iTDEVCON
EUROPEAN DELPHI CONFERENCE

SPACCHIAMO IL «MONOLITE»! 👌
(VERSO I MICROSERVIZI, MA SENZA PAURA)

Marco Breveglieri

ROME NOV 3/4

Marco Breveglieri

Sviluppatore software,
trainer e consulente

Compila Quindi Va
www.twitch.tv/compilaquindiva

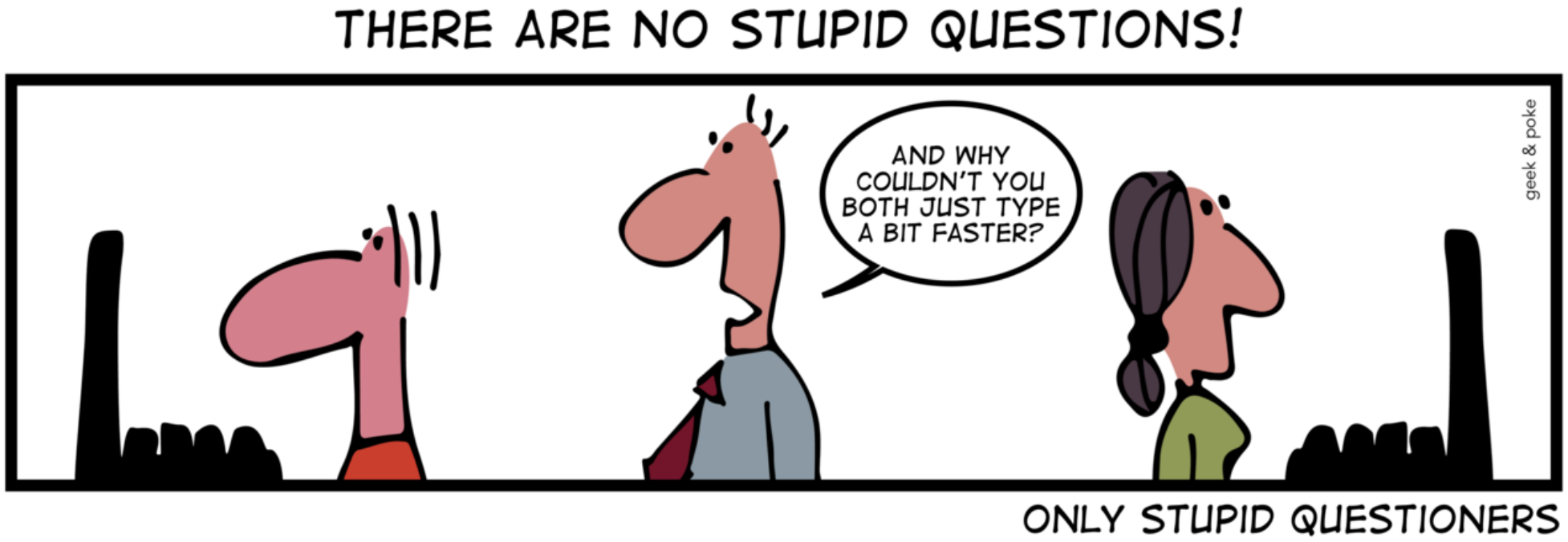
Delphi Podcast
www.delphipodcast.com

Blog personale
www.compilaquindiva.com

Social & more 👉 **www.breveglieri.it** 👈

Prima di iniziare…

Domande stupide: non esistono…



Domande stupide: non esistono…

Posso trasformare il mio gestionale Windows in «dei micro servizi»?

