

****

**System Design Document**

| **Riferimento** | C02\_SDD |
| --- | --- |
| **Versione** | 1.0 |
| **Data** | 06/01/25 |
| **Destinatario** | Prof.ssa F.Ferrucci, Prof F.Palomba |
| **Presentato da** | B. Gallo, A. De Filippo, G. Montella, M. Zurolo, B. Colella, A. Ceruso, V. Vernellati |
| **Approvato da** | L. Todisco, C. Venditto |

# Storia delle revisioni

| **Data** | **Versione** | **Descrizione** | **Autori** |
| --- | --- | --- | --- |
| 14/11/2024 | 0.1 | Prima Stesura, Creazione Sommario e tabelle | BC |
| 15/11/2024 | 0.2 | Creazione Design Goal | Tutto il team |
| 16/11/2024 | 0.3 | Revisione Design Goal | Tutto il team |
| 18/11/2024 | 0.4 | Revisione Design Goal, Aggiunta Trade off, Decomposizione in Sottosistemi, Mapping HW - SW | Tutto il team |
| 19/11/2024 | 0.5 | Diagramma Architetturale | Tutto il team |
| 20/11/2024 | 0.6 | Diagramma Architetturale per Gestione | Tutto il team |
| 21/11/2024 | 0.7 | Class Diagram | Tutto il team |
| 22/11/2024 | 0.8 | Modifica Class Diagram e Stesura Dizionario dei dati | AC ,BC ,ADF , BG, GM, MZ |
| 03/01/2025 | 0.9 | Modifiche globali | MZ, GM |
| 6/01/2025 | 1.0 | Revisione Globale | MZ, GM |

# Membri del team

| **Nome** | **Acronimo** | **Informazioni di contatto** |
| --- | --- | --- |
| Leopoldo Todisco | LT | l.todisco4@studenti.unisa.it |
| Carlo Venditto | CV | c.venditto@studenti.unisa.it |
| Biagio Gallo | BG | b.gallo9@studenti.unisa.it |
| Geraldine Montella | GM | g.montella19@studenti.unisa.it |
| Alessia De Filippo | ADF | a.defilippo31@studenti.unisa.it |
| Mario Zurolo | MZ | m.zurolo3@studenti.unisa.it |
| Vito Vernellati | VV | v.vernellati@studenti.unisa.it |
| Antonio Ceruso | AC | a.ceruso9@studenti.unisa.it |
| Benedetta Colella | BC | b.colella@studenti.unisa.it |

# 

# 

# 

Sommario

[**Storia delle revisioni 2**](#_heading=h.v0p1lzt9zsl3)

[**Membri del team 3**](#_heading=h.2et92p0)

[**1 - Introduzione 5**](#_heading=h.3dy6vkm)

[1.1 - Scopo del sistema 5](#_heading=h.1t3h5sf)

[1.2 - Design goals e tradeoff 6](#_heading=h.7ow3n0rf7n7k)

[1.3 - Definizioni, acronimi e abbreviazioni 8](#_heading=h.mmyv7n1o3n1t)

[1.4 - Riferimenti 9](#_heading=h.hjx0w4gf9i0n)

[1.5 - Overview 9](#_heading=h.o013wk41zbjz)

[**2 - Architettura del sistema corrente 10**](#_heading=h.mlfpjaj8yc0k)

[**3 - Architettura del sistema proposto 11**](#_heading=h.qx907bl5t8fb)

[3.1 - Panoramica 11](#_heading=h.g4lkopecyues)

[3.2 - Decomposizione in sottosistemi 12](#_heading=h.owc17xfuh190)

[Diagramma Architetturale 14](#_heading=h.tctig6j1pu11)

[3.2.1- Gestione Utente 15](#_heading=h.6bu773xxu34e)

[3.2.2- Gestione Alloggio 16](#_heading=h.b3hpfmhf174x)

[3.2.3- Gestione Annuncio 17](#_heading=h.g3pj2lqbbppu)

[3.2.4- Gestione Corso 18](#_heading=h.bmfuuj2f1gei)

[3.2.5- Gestione Evento 19](#_heading=h.bss8ul3nlhi4)

[3.2.6- Gestione Comunicazione 20](#_heading=h.7bkmpruq3tpd)

[3.2.7- CDN 21](#_heading=h.a7qus4gc2y0p)

[3.3 - Mapping Hardware/Software 22](#_heading=h.917xp9x23lff)

[3.4 - Gestione dati persistenti 23](#_heading=h.tjm7i2go8obh)

[- Database Relazionale (MySQL) 23](#_heading=h.a8nctbjbn0kb)

[3.4.1 - Class Diagram ristrutturato 24](#_heading=h.me90ezsskv9r)

[**4 - Glossario 25**](#_heading=h.pt0b0xlo9kc0)

# 

# 1 - Introduzione

## 1.1 - Scopo del sistema

Bridge è una piattaforma digitale concepita per promuovere l’accoglienza e l'integrazione di rifugiati e migranti in Italia, facilitando l'incontro con risorse e opportunità messe a disposizione dalla comunità locale.

