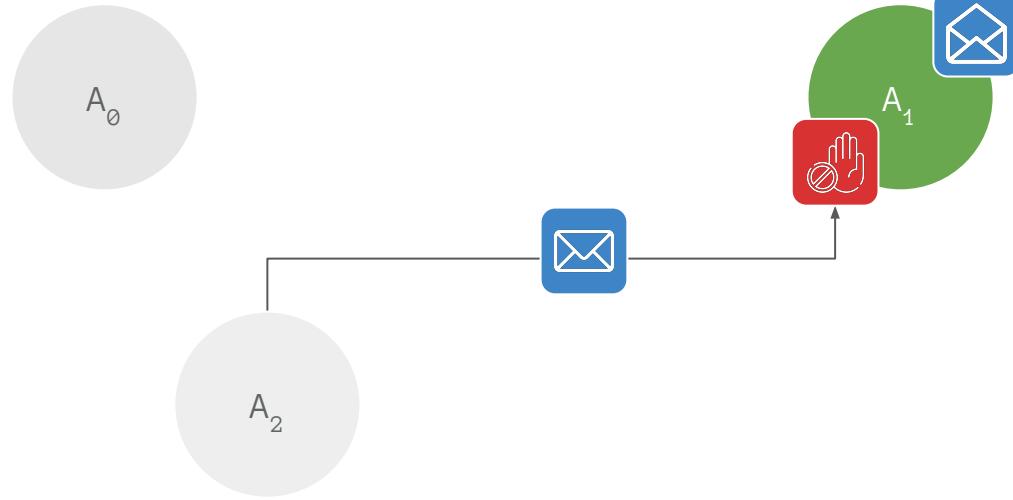


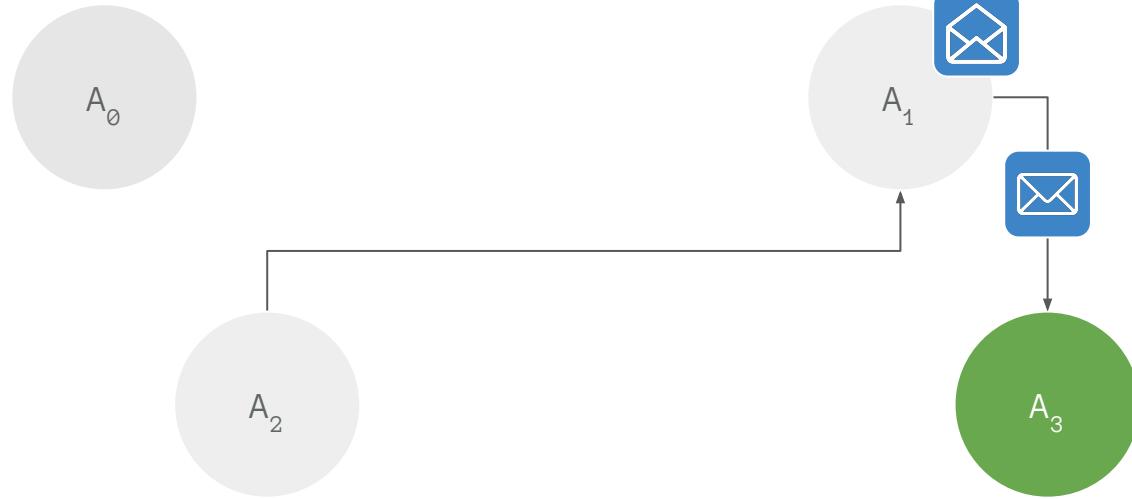


$A_0$

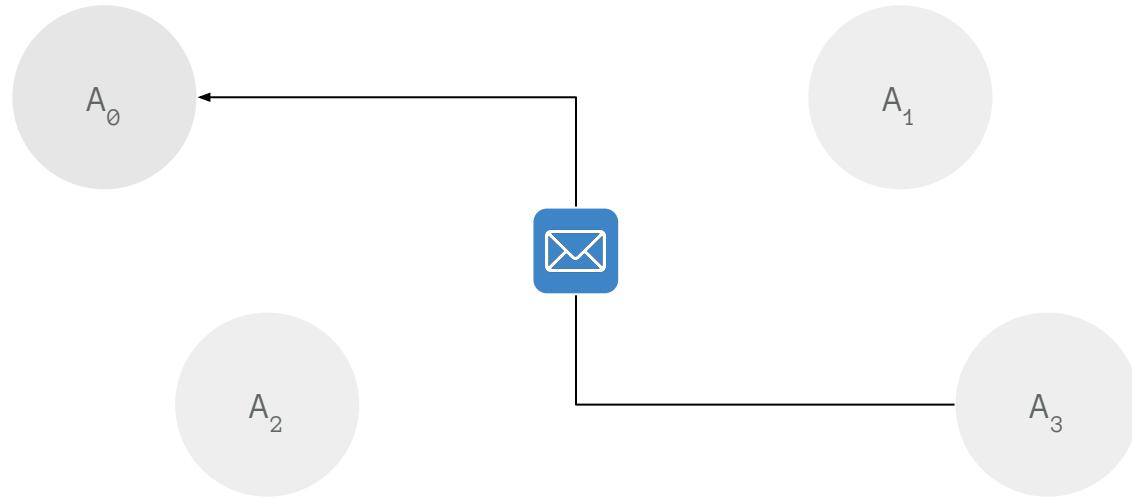


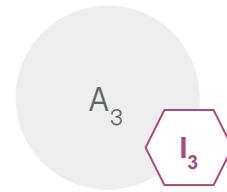
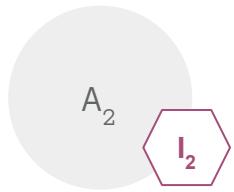
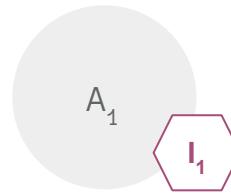
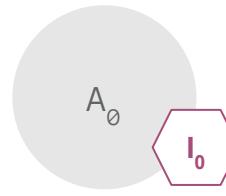


