ARCHITECTURE PATTERNS

INGEGNERIA: processo ripetibile per arrivare a risultati certi

PATTERN: QUALCOSA CHE SI PUÒ RIPETERE.

diversi tipi di architettura: → PATTERN PER BACK-END

deployment	Logica
MONOLITI ✓	LAYERED ✓
MICRO SERVIZI	HEXAGONAL ✓
→ ARCHITECTURE	CLEAN-ARC
MODULITHS	EVENT-SOURCING
	EVENT-DRIVEN

sono tutte combinabili le deployabili e logiche

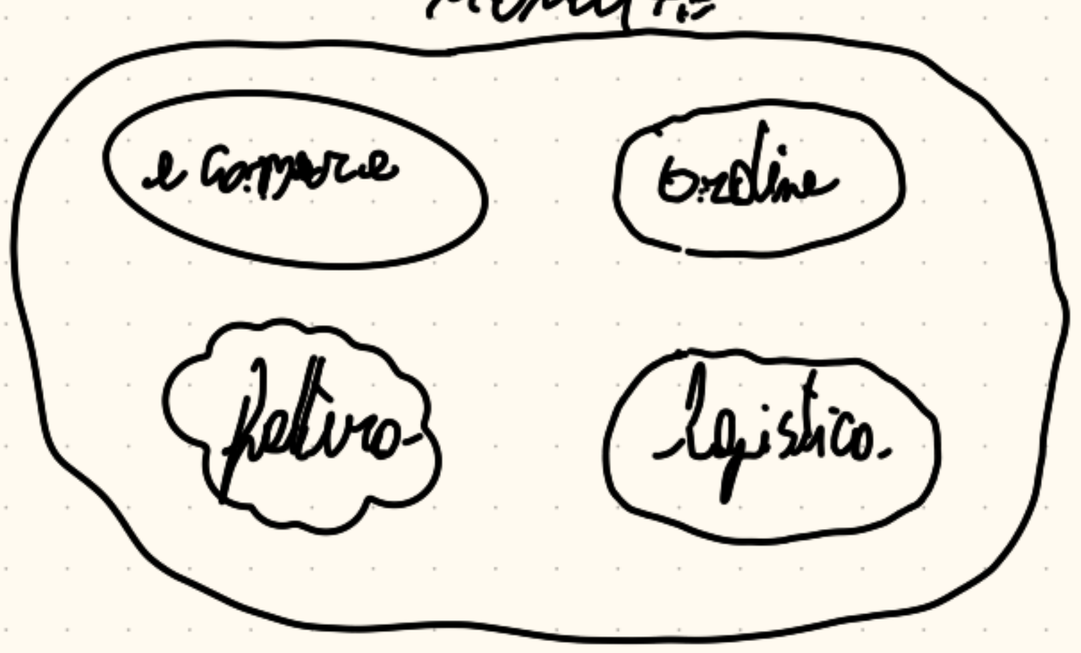
CONWAY LAW

→ L'architettura che scelgo rispecchia l'architettura e organizzazione della mia azienda.

MONOLITE (deployment) → L'artefatto è solo uno

↳ ogni volta che cambio un qualcosa su un settore devo fare un deploy su tutto

↳ AMAZON ad esempio deve rilasciare 24/7 non può permettersi di fare MONOLITE!



I requisiti non funzionali sono quelli che mi guidano per la scelta architetturale.

LAYER ARCHITECTURE ("LOGICHE")

MUAD: INSERIRE CHAR STRANI ← FAIL FIRST!

↑ USCITA PROGRAMMA INUTILE

Qualsiasi applicazione BACKEND INDIVIDUA 3 AZIONI:

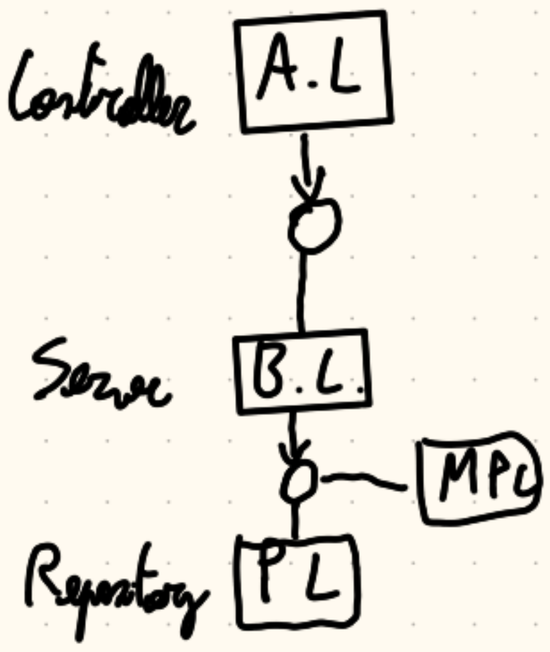
TIER

1. Application Logic → gestisce la applicazione dell'esterno → VERIFICHE SINTATTICHE SUI DATI
   ↳ esterno attore ATTIVO
2. Business Logic
3. Persistent Logic → ⚠ Attenzione è la più importante, il database ci permette di modellare tutto il resto

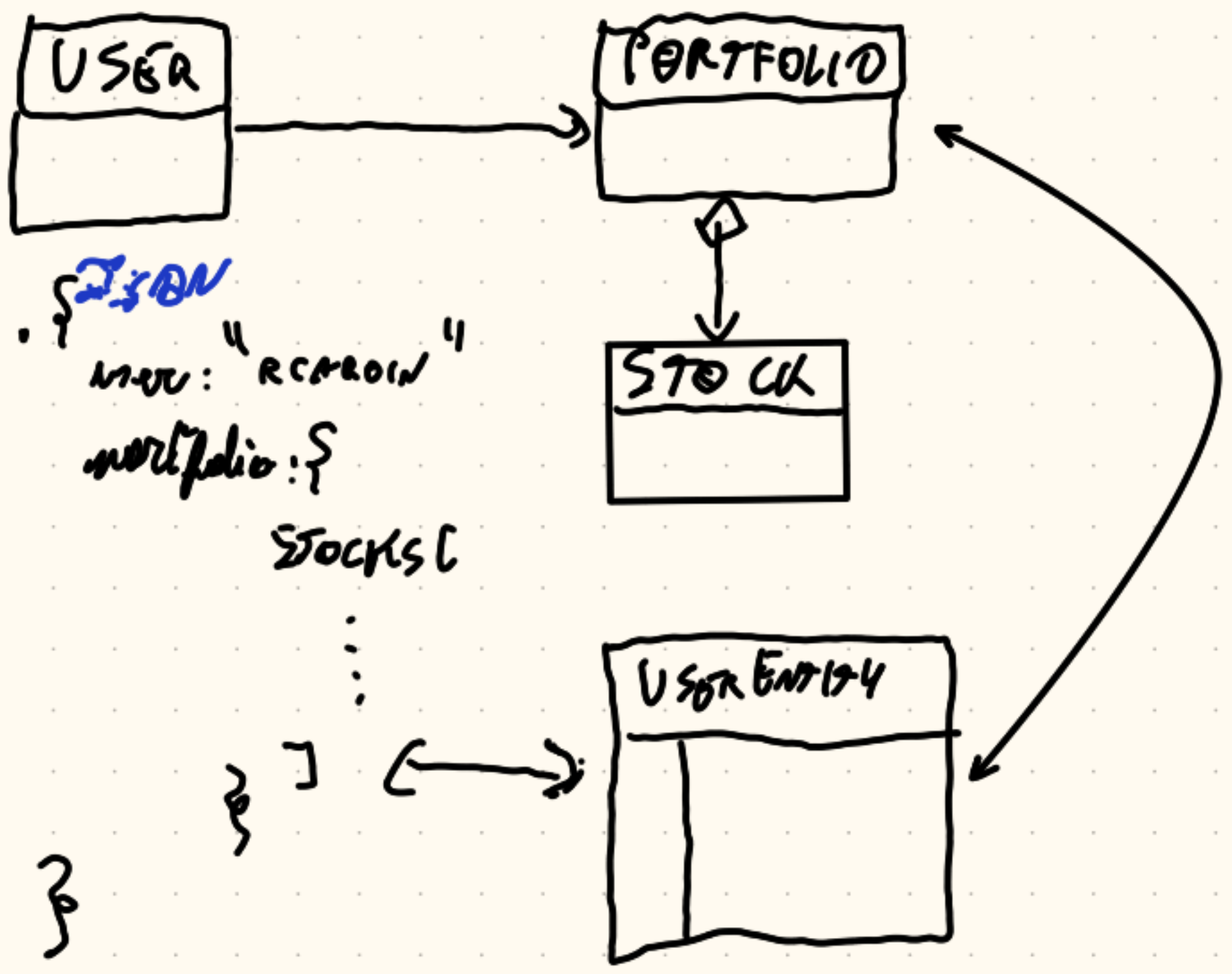
↓ esterno attore passivo

↳ Tutto ciò che mi permette di trattare database, connection error, fine spazio, eccezioni VARIE