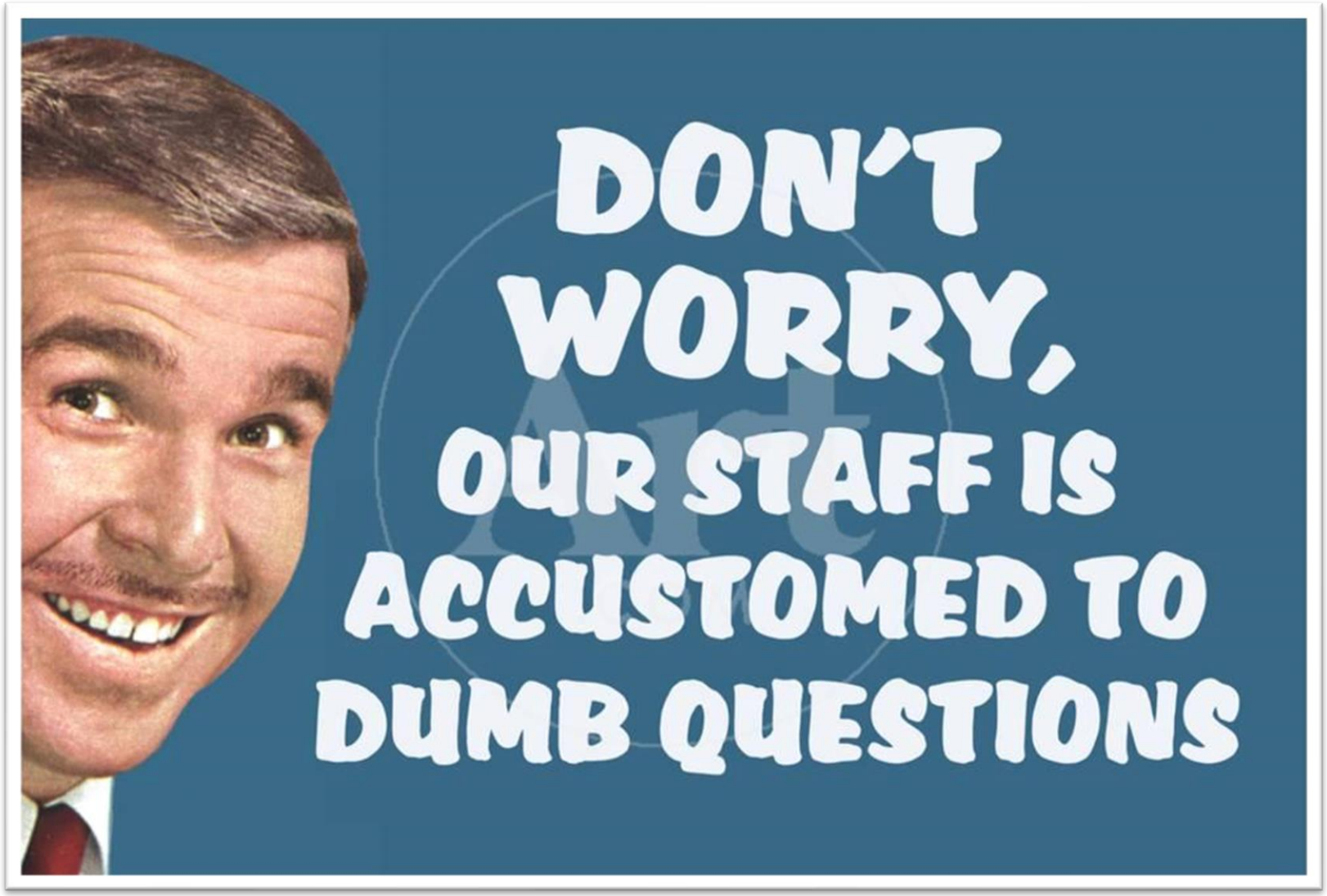
Come faccio a integrare una DLL fatta in .NET 6.0 nel mio programma?

Ho scalato tutto usando dei thread, però mi danno spesso delle Access Violation: come si tolgono?

Ma se trasformo tutto in svariate REST API, alla fine posso dire di avere dei microservice?

Dovevo mandare una mail quando salvo la fattura, ma ho risolto con una transazione: va bene?

Don’t worry!



Le reali esigenze

- Le domande nascondono in realtà problematiche concrete dello sviluppo software moderno
- Si tratta di esigenze ormai imprescindibili dell'attività di sviluppo del software
- Sono mosse dalla necessità di dover fornire risposte concrete alle esigenze di clienti e utenti