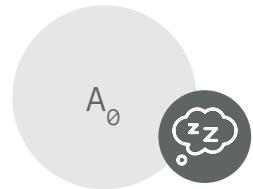


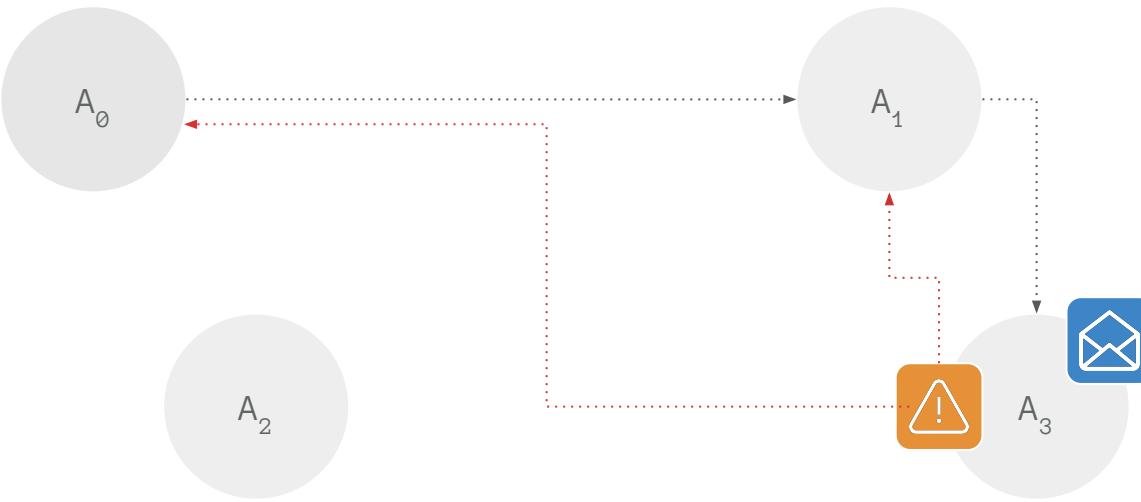


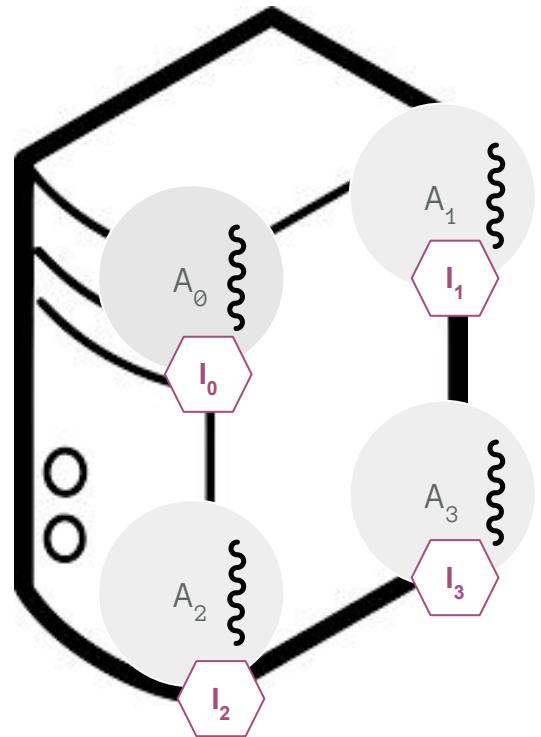


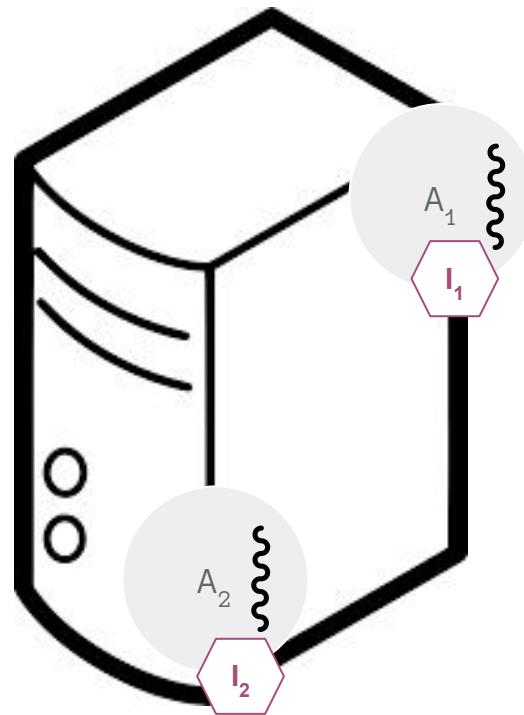
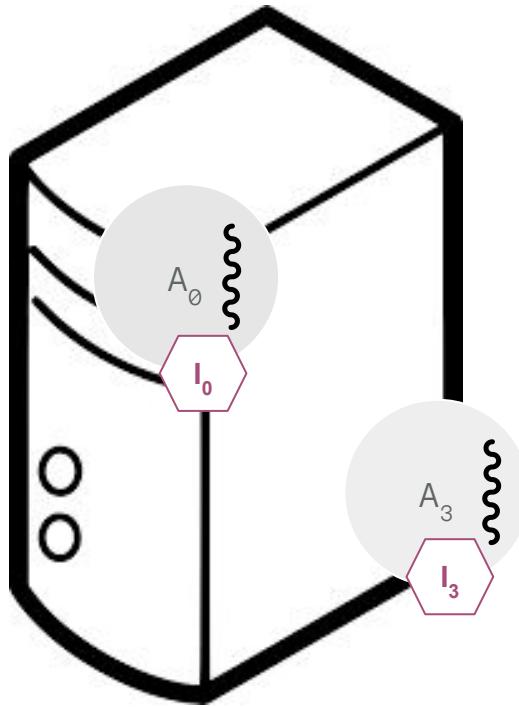


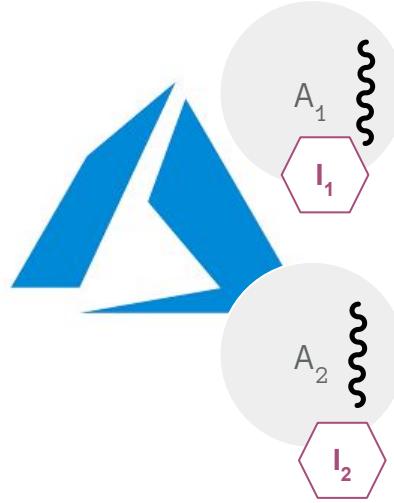
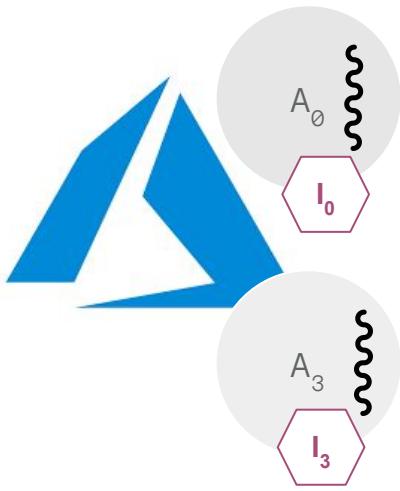












# Actor Model... per gli altri

*Erlang*



Dart



elixir



# Actor Model... nel mondo .net



microsoft/coyote

Coyote is a library and tool for testing concurrent C# code and deterministically reproducing bugs.



19  
Contributors

47  
Used by

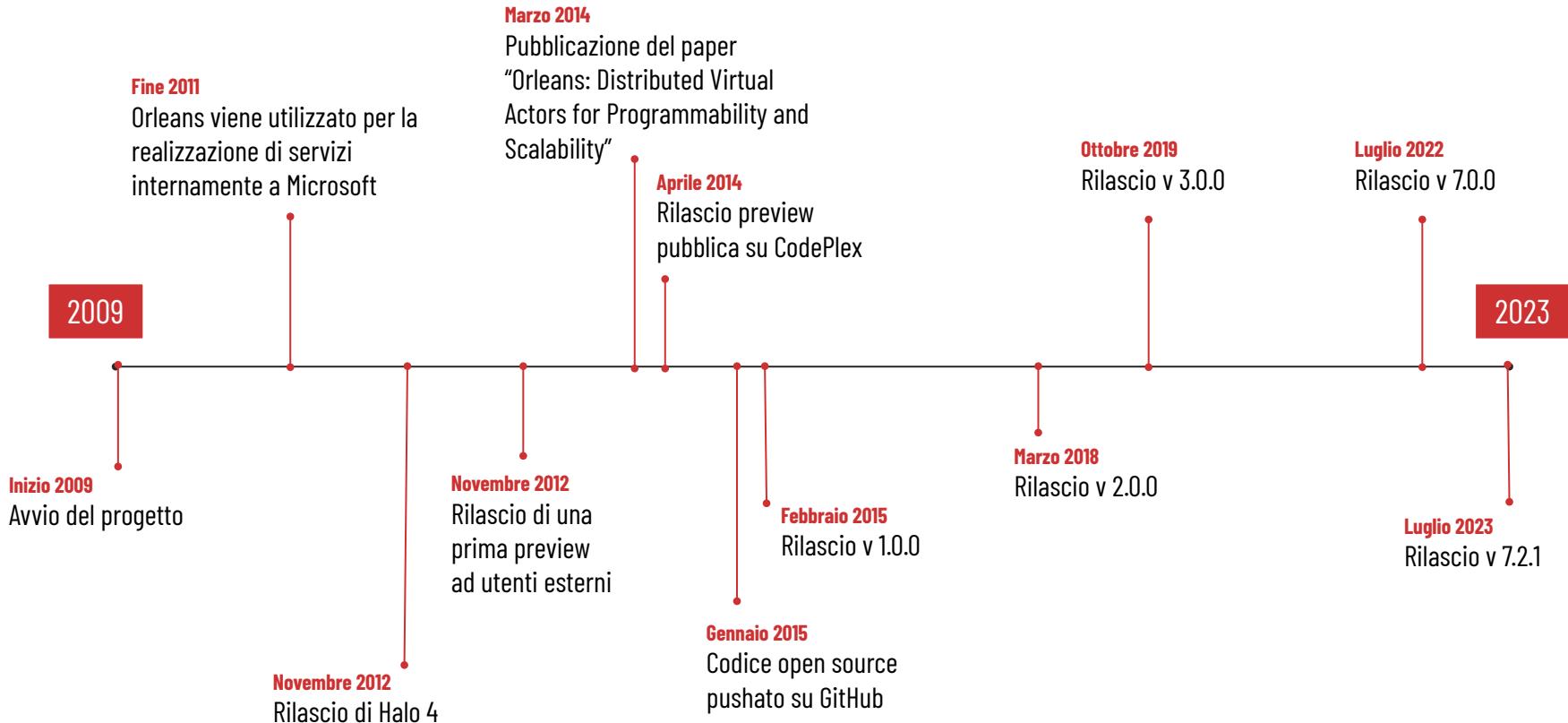
23  
Discussions

1k  
Stars

75  
Forks



# Orleans timeline



# Microsoft Orleans viene utilizzato da



Diversi servizi di **Microsoft Azure**



**Xbox**



**Skype**



**Halo**



**PlayFab**



**Gears of War**







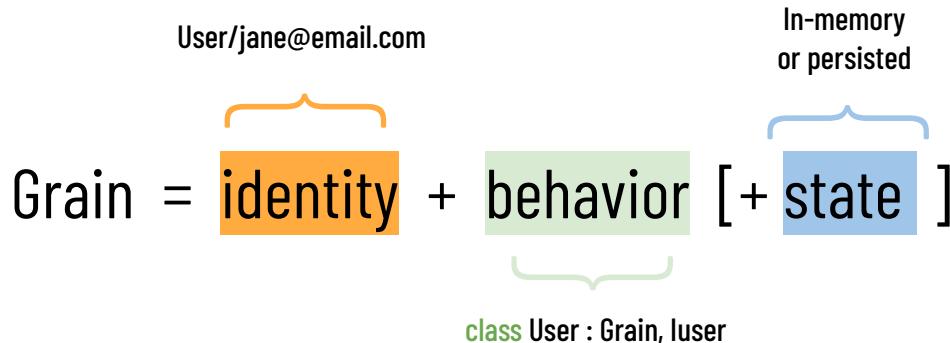
## GRAINS



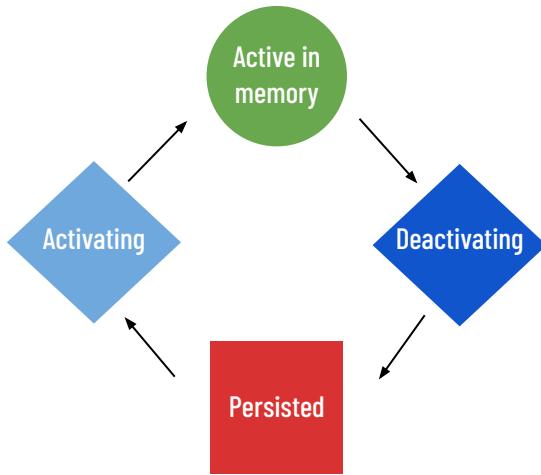
In analogia con i granelli di grano, sono  
**oggetti piccoli** e solitamente **abbondanti**