Mira a migliorare l'integrazione sociale ed economica dei rifugiati attraverso la gestione e l'assegnazione ottimizzata di alloggi e corsi di formazione. I rifugiati e i volontari potranno interagire direttamente tramite una chat dal vivo, inoltre il rifugiato avrà a disposizione un chatbot per ricevere supporto immediato e informazioni utili.

Bridge si propone così come un punto di riferimento per la creazione di connessioni reali e solide tra chi arriva e chi accoglie, favorendo l'inclusione sociale e l'autosufficienza dei rifugiati.

## 1.2 - Design goals e tradeoff

| **Rank** | **ID design goal** | **Descrizione** | **Categoria** | **RNF di origine** |
| --- | --- | --- | --- | --- |
| **1** | **DG\_1** | Il sistema deve prevedere un’interfaccia che rispetti le otto regole d'oro di Ben Shneiderman e che consenta agli utenti, indipendentemente dal loro livello di competenza tecnologica, di completare le operazioni principali in meno di 15 minuti senza necessità di assistenza. | End User Criteria | RNF\_US\_1  RNF\_US\_2 |
| **2** | **DG\_2** | Progettare il sistema per garantire la sicurezza dei dati e delle comunicazioni, utilizzando la cifratura delle password tramite i metodi nativi offerti da Spring Boot. La gestione dei token è integrata con filtri che validano e controllano l'accesso alle risorse protette, mentre la configurazione delle politiche CORS permette richieste sicure da domini autorizzati, contribuendo a preservare l'integrità e la riservatezza delle informazioni degli utenti. | Dependability | RNF\_AFF\_2  RNF\_AFF\_4  RNF\_INT\_4 |
| **3** | **DG\_3** | L'interfaccia grafica deve supportare la visualizzazione corretta su schermi con risoluzioni comprese tra 320x480 e 2560x1440 pixel | End User Criteria | RNF\_IM\_1 |
| **4** | **DG\_4** | Il sistema garantisce un codice facilmente manutenibile e comprensibile, progettato con una struttura modulare e scalabile. Grazie all'adozione di principi SOLID e agli standard di programmazione, si favorisce la leggibilità, la collaborazione tra sviluppatori e la facilità di aggiornamento o estensione delle funzionalità | Maintainability | RNF\_ST\_1  RNF\_ST\_2  RNF\_OP\_1 |
| **5** | **DG\_5** | Progettare un sistema per la gestione dei file che includa l'importazione e l'esportazione dei file in formato .PDF con una dimensione massima di 8 megabyte, il caricamento di immagini in formato .jpeg per profili utenti ed alloggi con una dimensione massima di 8 megabyte. In caso di errore notifica, tramite un messaggio chiaro, all'utente il motivo | End User Criteria | RNF\_INT\_1  RNF\_INT\_2  RNF\_INF\_3 |
| **6** | **DG\_6** | Il sistema deve garantire che i tempi di risposta siano inferiori a 10 secondi e che almeno l'80% delle richieste degli utenti riceva risposta entro 10 secondi, anche sotto carico elevato, con il chatbot statico progettato per rispettare questi vincoli e che faccia riferimento al documento SLA di OpenAI | Performance | RNF\_INT\_5  RNF\_PR\_1  RNF\_PR\_3  RNF\_OP\_3 |
| **7** | **DG\_7** | Progettare un sistema che assicuri un livello di disponibilità dei servizi, pari almeno al 90%, minimizzando i tempi di inattività. | Performance | RNF\_AFF\_1 |
| **8** | **DG\_8** | Progettare un sistema che in caso di aggiornamento utilizzi una pipeline CI/CD che consenta il testing automatico e il deployment senza interruzioni dei servizi o errori nell’esecuzione, per migliorare la velocità e la qualità dei rilasci. | Dependability | RNF\_OP\_2 |
| **9** | **DG\_ 9** | Progettare un sistema che permette di offrire un corretto funzionamento anche con 1000 utenti collegati concorrentemente | Performance | RNF\_PR\_2 |
| **10** | **DG\_10** | Il modulo di intelligenza artificiale, sviluppato in python, deve poter gestire un massimo di 1000 richieste simultanee garantendo tempi di predizione in 2 secondi. | Performance | RNF\_IM\_2 |

| **Trade off** | **Descrizione** |
| --- | --- |
| SICUREZZA  **VS**  PRESTAZIONI | **CONFLITTO**:L’implementazione di politiche di sicurezza avanzate (crittografia delle password, gestione dei token e politiche CORS) può aumentare la latenza del sistema.  **SOLUZIONE:** Ottimizzare la gestione dei token e la validazione con filtri centralizzati e ridurre la complessità delle politiche CORS mantenendo la sicurezza per i domini principali, prioritizzando le prestazioni. |
| INTERFACCE INTUITIVE  **VS**  PRESTAZIONI | **Conflitto**: Progettare un'interfaccia intuitiva che consenta agli utenti di accedere rapidamente alle proprie aree di interesse (es. area personale, annunci, ecc.) richiede l'implementazione di layout personalizzati, percorsi utente chiari e funzionalità interattive. Questo può introdurre complessità che aumenta il carico sul front-end e il tempo di rendering.  **SOLUZIONE**:Implementare una navigazione gerarchica semplice e pulsanti rapidi per le aree chiave. Maggiore attenzione per garantire che gli utenti possano navigare rapidamente e senza difficoltà verso le aree di interesse principali, cercando di ottimizzare le performance utilizzando tecnologie come il rendering lato server o il pre-rendering, limitando le interazioni complesse a casi strettamente necessari. |
| TEMPI DI RILASCIO  **VS**  QUALITA’ | **CONFLITTO**: garantire un’alta qualità del software richiede tempi di sviluppo più lunghi e tediosi, nonché una maggiore attenzione ai dettagli, il che può rallentare i tempi di rilascio.  **SOLUZIONE**: in caso di scadenze particolarmente strette sarà data priorità al rispetto delle scadenze di rilascio prefissate, adottando un approccio che consenta di rilasciare software con eventuali bug o imperfezioni minori, da risolvere successivamente tramite aggiornamenti. |