Velocità di esecuzione


Tempi di risposta ridotti


Robustezza e affidabilità


Integrazione dei sistemi


Monitoraggio e diagnostica


Evoluzioni e migliorie

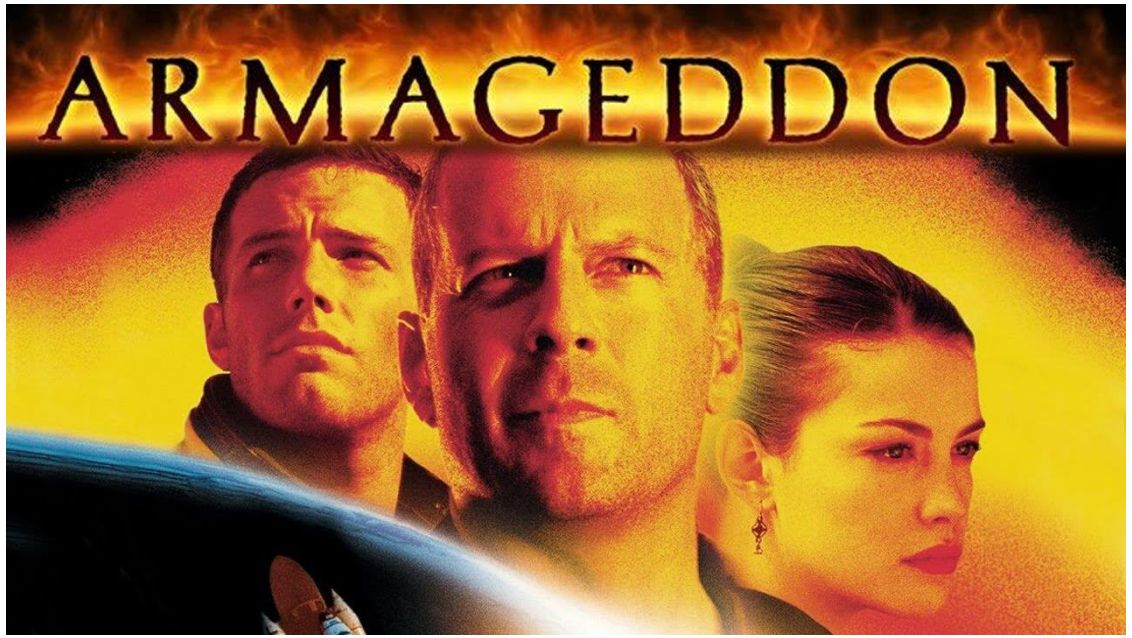

Sfruttamento delle skill


Business!

Il colpevole?

Dobbiamo romperlo!

Ci hanno provato in tanti…



Ce la faremo noi? 🤔

L’elefante nella stanza

Lui, il «monolite»

- Racchiude l'intera «business logic» all'interno di una unica e singola base di codice
- E' una delle architetture più semplici da realizzare
- Delphi ci ha abituato molto (troppo) bene a questa soluzione
- Spesso rispecchia l'organizzazione stessa dell'azienda

SOA (Service Oriented Architecture)?

- Rappresenta il primo passo verso l'abbandono di una architettura monolitica
- Spesso è costituito dallo spezzamento dell'architettura in diversi servizi, gestiti e installati separatamente
- **Può essere considerato come un «monolite distribuito»** 😢

Vantaggi e svantaggi

- Più semplice da sviluppare rispetto alle altre architetture
- Più facile da distribuire, poiché viene racchiuso in un unico file eseguibile
- Problemi di latenza di rete e sicurezza minori o assenti
- Gli sviluppatori non devono imparare applicazioni diverse: possono concentrarsi su un'unica applicazione