# Microsoft Orleans - Grains

- E' una **classe**, che eredita da **Grain** ed implementa un'interfaccia **IGrainWith...Key**
- Ciascun *grain* ha una **chiave**
  - *Long, GUID, String, GUID + String, Long + string*
- Per una data **identity** e **tipologia di grain**, viene attivata una **singola istanza**



# Microsoft Orleans - Grains - Lifecycle

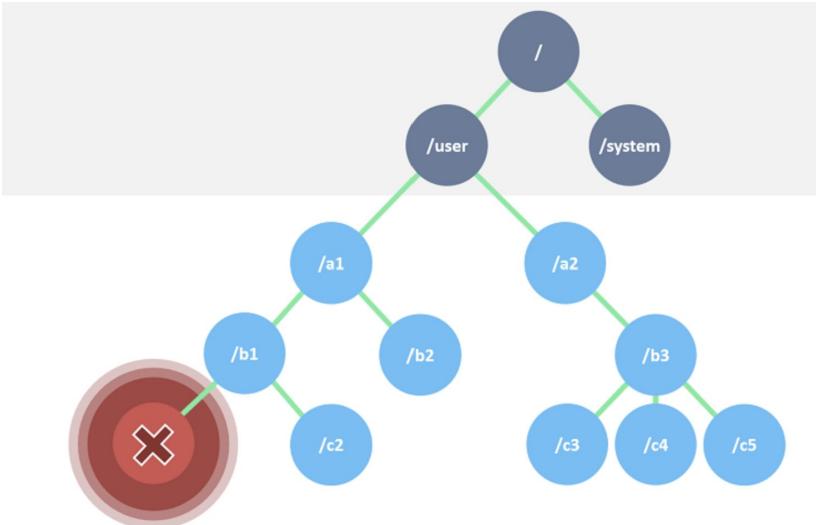


- Entità logiche che **esistono sempre**
- L'applicazione non crea/distrugge i grains
- L'Attivazione dell'istanza del *grain* avviene **quando serve**
  - Attivazione non esplicita, ma legata alla ricezione di un messaggio
- Il **GC** del runtime di Orleans si occupa di **disattivare** i grains che **non ricevono messaggi**
- Ogni grain processa **un messaggio alla volta** (*turn-based concurrency*)



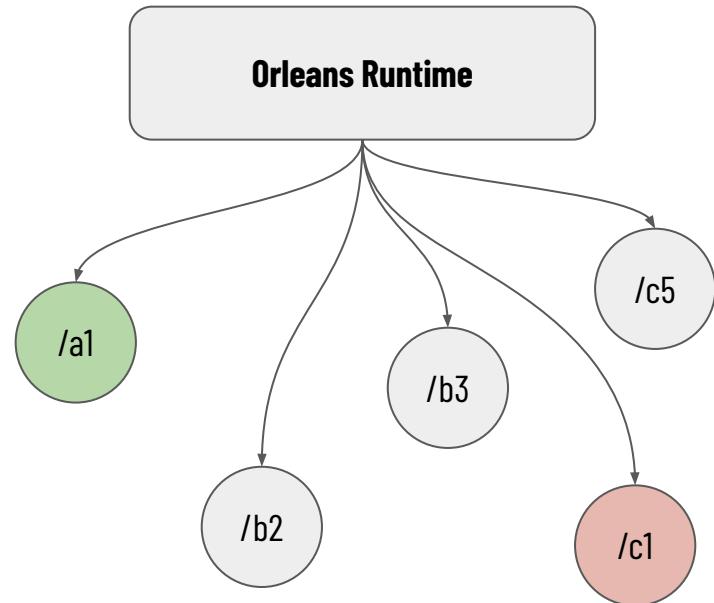
# Microsoft Orleans - Grains - Virtual Actors

Sistema ad attori **tradizionale**



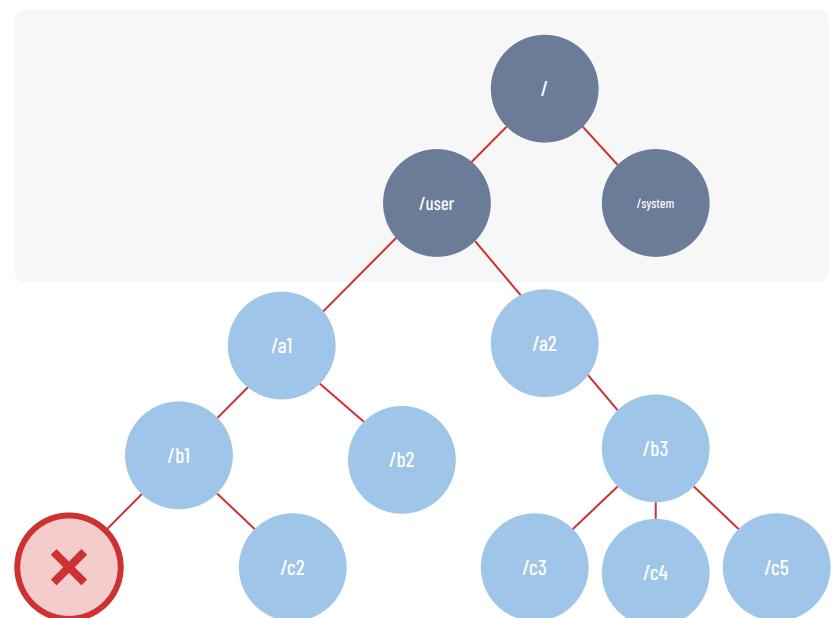
<https://getakka.net/articles/concepts/supervision.html>

Microsoft **Orleans**

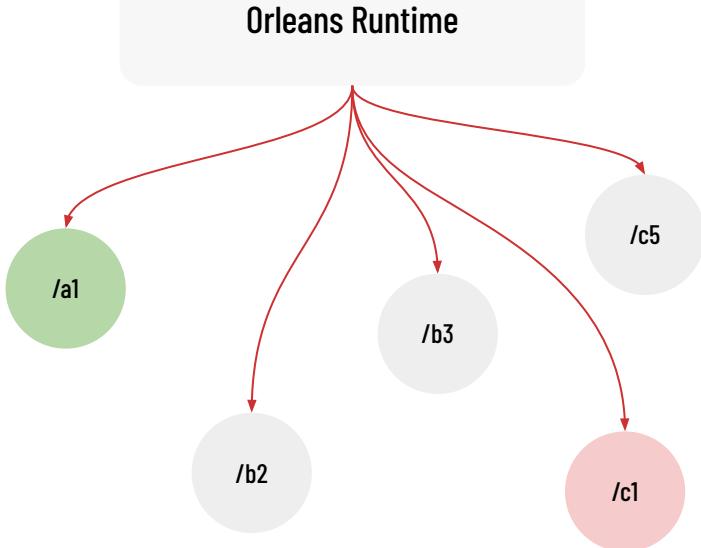


# Microsoft Orleans - Grains are *Virtual Actors*

Sistema ad attori **tradizionale**



Microsoft **Orleans**





Sono le **strutture che contengono**  
**i granelli di grano**

SILOS

GRAINS



# Microsoft Orleans - Silos

- I silos sono il **runtimes** che gestisce i *grains*
- Può essere hostato in un Windows Service, in una Console Application
  - *On-premise, in Docker container, nel cloud*
- Tipicamente si ha **un silo per macchina**
- I silo richiedono l'apertura di 2 porte TCP
  - *30.000 per comunicazioni silo-to-silo*
  - *11.111 per comunicazioni client-to-silo*
- Un silo **non esponde** direttamente delle **API**