## 1.3 - Definizioni, acronimi e abbreviazioni

● **Design Goal (DG)**: obiettivi di design per il sistema proposto;

● **Design Trade-off**: scelte e compromessi tra design goals;

● **Deployment Diagram**: diagramma UML di specifica per le relazioni tra le componenti;

● **Three-tier**: architettura software che divide l’applicazione in tre strati: presentazione, persistenza e logica;

● **SDD**: System Design Document;

● **RNF**: Requisito non funzionale;

## 1.4 - Riferimenti

●  [Statement of Work](https://drive.google.com/file/d/1lI7-PPEv1z373S7F6sfjZjYJFmtTBTfH/view?usp=drive_link)

●  [Matrice di tracciabilità](https://docs.google.com/spreadsheets/d/15nmHqjWaqj7P9LyKv4gSCzlvh9GUT5eg/edit?usp=drive_link&ouid=112179462567237080194&rtpof=true&sd=true)

●  [Requirements Analysis Document](https://docs.google.com/document/d/13qPqBlr6AQ5HTP-HkMtEVIdEKVk3_34R/edit?usp=drive_link&ouid=112179462567237080194&rtpof=true&sd=true)

●  [Cots&DesignPatternDocument](https://docs.google.com/document/d/1u83xNIdOVI1lTE2rx6KJbOymEepipBQk/edit?usp=drive_link&ouid=115953621529974241157&rtpof=true&sd=true)

## 1.5 - Overview

Il System Design Document consta di quattro sezioni:

1. **Introduzione:** viene descritto lo scopo del sistema e gli obiettivi di design;
2. **Architettura del Sistema Corrente:** vi è una descrizione dell’architettura già presente;
3. **Architettura del Sistema Proposto:** è descritto come il sistema proposto sarà composto. Si andrà a dividere il sistema in sottosistemi, verrà eseguito il mapping hardware/software e verrà illustrato il class diagram del sistema da implementare.
4. **Glossario:** contiene la lista dei termini usati nel documento e il loro significato.

# 

# 2 - Architettura del sistema corrente

Non esiste un sistema attuale, per cui non è possibile individuare un’architettura del sistema corrente.

L'unica eccezione riguarda la Content Delivery Network (CDN), per la quale abbiamo preso spunto dal funzionamento di Airbnb. Tale ispirazione si è limitata esclusivamente al concetto e alla strategia di ottimizzazione nella distribuzione dei contenuti multimediali, integrando un approccio efficace per garantire elevate prestazioni e scalabilità. La scelta di ispirarsi ad Airbnb è stata dettata dalla necessità di gestire alloggi, una funzionalità condivisa tra il nostro sistema e il loro. Conseguentemente, si è deciso di simulare il comportamento per quanto riguarda la gestione dei contenuti multimediali tramite CDN, adattando tale approccio anche per altre necessità del sistema, come la gestione di documenti e immagini.

Al momento, non esiste un sistema attuale che possa fungere da base per l’analisi. Di conseguenza, non è possibile individuare né descrivere un’architettura del sistema corrente.

# 3 - Architettura del sistema proposto

## 3.1 - Panoramica

L’architettura di riferimento per il sistema proposto è un’architettura Three-Tier scelta per le sue caratteristiche di semplicità, versatilità ed efficienza. Questa architettura offre un equilibrio ideale tra i tempi di consegna del progetto e l’implementazione dei requisiti funzionali, garantendo al contempo il rispetto della maggior parte dei design goal definiti.

Uno dei suoi principali vantaggi è la gestione ottimale della sicurezza, poiché il client non accede direttamente ai dati memorizzati nel database, ma interagisce attraverso il livello intermedio della logica di business. Questo approccio riduce le vulnerabilità e migliora la protezione dei dati.

Inoltre, l’architettura Three-Tier si distingue per la sua efficienza in termini di performance e portabilità. La separazione tra il livello di presentazione e quello della logica di business semplifica l’adattamento del sistema a diverse piattaforme in futuro, rendendo più agevole l’espansione o la migrazione verso nuovi ambienti tecnologici.