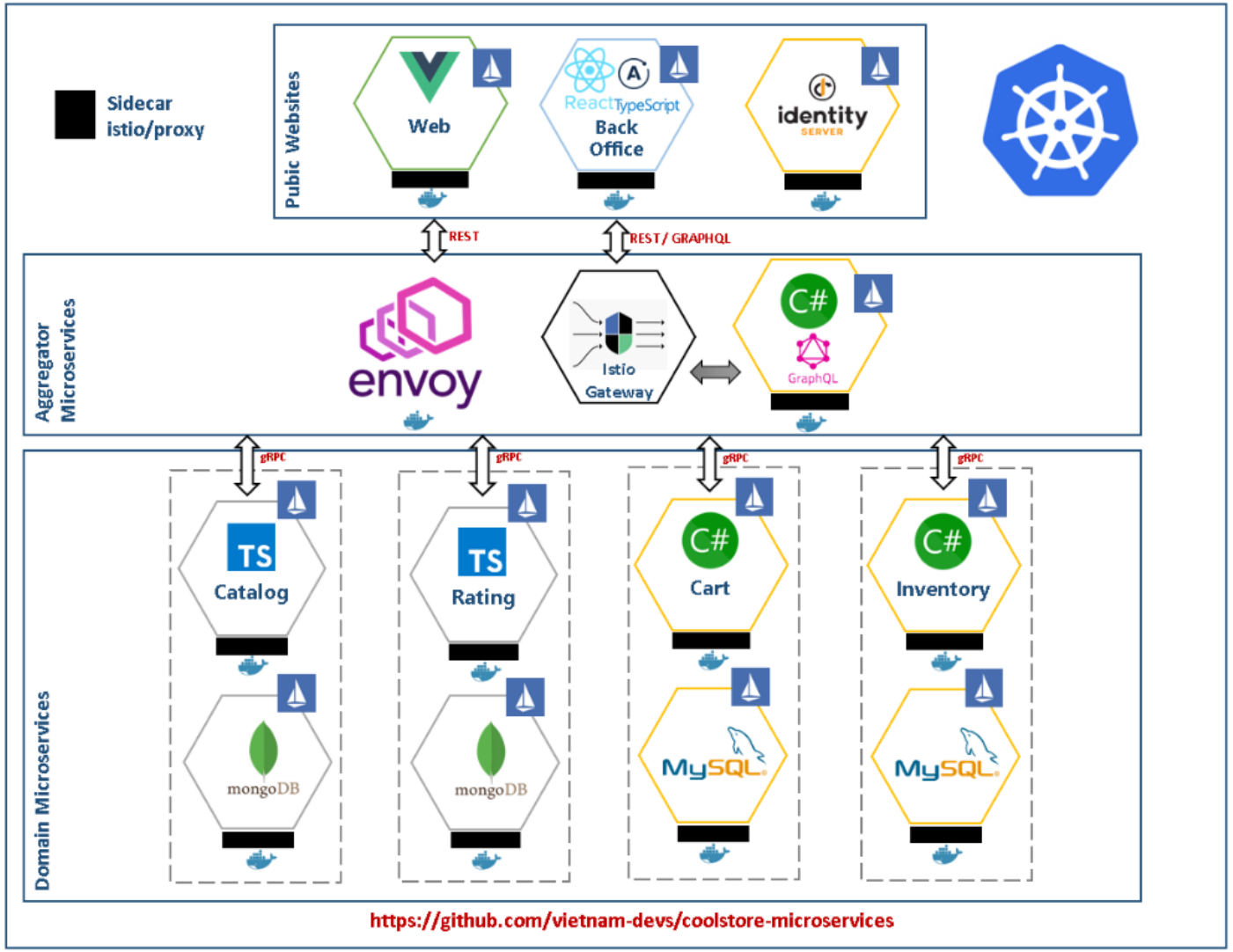
- Diventa troppo grande con il tempo e sempre più arduo da gestire
- Anche per una piccola modifica, dobbiamo distribuire l'intera app
- I tempi di distribuzione e avvio aumentano
- Qualunque sviluppatore si unisca al progetto, deve essere adeguatamente formato sul progetto
- Può richiedere più risorse inutilmente
- Ridimensionamento orizzontale difficoltoso (leggi: impossibile)
- Non è possibile differenziare le tecnologie perfette per ciascuna funzionalità
- Un singolo bug o instabilità può far cadere l'intera applicazione