Un insieme di silo che **cooperano**  
e realizzano un sistema

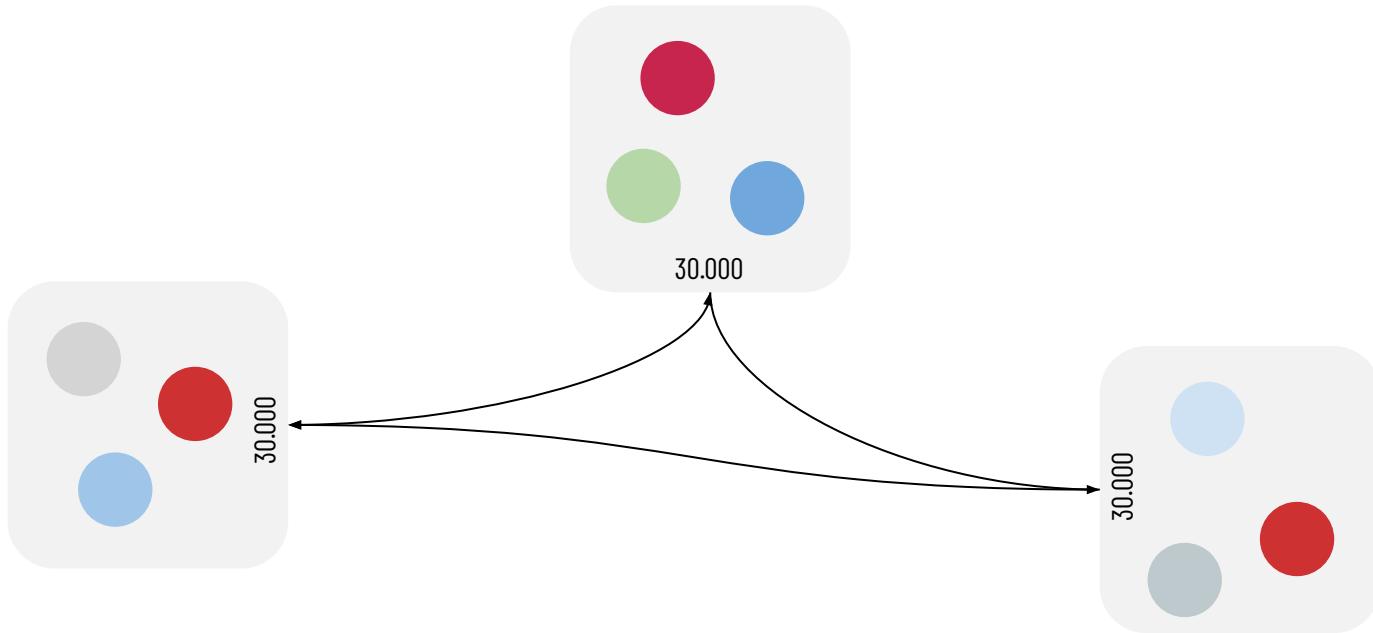
**CLUSTER**

**SILOS**

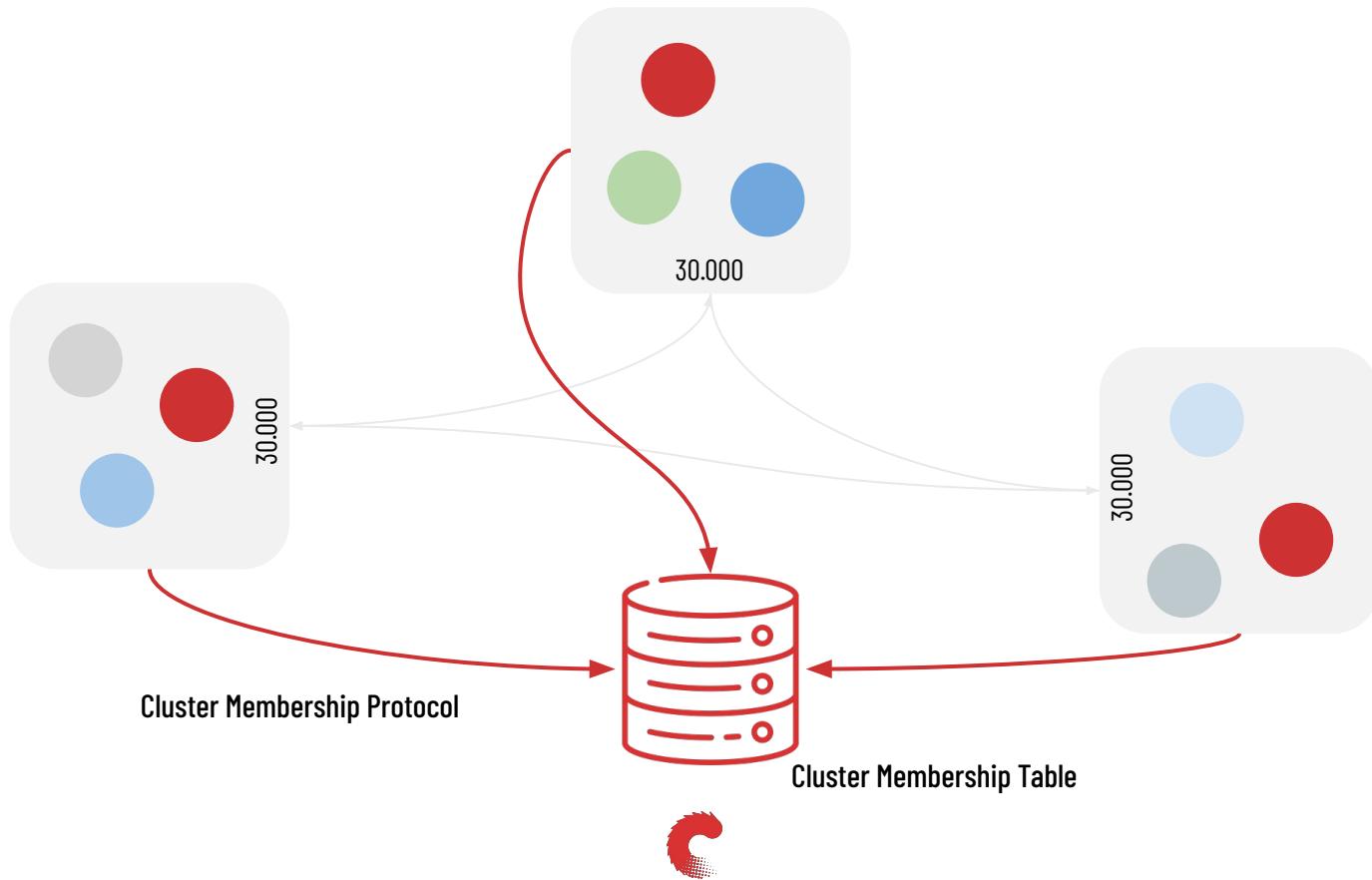
**GRAINS**



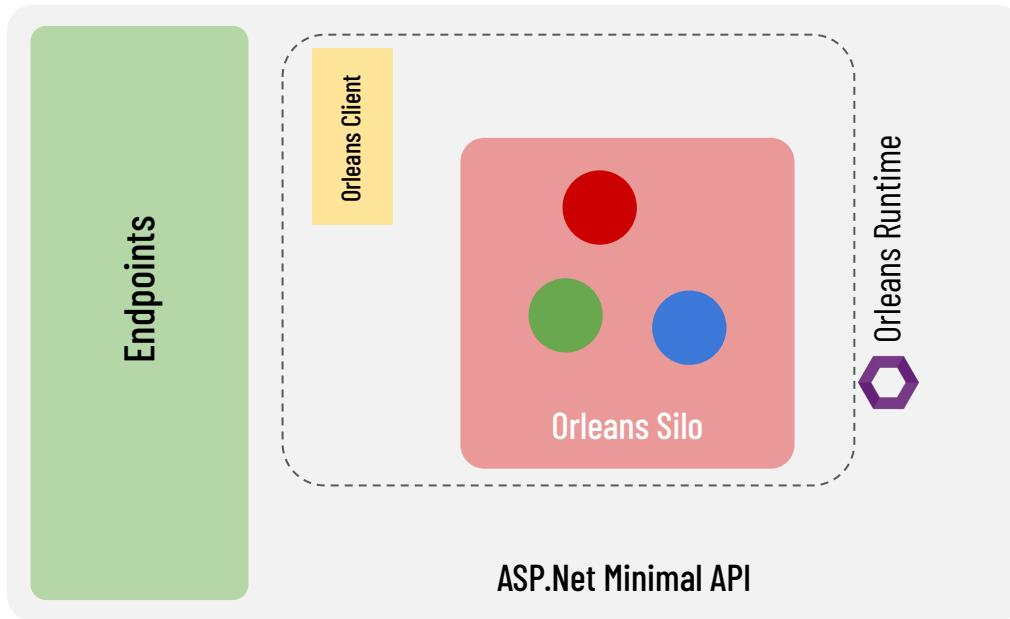
# Microsoft Orleans - Cluster



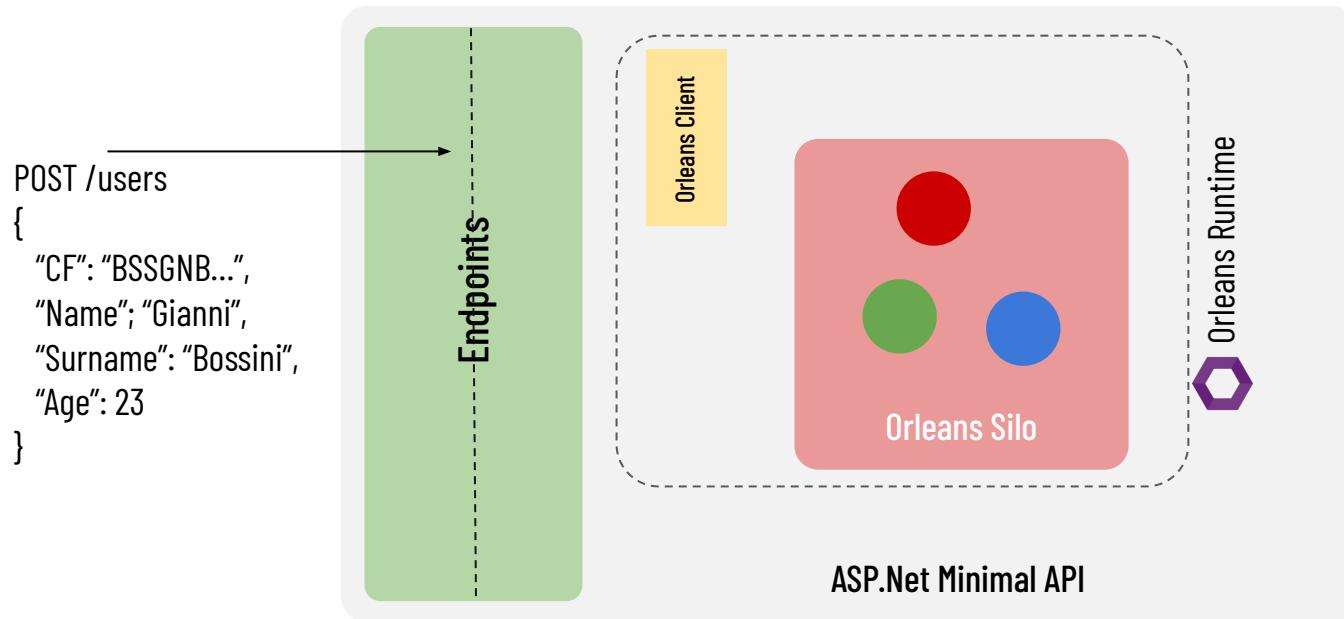