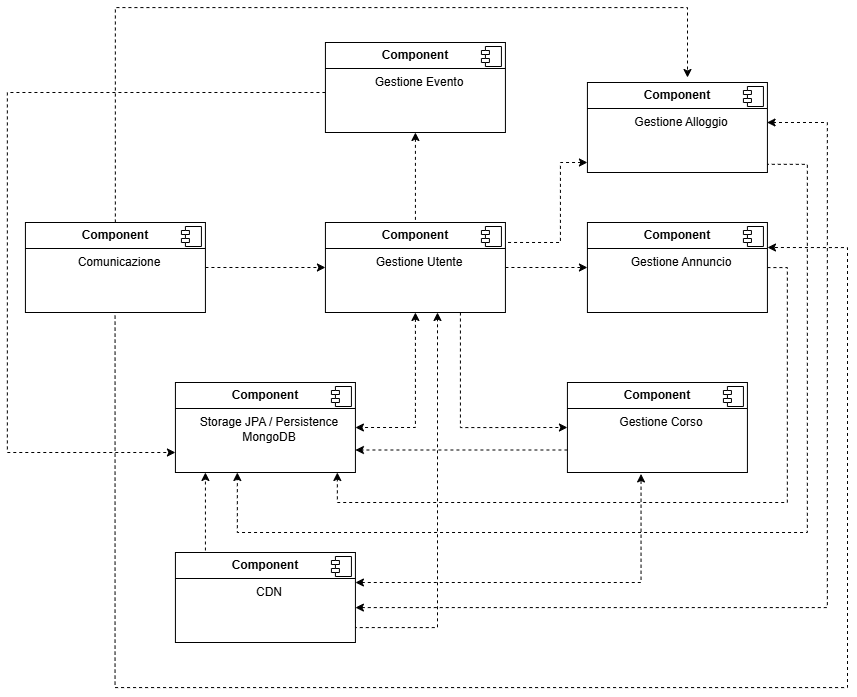
## 

## 

## 

## 3.2 - Decomposizione in sottosistemi

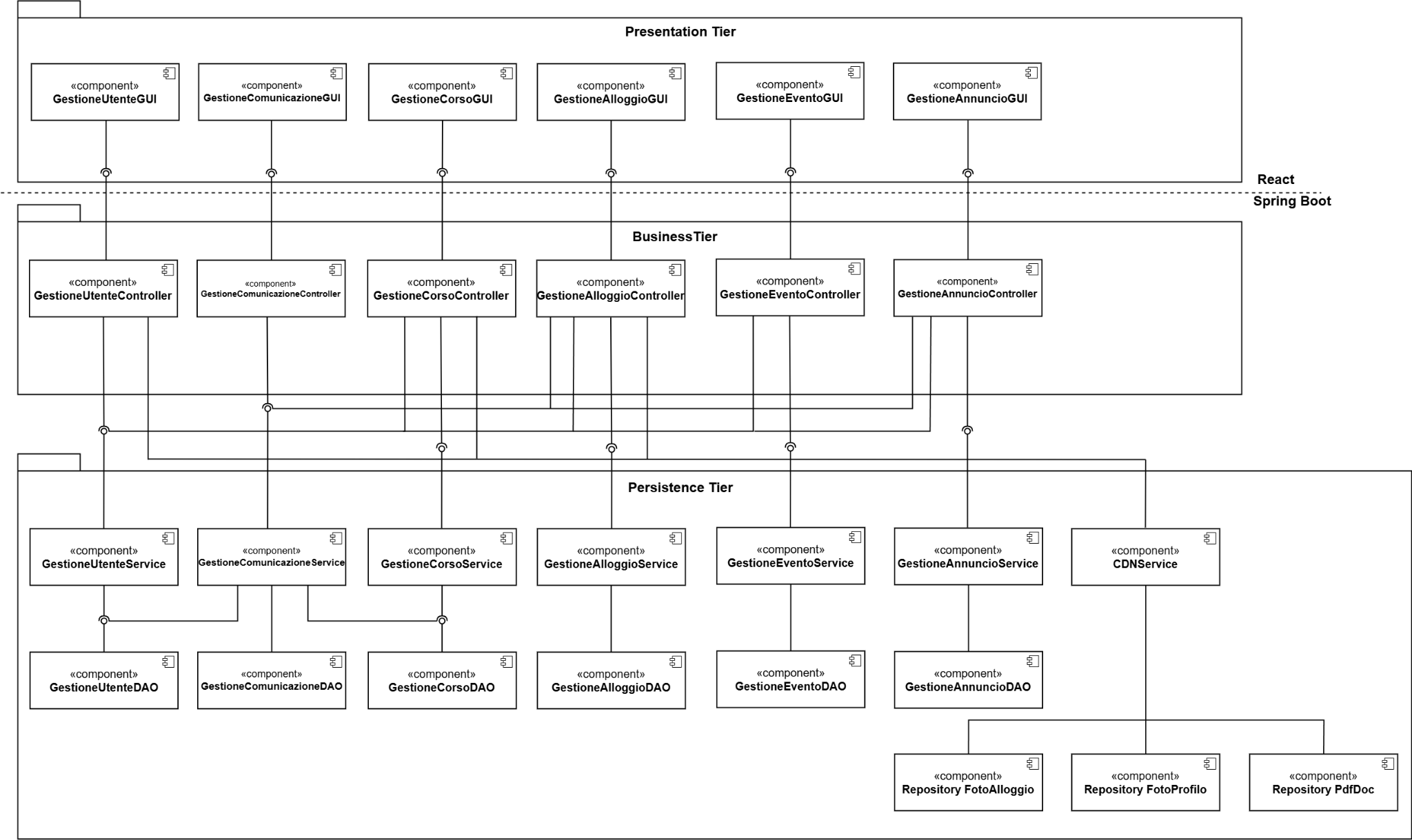
* **Gestione Utente**: si occupa delle funzionalità di login, logout e registrazione. Inoltre si occuperà anche della gestione dei dati personali degli account, della loro modifica e visualizzazione. Inoltre si occuperà dell'eliminazione dell'account stesso.
* **Gestione Annuncio**: si occupa della creazione, modifica ed eliminazione di un annuncio. Inoltre gestisce anche le funzionalità tipiche di un annuncio di lavoro, come la candidatura allo stesso. Gestisce anche le funzionalità di un annuncio di consulenza con annessa somministrazione della preferenza da parte di un Rifugiato, visualizzazione, accettazione e utilizzo della chat come mezzo di comunicazione da parte della Figura Specializzata
* **Gestione Evento**: si occupa della creazione di un evento da parte di un Volontario. Inoltre gestisce le funzionalità di iscrizione e disiscrizione all'evento da parte di un Rifugiato.
* **Gestione Alloggio**: si occupa della gestione degli alloggi e delle funzionalità ad essi legate, ovvero inserimento e assegnazione, nonché la possibilità di manifestare interesse per un alloggio da parte di un rifugiato;
* **Gestione Corso:** si occupa della gestione dei corsi e delle funzionalità ad essi legate, ovvero inserimento, download contenuto;
* **Gestione CDN**: si occupa della gestione dei file multimediali attraverso un sistema di archiviazione in cloud (CDN), con particolare attenzione alla gestione di immagini e documenti. Fornisce le funzionalità per il salvataggio, il recupero e l'eliminazione di foto di profilo, foto relative agli alloggi e documenti PDF, permettendo l'archiviazione sicura e l'accesso a questi file. Questo modulo interagisce con il database MongoDB per la persistenza dei dati, garantendo un'adeguata gestione dei file e l'efficienza nelle operazioni di memorizzazione e recupero.
* **Comunicazione:** si occupa della gestione della chat tra rifugiato e volontario, nonché del sistema di notifiche via email. Inoltre si occupa della gestione del chatbot di supporto utente, andando a definire il sistema di risposte statico ed il supporto dinamico;
* **Storage JPA / Persistence:** si occupa della gestione dei dati dei vari sottoinsiemi con l'uso del modulo Java Spring, nonché delle interazioni con il database relazionale;