Microservizi: una panoramica

Microservizi

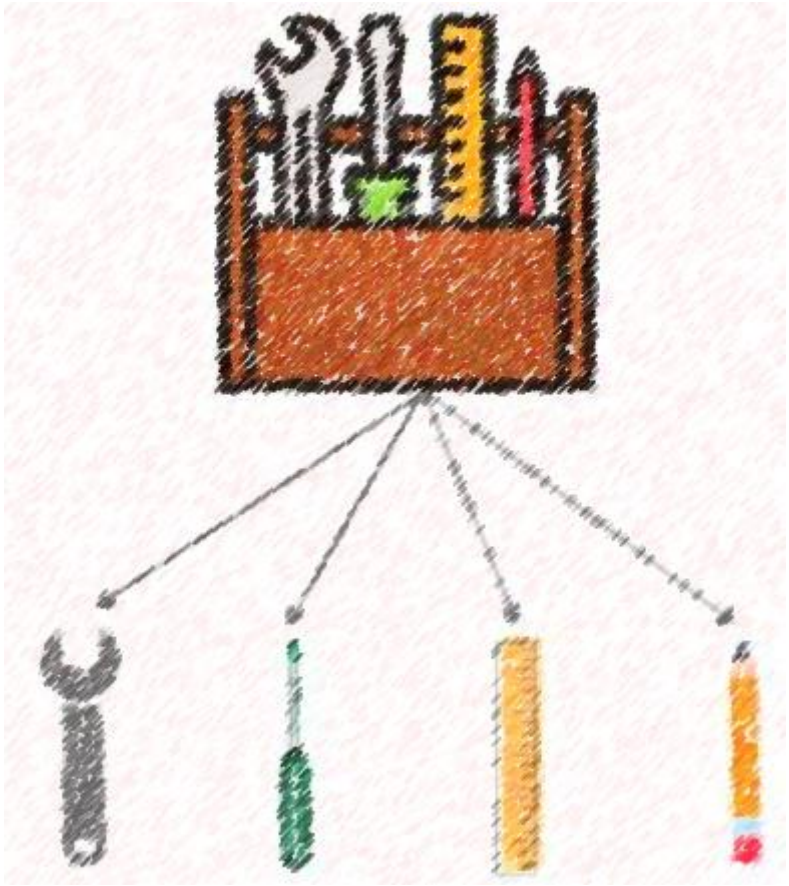
- Sistema federato di servizi numerosi e piccoli, liberamente accoppiati
- Ciascun sistema può essere distribuito in modo indipendente
- I singoli servizi sono individualmente mantenibili
- Spesso localizzati su nodi differenti, comunicano tramite protocolli sincroni o asincroni

Un esemplare



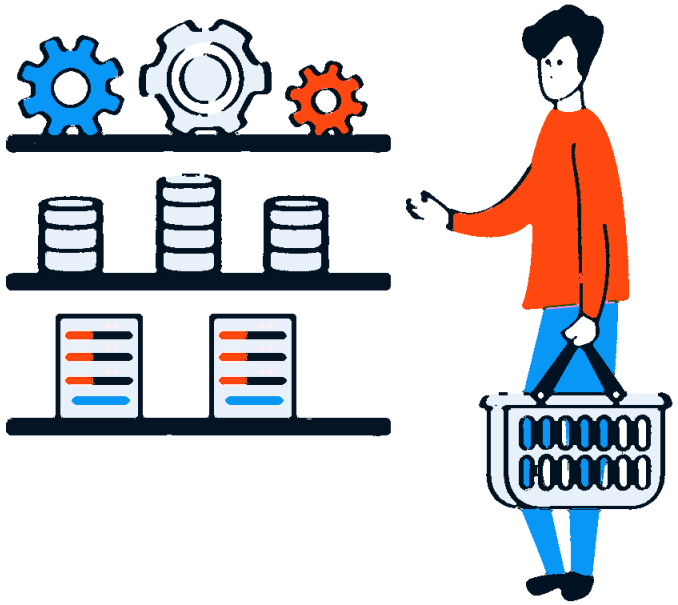
Calma... procediamo per step...

Single Responsibility Principle



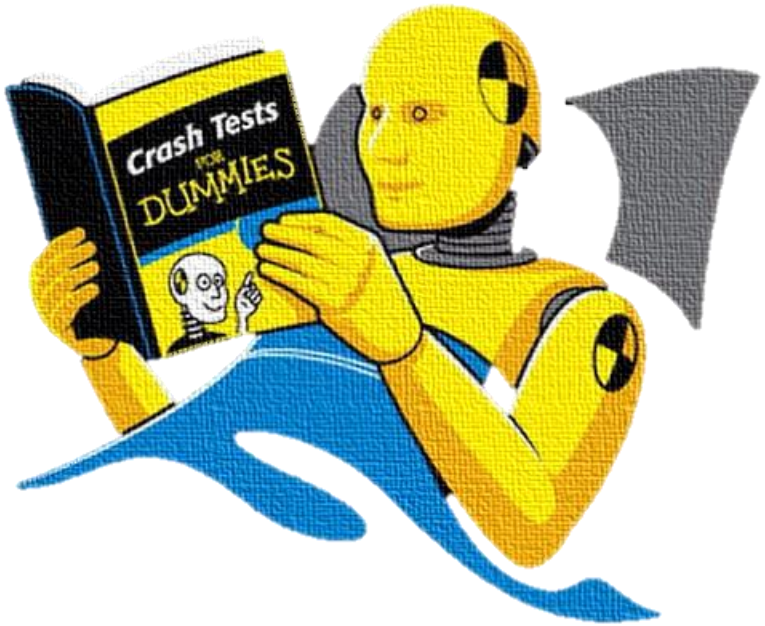
- Uno dei capisaldi dei principi SOLID, gli stessi che applichiamo nello sviluppo del software
- Una singola unità (classe, metodo e quindi anche microservizio) deve avere una unica responsabilità e fornire una sola funzionalità