# Microsoft Orleans - Cluster



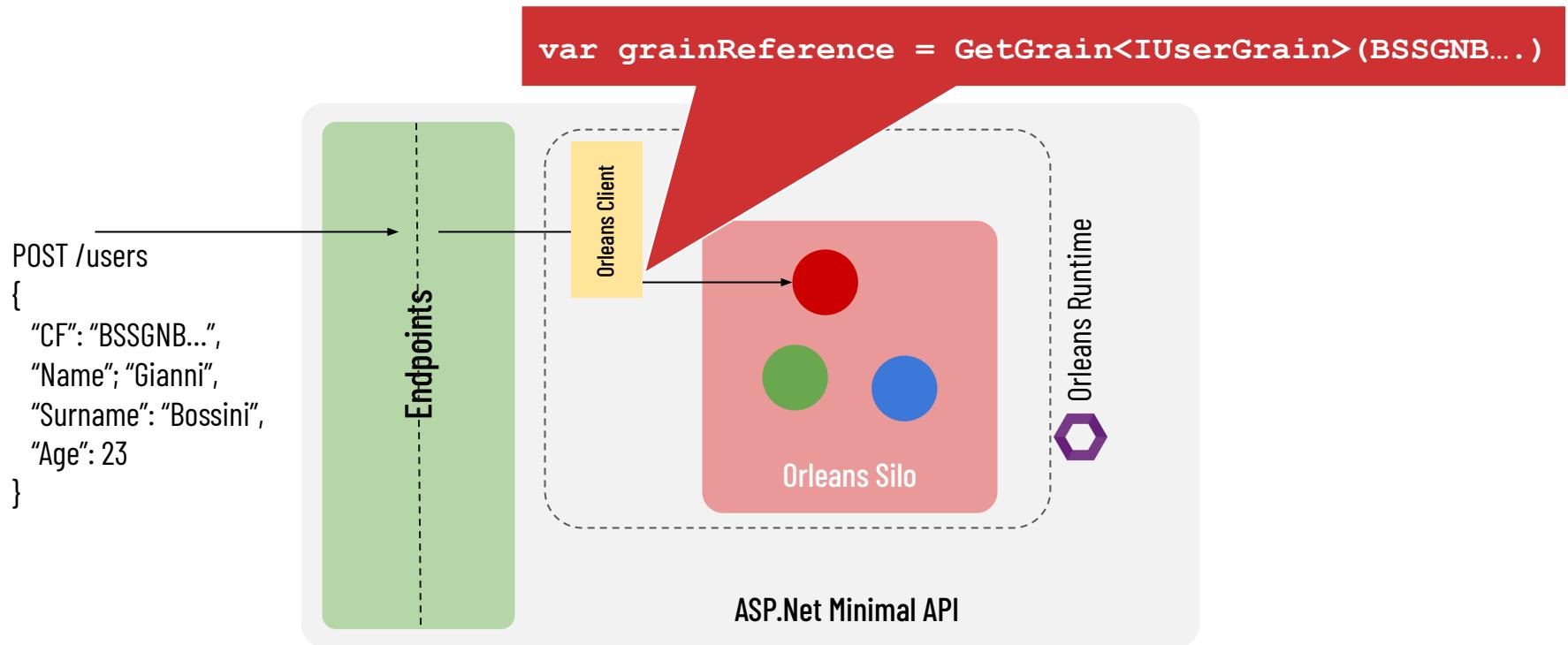
# Microsoft Orleans - Hosting - CO-HOSTED CLIENTS



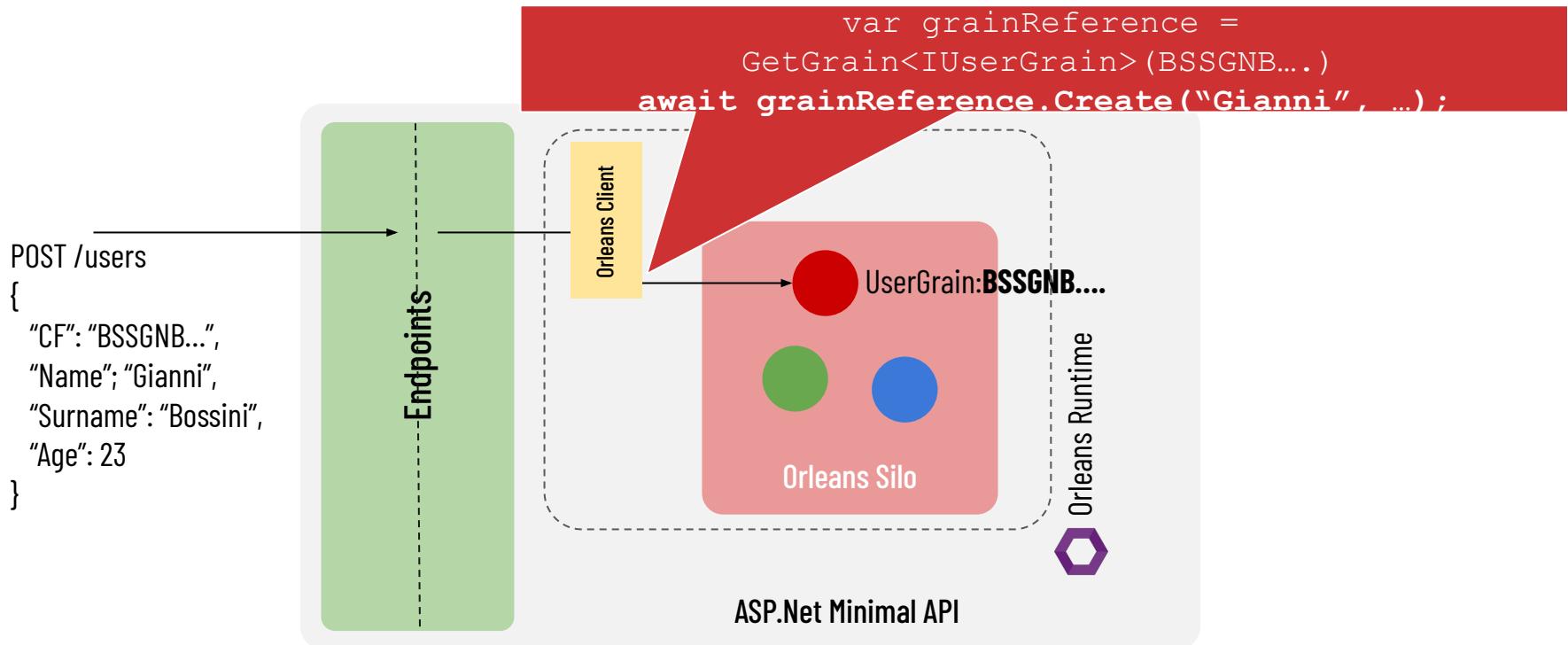
# Microsoft Orleans - Hosting - CO-HOSTED CLIENTS



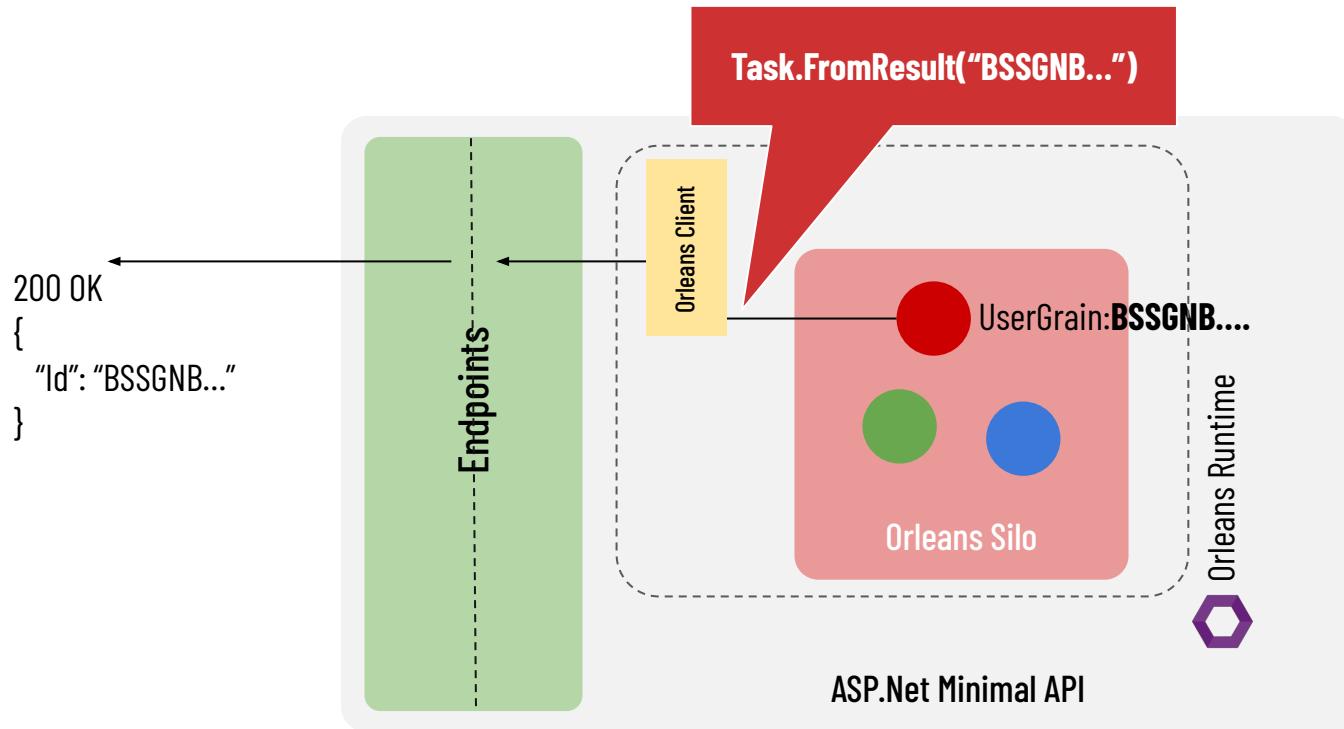
# Microsoft Orleans - Hosting - CO-HOSTED CLIENTS