↳ Tutti gli algoritmi sui quali si fonda il mio algoritmo

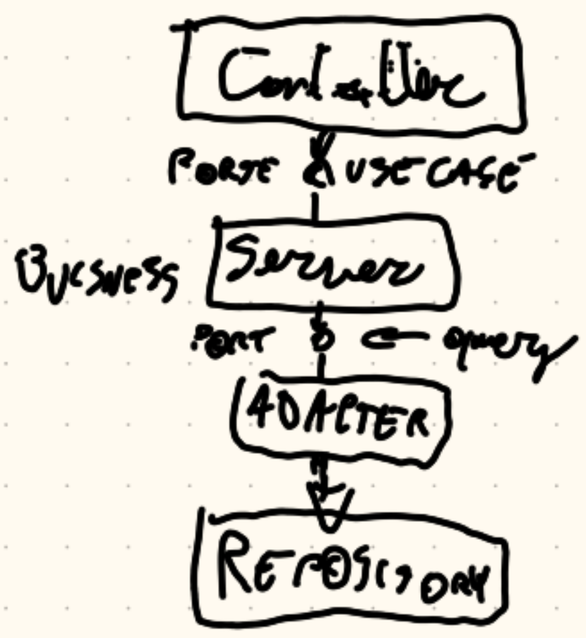


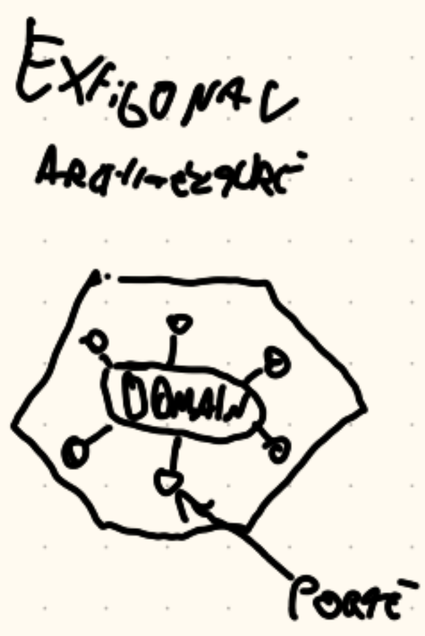
Metti caso che dobbiamo cambiare database da MySQL a Mongo, è un bel problema perché dovrà cambiare anche la BUISNESS LOGIC.



DDD → domain driven development

↳ MODIFICA LA PARTE DI BUISNESS, rendendola molto più modificabile, MA RICHIEDE MOLTO SOVRALLIVELLO





NAPSTER, ha smesso: nasce Spotify in un anno preso da 4 SVILUPPATORI
e. 200, NON applicano un modello gerarchico, ma applicano un
modello divide et impera, dividendo gli SVILUPPATORI in diversi
gruppi.

Si dice che il numero massimo di sviluppatori sia 6 e questo stesso
il tempo per sviluppare un prodotto. E ognuno di questi è fortemente
disinnescato

SPOTIFY → CREAZIONE dell'architettura a microservizi

ogni team si auto-
AUTOGESTISCE e gestisce un determinato
numero di servizi

+ CONTENUTO, + VELOCI A SVILUPPARE

- DIFFICILMENTE UN MICROSERVIZIO È AUTOSUFFICIENTE
- DIFFICILE GESTIONE DEGLI ERRORI "GESTIONE ROLLBACK"

SEMANTIC VERSIONING

	MAJOR	. MINOR	. FIX	
VERSIONE:	2	2	0	
	2	2	(1)	→ RISOLVO UN BUG
	2	(3) → AGGIUNGO UNA FUNZIONE	0	
MODIFICA NON RETROCOMPATIBILE	(3)	0	0	