Scegliere lo strumento giusto



- C’è sempre una tecnologia più adatta a implementare una determinata funzionalità
  - Delphi è un’ottima soluzione ma, come tutte le altre, non può essere il «top» per ogni necessità
- Siamo liberi di scegliere diverse piattaforme e tecnologie
  - L’approccio monolitico ci obbliga a dei compromessi
  - Con i microservizi, siamo guidati solo dai requisiti aziendali

Progettati per… fallire



- I microservizi devono essere progettati tenendo conto dei possibili casi di errore/fallimento
- Le problematiche relative a un singolo servizio non devono inficiare sulla stabilità del sistema
- L'eventuale indisponibilità di una funzionalità non deve influire sulle altre, che devono restare disponibili all'utente

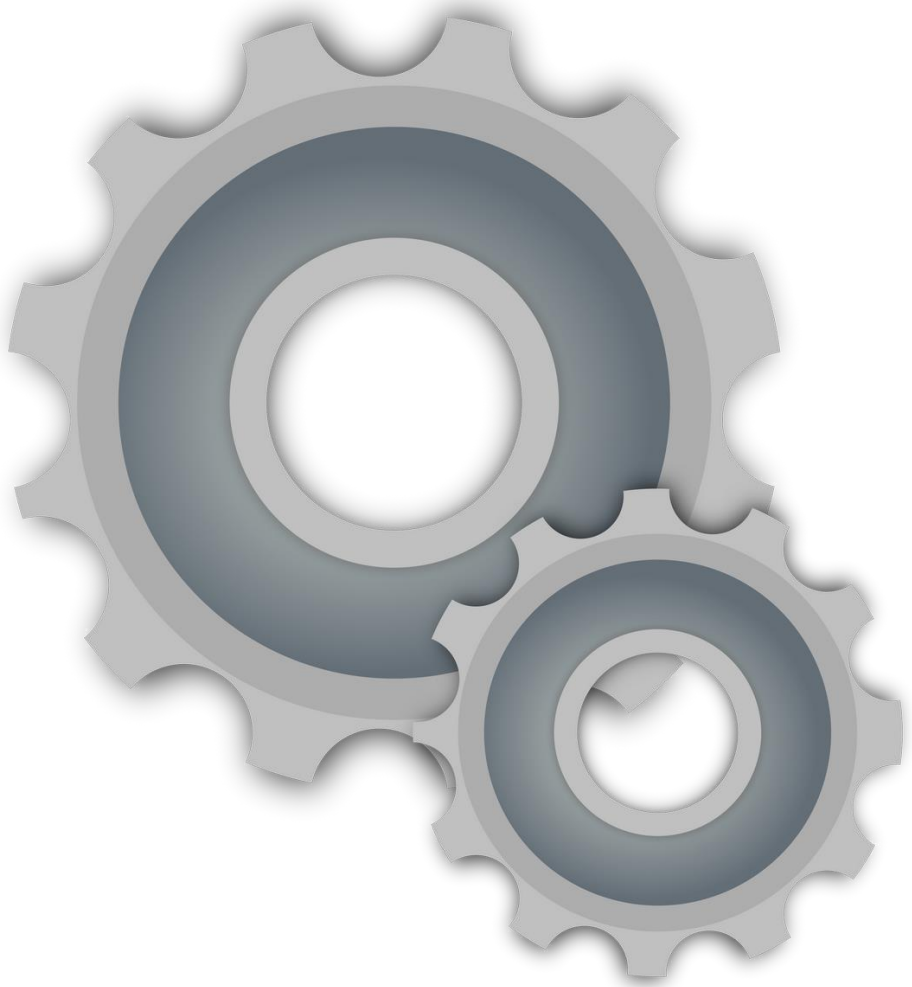
Vantaggi e svantaggi

- Sono facili da gestire in quanto (relativamente) più piccoli, autonomi e indipendenti
- Se sono necessari aggiornamenti a un microservizio, basta ridistribuire solo quello
- I loro tempi di avvio e implementazione sono mediamente inferiori
- Facilita l'integrazione del sistema da parte di un nuovo sviluppatore
- Se variano i requisiti di carico per una determinata funzionalità, basta ridimensionare solo il microservizio che se ne occupa (*aka* «horizontal scaling»)
- Ogni microservizio può utilizzare una tecnologia differente in base ai requisiti
- Se un microservizio particolare si interrompe, il sistema nel suo complesso rimane intatto e continua a fornire le proprie feature agli utenti

- Essendo un sistema distribuito, è molto più complesso tanto più è elevato il numero di servizi
- Gli sviluppatori devono avere le skill necessarie per operare con i microservizi e gestire la loro interoperabilità