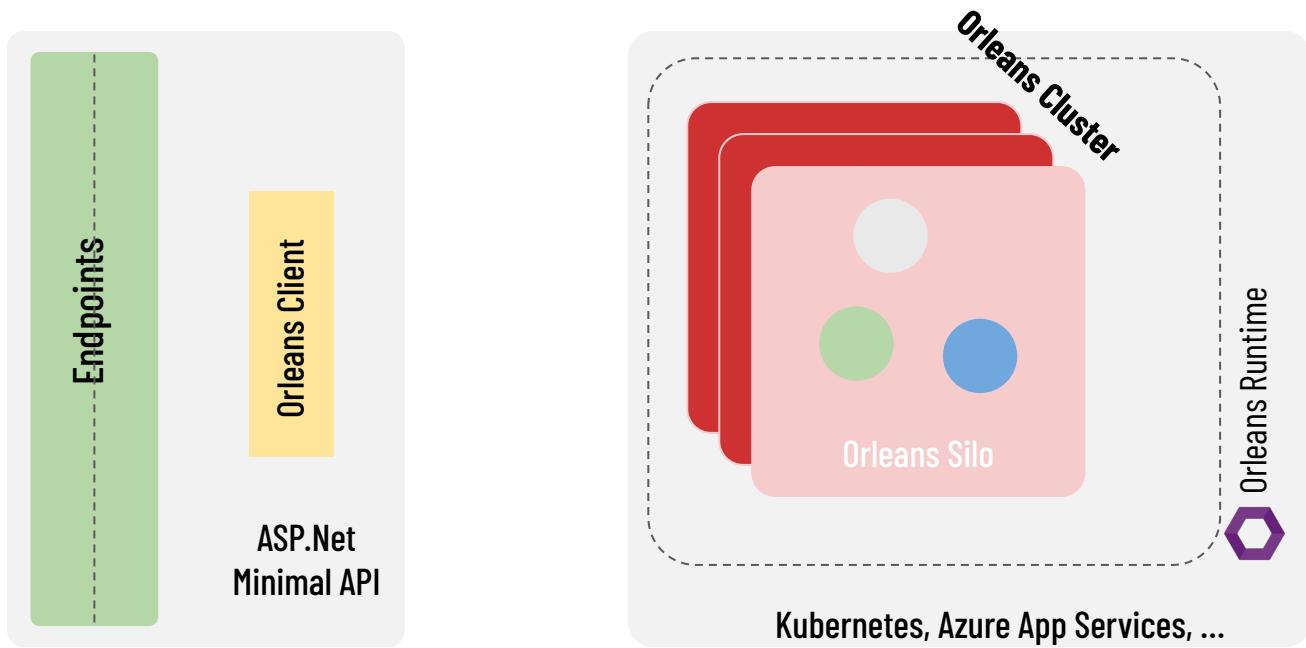
# Microsoft Orleans - Hosting - CO-HOSTED CLIENTS



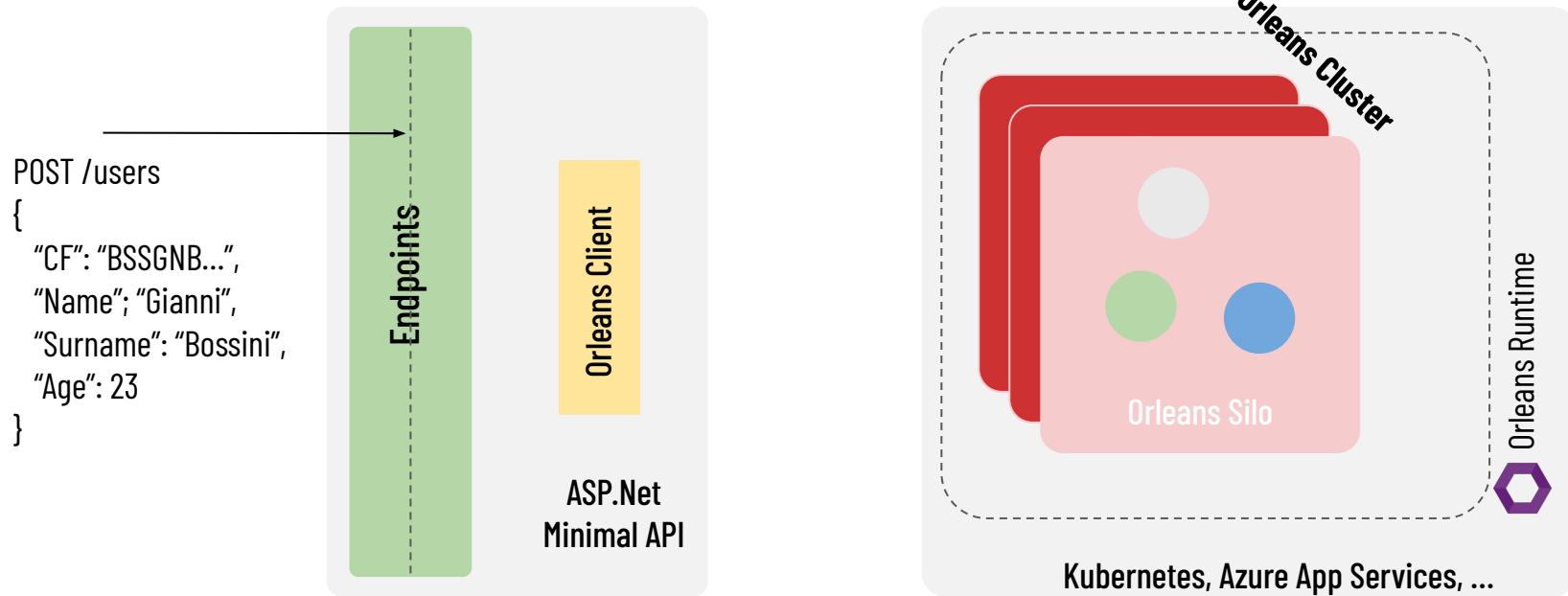
# Microsoft Orleans - Hosting - CO-HOSTED CLIENTS



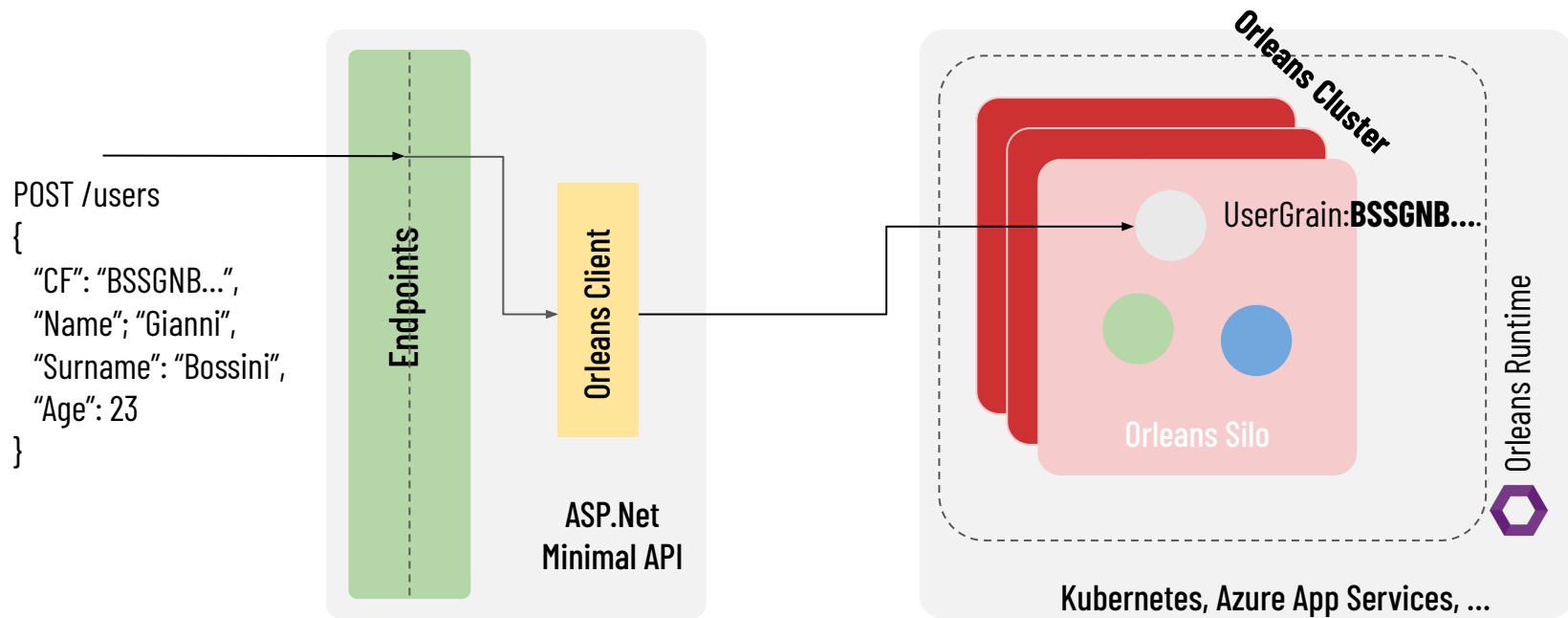
# Microsoft Orleans - Hosting - EXTERNAL CLIENTS



