# Progettazione architetturale

Bisogna aggiungere alla tabella dei vincoli il vincolo sul multipiattaforma che interferisce con l’usabilità.

## *Requisiti non funzionali*

Nell’analisi del problema sono emersi tre requisiti non funzionali, due dei quali impongono vincoli sul sistema:

* Sicurezza
* Usabilità
* Multi-piattaforma

Questi due requisiti sono purtroppo in netto contrasto tra di loro. La sicurezza imporrebbe una architettura di sicurezza a Layer, ma ciò ne peggiorerebbe l’usabilità e avrebbe la diretta conseguenza dell’effetto post-it nel primo cassetto della scrivania.

Considerata la tabella di valutazione dei beni e tutte le analisi svolte in precedenza si può trovare un equilibrio grazie alla suddivisione prevista tra amministratore e utente. L’utente avrà alta usabilità e bassa sicurezza, l’amministratore invece il contrario, essendo la sua figura unica e quindi di netta rilevanza per l’intero sistema. Verrà aumentata la sicurezza anche nella comunicazione con il sistema esterno

## *Scelta dell’architettura*

L’architettura più adatta a un sistema multipiattaforma è un’architettura client/server a tre livelli.

L1-Client

Viene sviluppato un Client per ogni Attore presente nel Sistema. Nel nostro caso specifico:

* Un client per l’utente non registrato
* Un client per l’utente
* Un client per l’amministratore

Il client dell’amministratore risiede su una macchina destinata al suo esclusivo accesso. I requisiti di sicurezza devono essere infatti i più stringenti possibile. Sulla stessa macchina risiederà la password dell’amministratore protetta con più chiavi di cifratura e accessibile solo dal controller della validazione credenziali. Dal client amministratore non si possono accedere alle funzionalità utente.

L2-Server

Il lato Server rispecchia la suddivisione fatta per i Client:

* Un server per l’utente insieme a tutte le operazione attuabili
* Un server per l’amministratore con tutte le operazioni attuabili

Si è deciso di non suddividere ulteriormente i Server a causa della forte interoperabilità necessaria tra i vari Controller. Più nello specifico, i vari controller associati all’utente operano sullo stesso dominio e separando il server in più componenti non verrebbe rispettato il *Single Responsability Principle* in quanto un Server potrebbe potenzialmente modificare i dati nel dominio di un altro Server. Viene inserito un Server dedicato all’amministratore per questioni di performance e divisione concettuale: ci si aspetta infatti che siano molte più le letture dai log che le rimozioni delle informazioni amministrabili.

L-3 Persistenza

Vengono inseriti due database, uno per log e uno per i restanti dati. Sia per i database di log che di dati sarà possibile accedere in lettura e scrittura. Per il database di log verranno utilizzati metodi di sicurezza più avanzati in quanto sarà utilizzato per eventuali situazioni di recovery.

Pattern & Design Principle

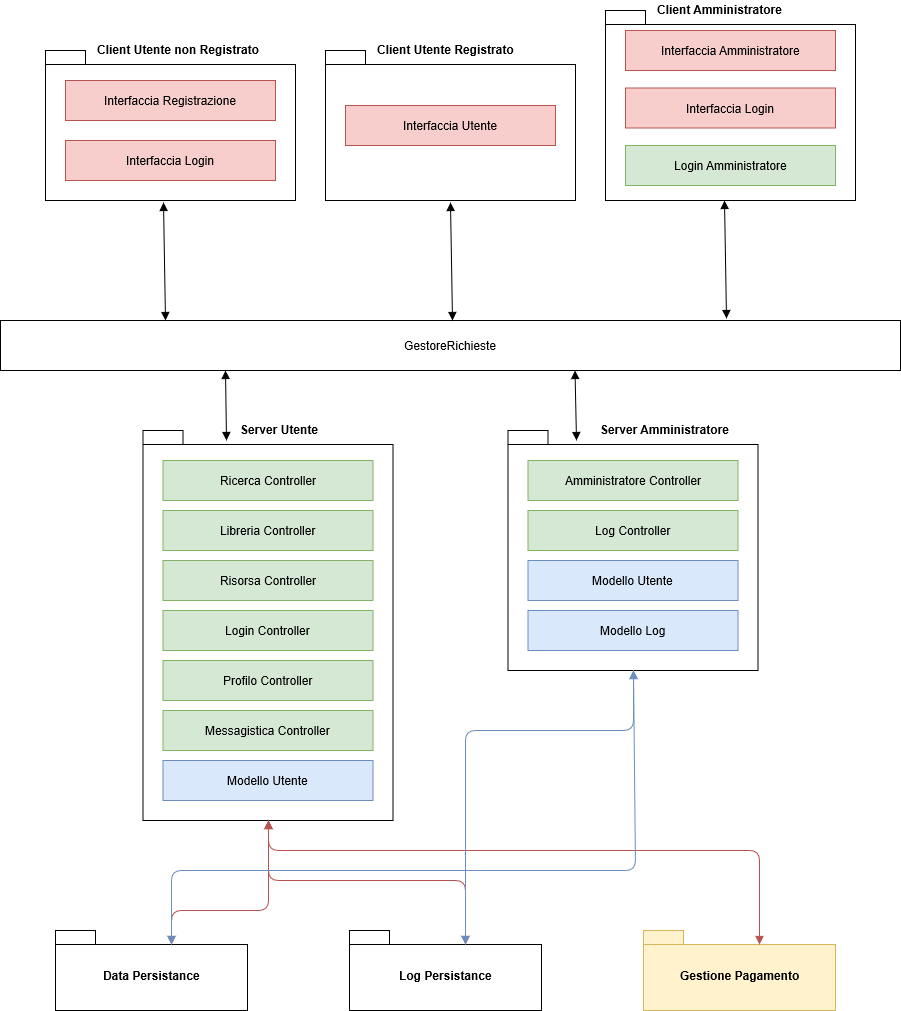
Verrà adottato il Pattern Broker per gestire lo stato della sessione dei vari utenti e per dare ulteriore protezione ai Server (soprattutto per gli utenti che hanno bassa sicurezza). La comunicazione con questo componente avverrà attraverso interfacce che verranno definite in conformità il *Dependency Inversion Principle*. Per garantire la sicurezza tra le comunicazioni cliente e servitore verrà utilizzato il protocollo TLS.