## 

## 

## Diagramma Architetturale

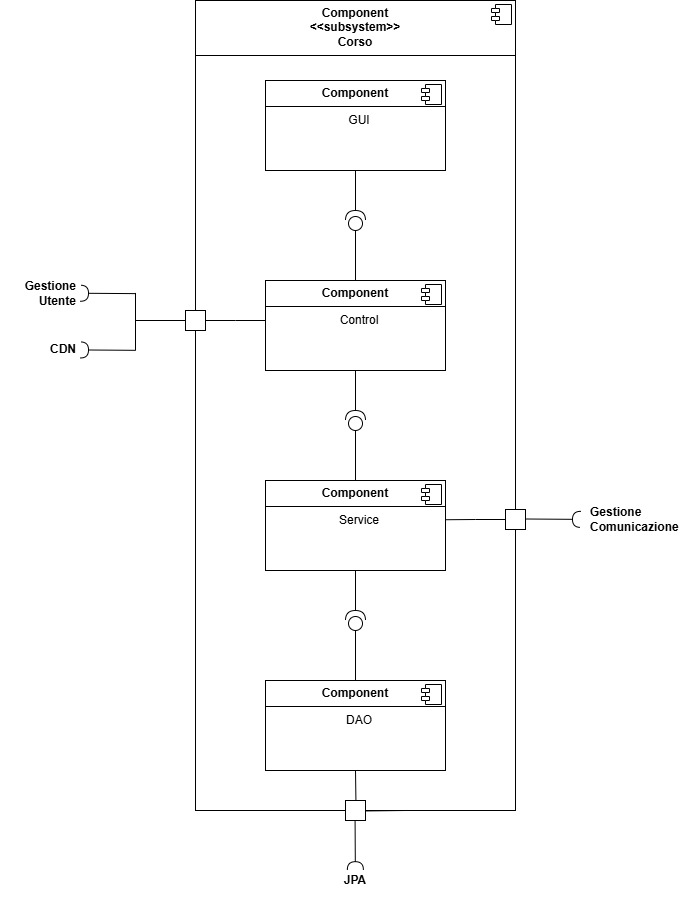


## 

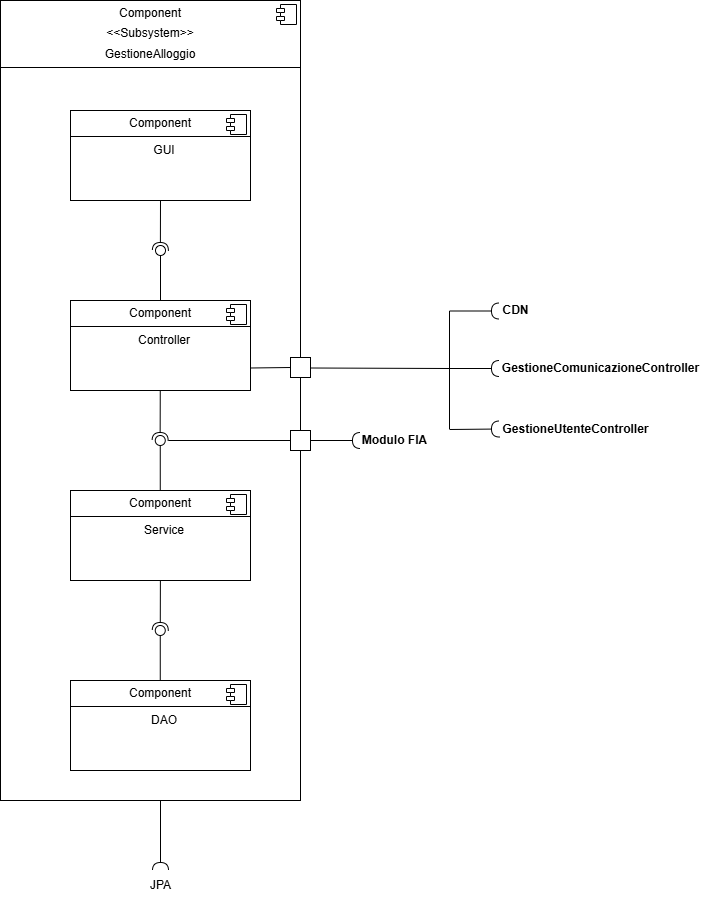
## 

## 

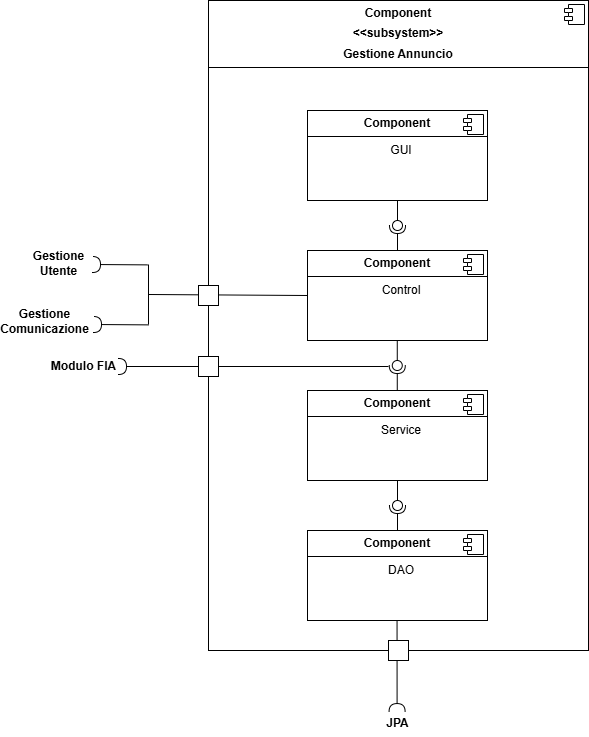
## 3.2.1- Gestione Utente



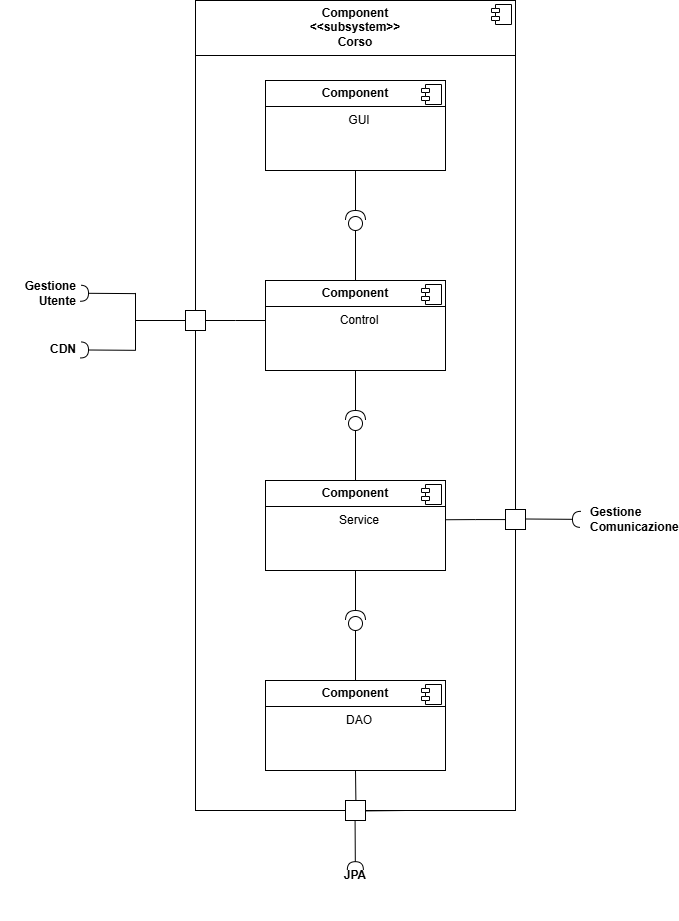
## 3.2.2- Gestione Alloggio



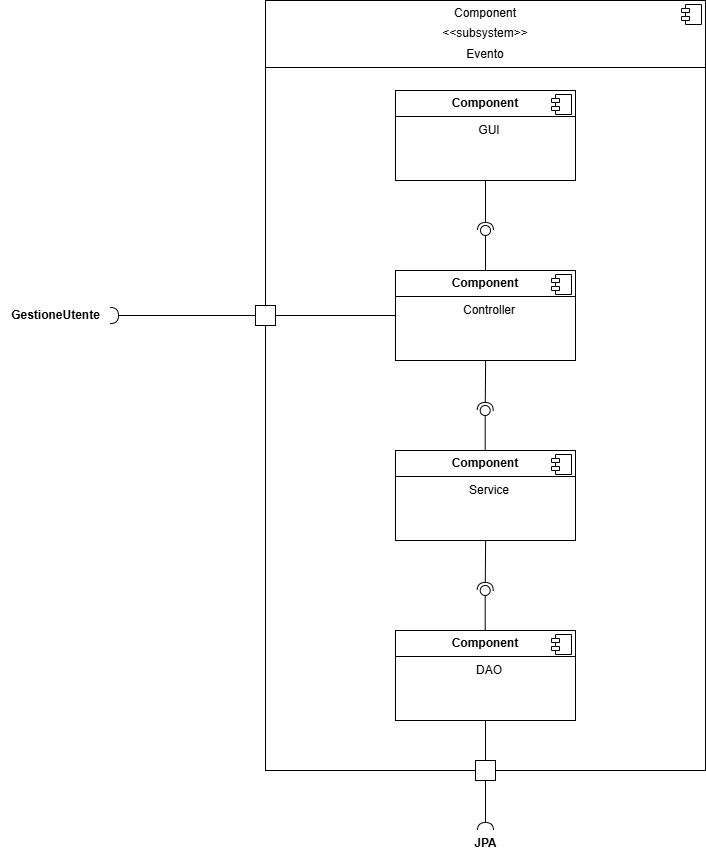