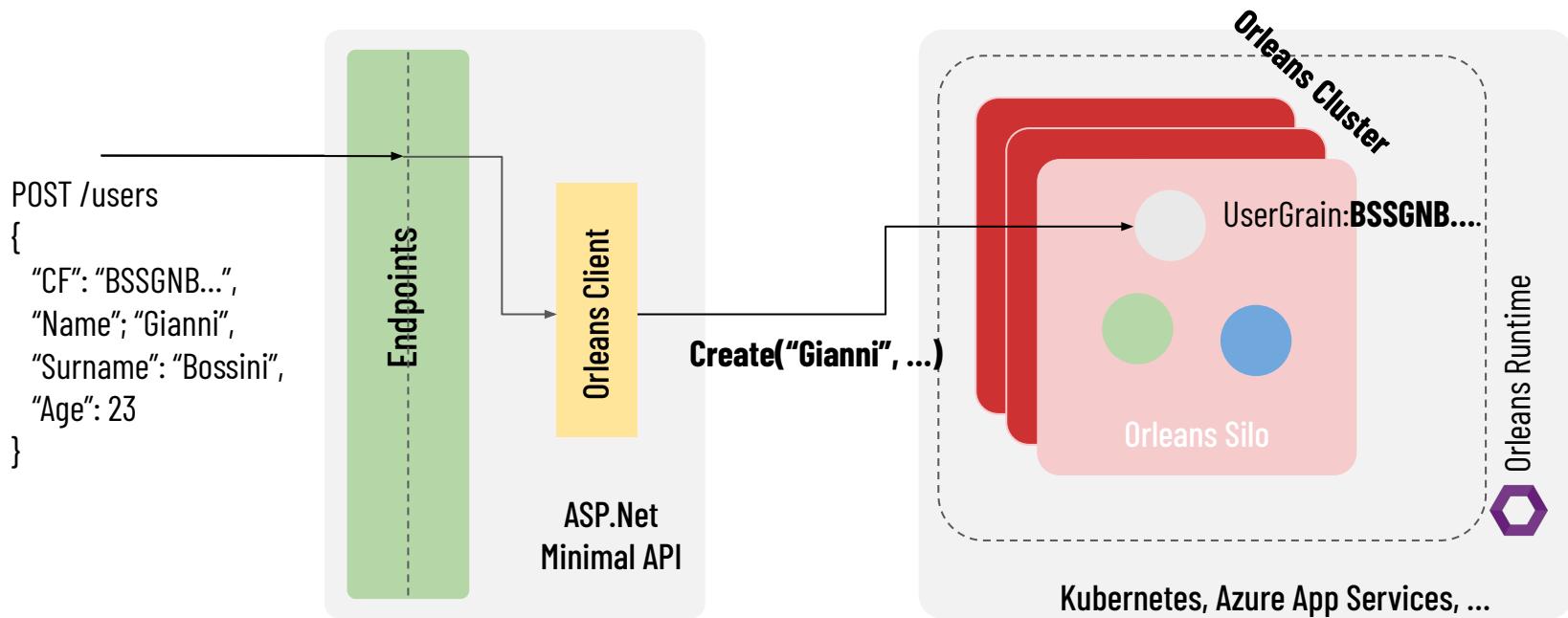
# Microsoft Orleans - Hosting - EXTERNAL CLIENTS



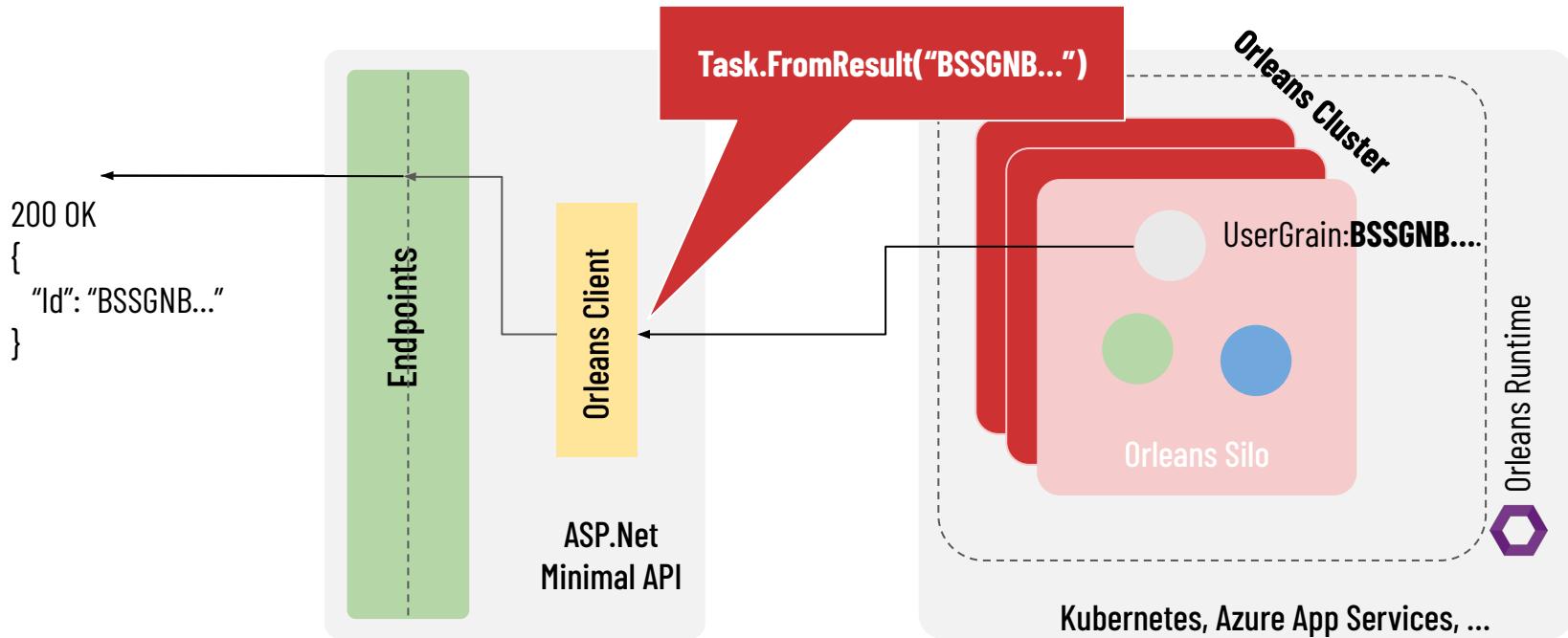
# Microsoft Orleans - Hosting - EXTERNAL CLIENTS



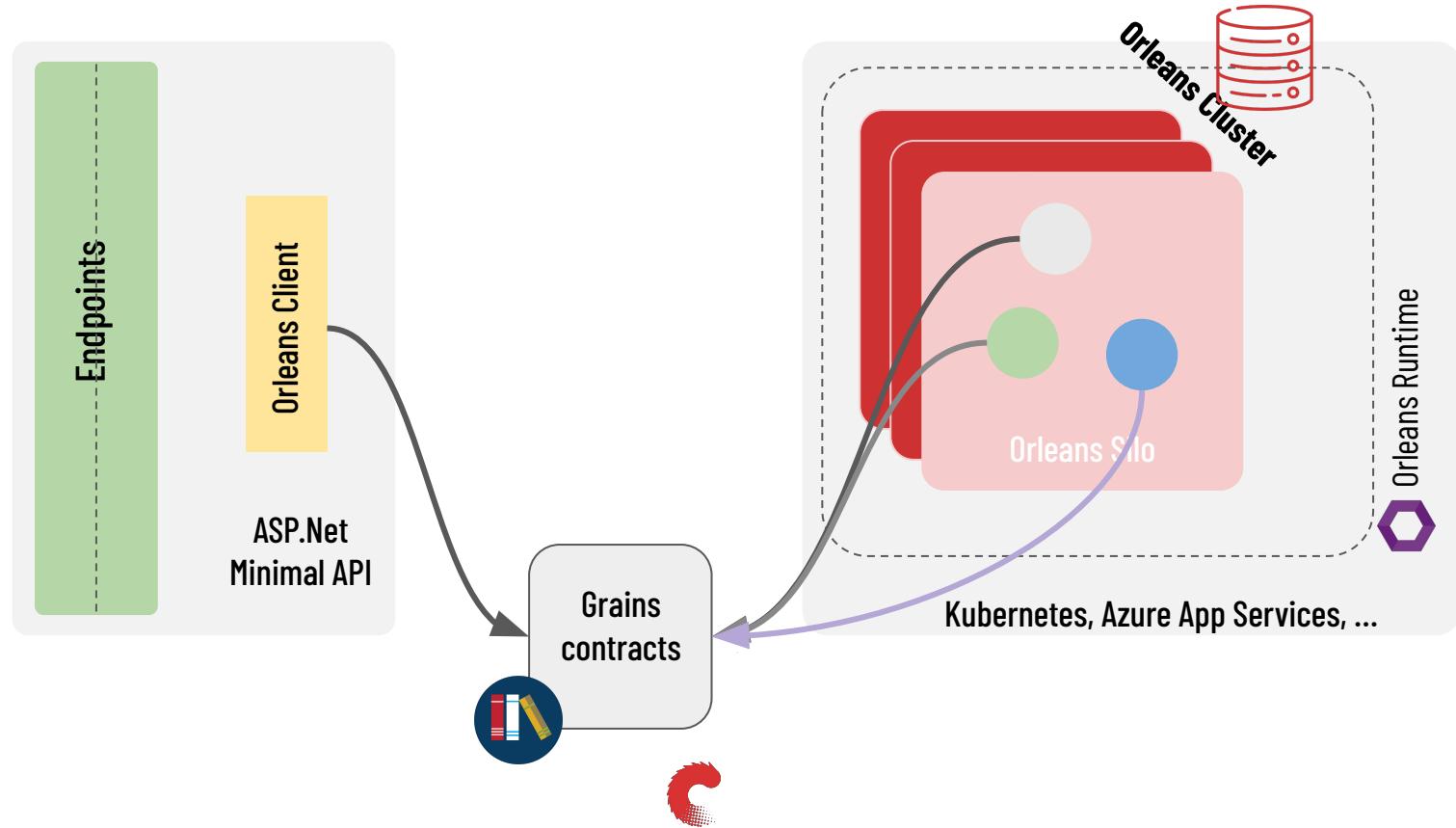
# Microsoft Orleans - Hosting - EXTERNAL CLIENTS