## *Scelta della tecnologia*

Il vincolo sull’usabilità e sul multipiattaforma porta a scegliere le tipiche tecnologie web:

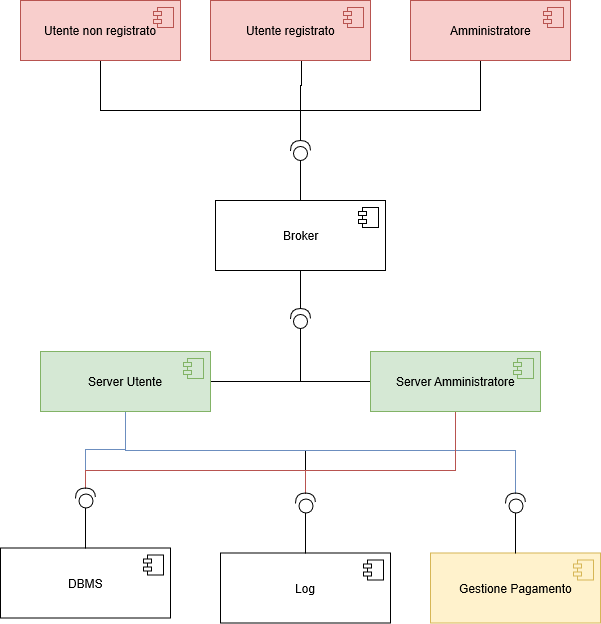
* Lato Client verranno utilizzati linguaggi di programmazione come: HTML, CSS, JavaScript. JavaScript sarà molto utile per la validazione dell’input, permettendo direttamente lato cliente di controllare i dati inseriti. HTML e CSS permetteranno usabilità su ogni browser di ogni macchina.
* Lato Server verranno sfruttate le potenzialità del Web Server Apache-Tomcat con conseguente utilizzo del linguaggio Java

Di seguito viene mostrata l’architettura del sistema sotto-forma di diagramma dei package



I colori inseriti nelle frecce non hanno alcun significato, servono solo a far capire meglio i collegamenti tra i vari package.

Questa è la struttura dal punto di vista dei componenti:



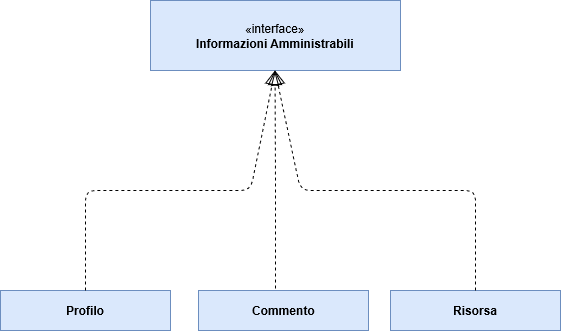
Anche in questo caso le frecce colorate servono solo a indicare meglio le direzioni dei collegamenti.

## Diagramma del dettaglio

Le uniche differenze nel dominio rispetto ai diagrammi precedenti sono le seguenti:

* Viene cancellato il dominio vendita. Non è espressamente richiesto dal committente la registrazione delle vendite. Inoltre, è superfluo anche dal punto di vista di eventuali recovery, perché di ogni azione viene tenuta traccia all’interno del log e l’informazione sarebbe quindi replicata.

Al dominio Utente viene aggiunta la seguente interfaccia:



Questo semplificherà la vita agli sviluppatori quando dovranno definire nuovi oggetti presenti all’interno del sistema. Ogni oggetto deve infatti essere controllabile dall’amministratore, e estendendo questa interfaccia diventa possibili (*Design for Change*).

Interfacce Server



Queste interfacce permettono di applicare il “*Dependency Inversion Principle”*. Basta infatti che future implementazioni rispettino l’interfaccia mostrata (rimuovere la i prima di String in IGestioneAmministratore)