- I microservizi hanno un costo legato alla latenza di rete a causa delle numerose chiamate di rete necessarie
- I microservizi sono meno sicuri per via delle necessarie comunicazioni di rete
- Il debug è difficoltoso poiché il flusso di elaborazione si distribuisce su più servizi e identificare l'errore può risultare più arduo

Suggerimenti pratici

Tipologie di applicazioni



Windows Service

Console Application

Comunicazione tra servizi /1

- **Socket tradizionale**
  - Troppo complesso, una «black box»
  - Tendenzialmente inefficiente

- **Protocollo HTTP (e derivati)** *API REST, API GraphQL, JSON/RPC*
  - Scelta tradizionale e diffusa
  - Ricorso a servizi e architetture come proxy e API gateway

Siamo sicuri che siano le alternative migliori per le nostre esigenze? 🤔

Comunicazione tra servizi /2

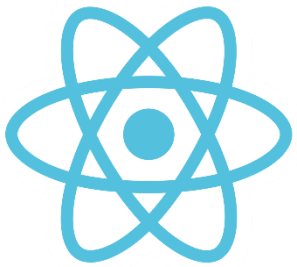
- **Sistema di messaggistica**
  - RabbitMQ
  - NATS
  - Altri MOM…

- *Alta affidabilità e robustezza*
- *Performance molto elevate*
- *Possibilità di clustering e gestione del fail over*
- *Protocolli di comunicazione semplici, estremamente noti e diffusi (es. Stomp)*
- *Iniziamo a **ragionare ad eventi!***

Crea il tuo «stack» ideale

Possibilità di combinare le tecnologie più adatte agli scopi!

- **Tecnologie per applicazioni Web**
  - Angular, React, Vue, Svelte
- **Backend e Web API (REST e non)**
  - .NET Framework, .NET Core e successivi, Node.JS
  - Delphi (DMVC, MARS, WiRL, RAD Server, …)
- **Data Processing, Machine Learning**
  - Python
  - R
- **Messaging**
  - Rabbit MQ
- **Database**
  - SQL Server, PostgreSQL, MySQL
  - MongoDB, Redis



…e tanti altri ancora!

Monitoriamo i servizi

Installiamo tool specifici per verificare le prestazioni dei servizi e tenere sotto controllo il sorgere di eventuali problematiche