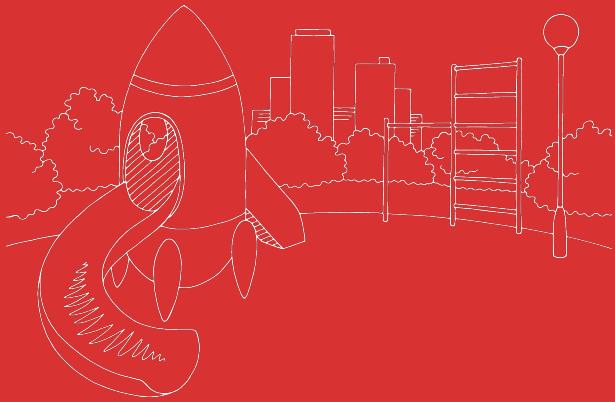


# Microsoft Orleans - Hosting - EXTERNAL CLIENTS



# Microsoft Orleans - Hosting - EXTERNAL CLIENTS





# DEMO - UrlShortener

---

- Grain e test
- Ottimizzazioni
- Dashboard e osservabilità
- Stateless workers, Timers e Reminders
- Filtri
- Observer
- Persistenza
- Road to producion
- From Co-hosted to External clients

# Microsoft Orleans aka **Distributed .NET**



- Da applicazioni on-premise fino a complessi sistemi distribuiti
  - senza cambiare l'approccio OOP
- Applicazioni scalabili e affidabili
  - *Scala da un singolo server locale a centinaia a migliaia di applicazioni distribuite e a disponibilità elevata nel cloud*
  - *"Potrebbe supportare la creazione di un attore per ogni abitante della Terra"*



# Addio dipendenze, benvenuto Orleans



**Milan Jovanović** @mjovanovic... · 9h   
Here are 13 excellent libraries I use in my  
Microservices:

1. EF Core
2. Dapper
3. MediatR
4. Refit
5. Polly
6. Scrutor
7. MassTransit
8. FluentValidation
9. Hangfire
10. Testcontainers
11. FluentAssertions
12. NetArchTest.Rules
13. StackExchange.Redis

18

57

353

28,8K



# Considerazioni



**SDK maturo**

**Community attiva**

**Tante features complesse**

senza librerie esterne



# Considerazioni



**SDK maturo**  
**Community attiva**  
**Tante features complesse**  
senza librerie esterne



**Applicazioni cloud-native**  
**complesse**, scalabili e  
distribuite richiedendo  
**basso effort** per uno  
sviluppatore .net



# Considerazioni



**SDK maturo**  
**Community** attiva  
**Tante features complesse**  
senza librerie esterne



**Applicazioni cloud-native**  
**complesse**, scalabili e  
distribuite richiedendo  
**basso effort** per uno  
sviluppatore .net



**Monolite modulare**  
composta da  
**nanoservizi** che  
comunicano tramite  
messaggi



# Considerazioni



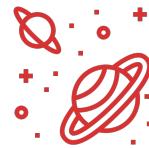
**SDK maturo**  
**Community attiva**  
**Tante features complesse**  
senza librerie esterne



**Applicazioni cloud-native**  
**complesse**, scalabili e  
distribuite richiedendo  
**basso effort** per uno  
sviluppatore .net

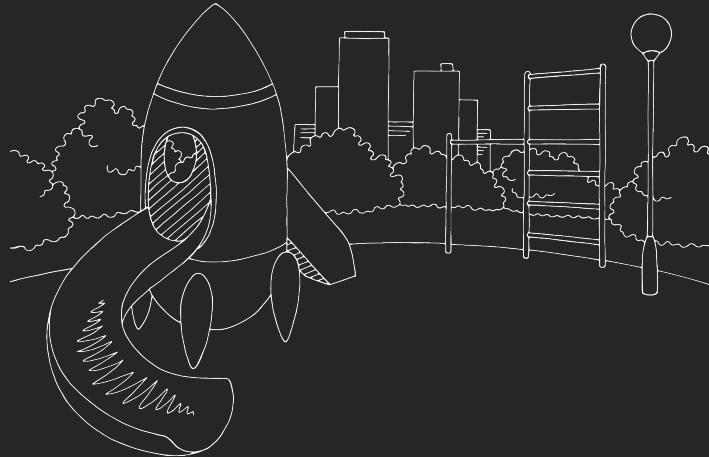


**Monolite modulare**  
composta da  
**nanoservizi** che  
comunicano tramite  
messaggi



**Indexing aka**  
**Actor-Oriented**  
**Database** per  
dominare il mondo





[https://github.com/GianniBortoloBossini/  
innova2023-creare-applicazioni-reali-con-ms-orleans](https://github.com/GianniBortoloBossini/innova2023-creare-applicazioni-reali-con-ms-orleans)



# Materiale - Articoli



- Microsoft Orleans documentation  
<https://learn.microsoft.com/en-us/dotnet/orleans/>
- Intro to Virtual Actors by Microsoft Orleans  
<https://bogdan-dina03.medium.com/intro-to-virtual-actors-by-microsoft-orleans-6ae3264f138d>
- Actor model  
<https://codedocs.org/what-is/actor-model>
- Russ Hammett's blog  
<https://blog.kritner.com/categories/programming/microsoft-orleans/>
- Introduction to Project Orleans  
<https://ideasof.andersaberg.com/development/Introduction-To-Orleans>
- Indexing in an Actor-Oriented Database  
<https://www.cidrdb.org/cidr2017/papers/p29-bernstein-cidr17.pdf>



# Materiale - Video & Corsi



- Hewitt, Meijer and Szyperski: The Actor Model (everything you wanted to know...)  
[https://www.youtube.com/watch?v=7erJ1DV\\_Tlo&ab\\_channel=jasonofthel33](https://www.youtube.com/watch?v=7erJ1DV_Tlo&ab_channel=jasonofthel33)
- On .NET Live - Conway's Law, Microservices and Modular Monoliths with Microsoft Orleans  
<https://techcommunity.microsoft.com/t5/net-events/on-net-live-conway-s-law-microservices-and-modular-monoliths-ev-p/3790218>
- Architetture Distribuite - Microsoft Orleans e RabbitMQ  
<https://dev.marche.it/eventi/2023/07/12/architetture-distribuite-microsoft-orleans-e-rabbitmq-ancona-12-luglio-2023/>
- Introduction to Actor-based Development with Project Orleans - Chris Klug - NDC Oslo 2023  
[https://youtu.be/-NdkAW\\_NAb8?si=39SWluceoV1qUc7g](https://youtu.be/-NdkAW_NAb8?si=39SWluceoV1qUc7g)
- [Course] Introduction to Microsoft Orleans [2014]  
<https://www.pluralsight.com/courses/microsoft-orleans-introduction>
- [Course] Complete Microsoft Orleans .NET: From Zero to Hero [2019]  
<https://www.udemy.com/course/complete-orleans-net-from-zero-to-hero/>
- [Course] Actor Model Workshop [2023]  
<https://www.avanscoperta.it/it/training/actor-model-workshop/>





Per altri articoli nerd (e non solo)  
<https://blog.codiceplastico.com>





## GIANNI BOSSINI

Software Engineer @ [CodicePlastico](#)

[gianni.bossini@codiceplastico.com](mailto:gianni.bossini@codiceplastico.com)

**TW** [@bossinigianni](#) - **LK** [giannibortolobossini](#)

---



CODICEPLASTICO

ANALISI, SVILUPPO, FORMAZIONE, ASSESSMENT AZURE, UI & UX DESIGN

[www.codiceplastico.com](http://www.codiceplastico.com)





Grazie!

FOLLOW US