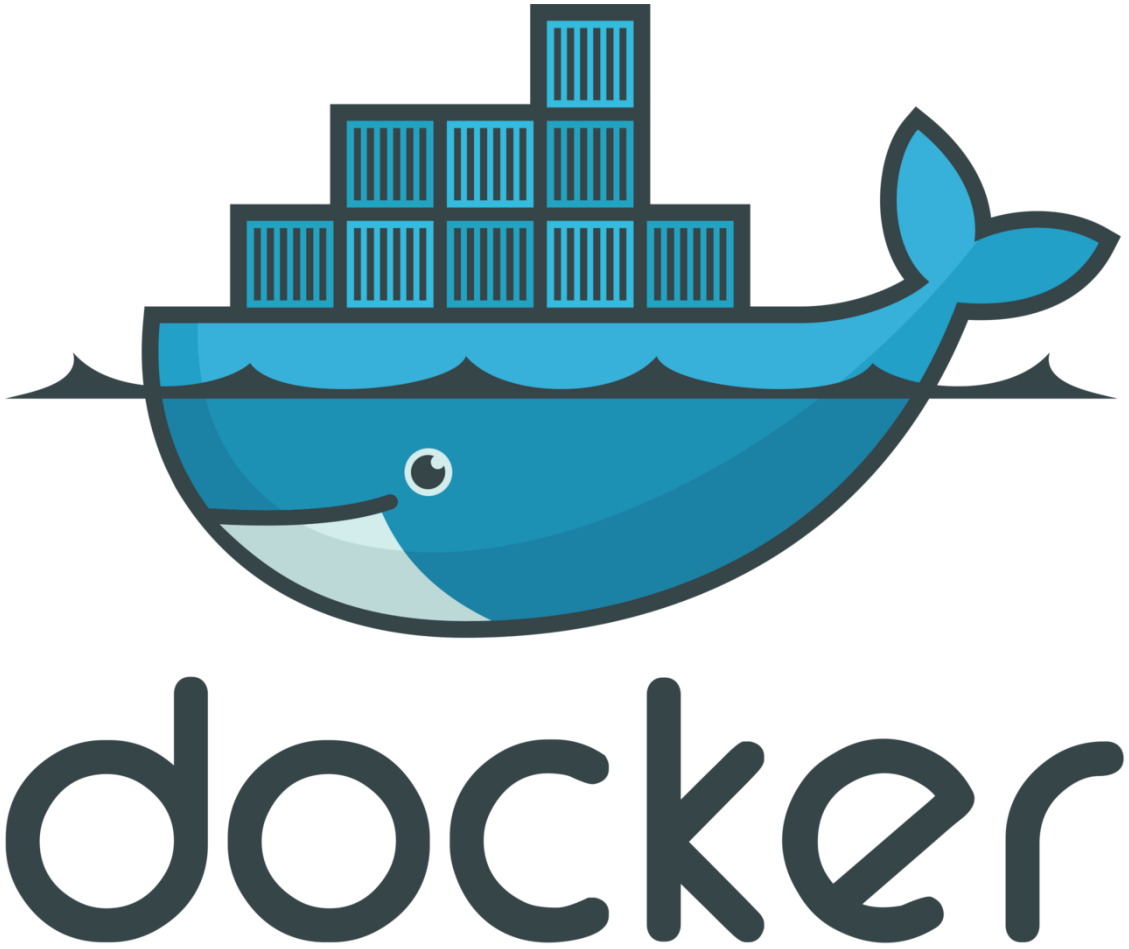
- Vedi il mio talk su *Prometheus* e *Grafana*


Demo?

Sto facendo bene?

- Se il tuo team può **distribuire un aggiornamento in qualsiasi momento** senza coordinarsi con altri team e viceversa.
- Occorre **predisporre un Integration Test** apposito per assicurare che tutti i microservizi funzionino insieme correttamente.
- **Quando qualcosa si rompe, i servizi accoppiati sono un inferno per il debug.** E quando viene rilevato il problema, risolverlo non è sempre facile come ripristinare un aggiornamento.
- **Passando ai microservizi, stai cambiando più del modo in cui codifichi: stai cambiando il modello operativo dell'azienda**
  - Devi imparare un nuovo stack tecnologico più complesso
  - Il management dovrà adattare la cultura del lavoro e riorganizzare le persone in team più piccoli e interfunzionali

Deploy?


docker


DEVCON
EUROPEAN
DELPHI
CONFERENCE
ROME22
3/4 NOV
DON'T
MISS
MY
SPEECH
www.itdevcon.it
Docker per
sviluppatori Delphi
Esploriamo le funzionalità di Docker a 360 gradi, scopriamo quali sono gli elementi della sua architettura, come si eseguono i comandi principali, in definitiva come è possibile sfruttare questo mirabolante strumento nella propria quotidianità di sviluppatori Delphi, ma non solo!
docker

Q & A


Thanks!