## 3.2.3- Gestione Annuncio



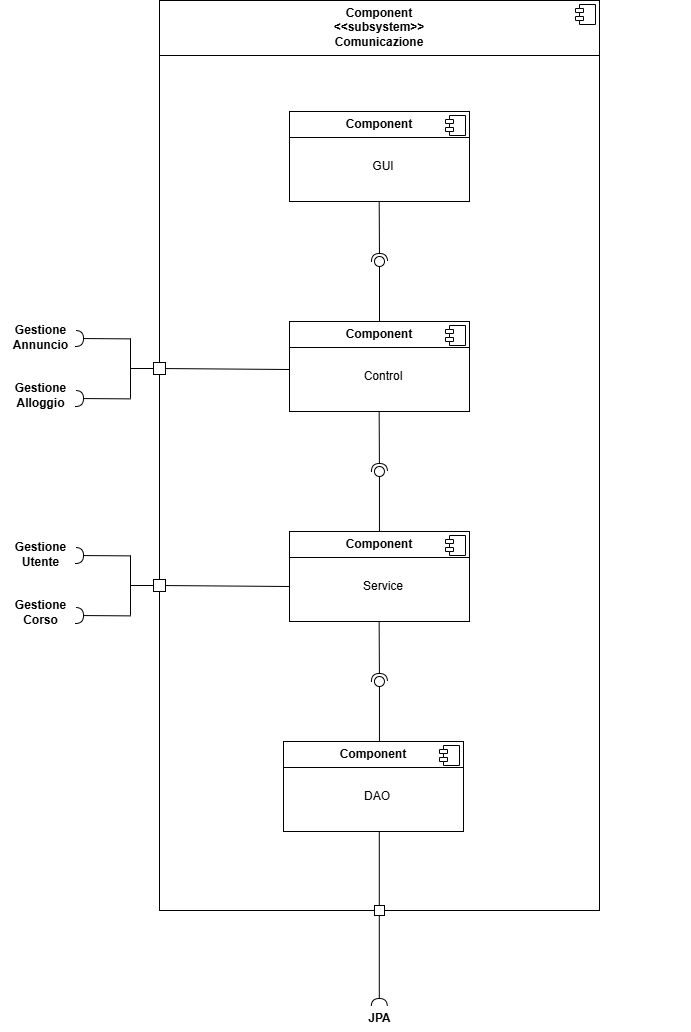
## 3.2.4- Gestione Corso



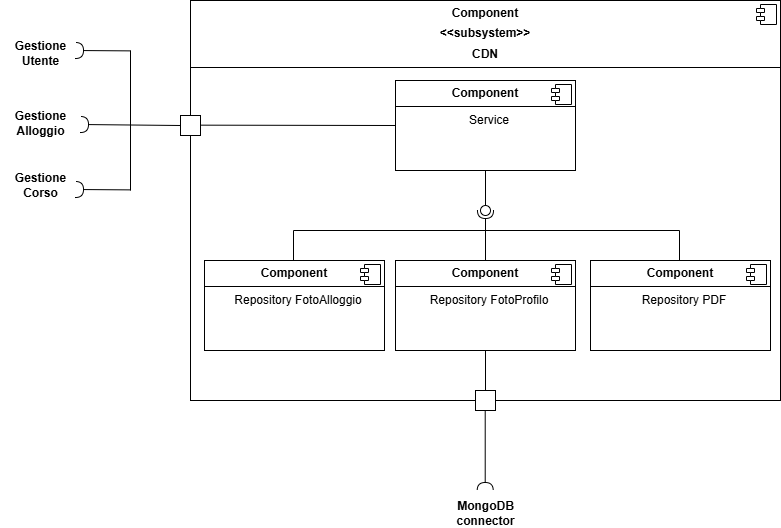
## 3.2.5- Gestione Evento



## 3.2.6- Gestione Comunicazione

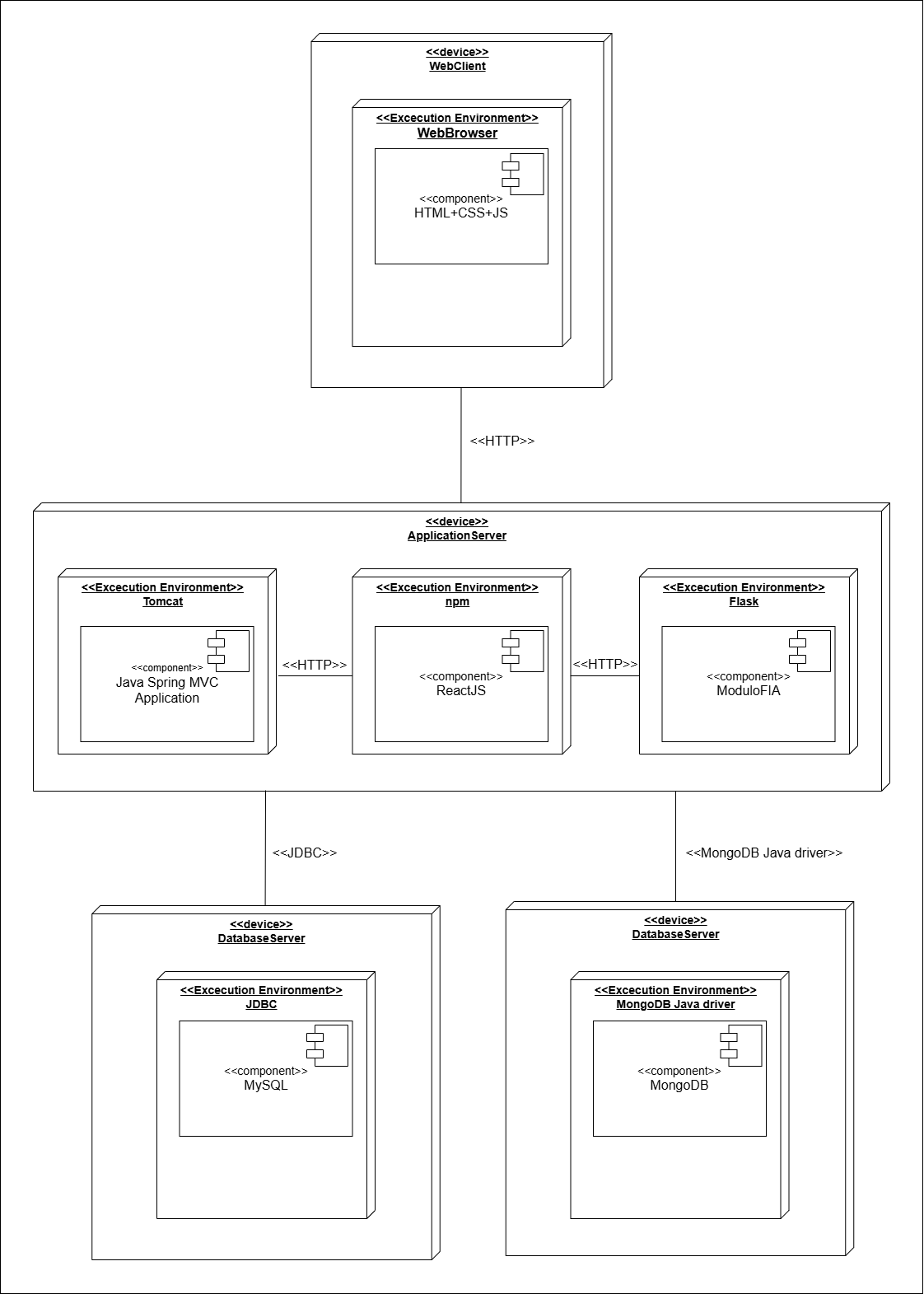


## 3.2.7- CDN



## 3.3 - Mapping Hardware/Software

**Deployment Diagram**

****

## 3.4 - Gestione dati persistenti

Per ottimizzare l'accesso ai dati e rispondere alle diverse esigenze di prestazioni e tipologia di contenuti, sono stati scelti due tipi di database: un **relazionale** (MySQL) per i dati strutturati e un **NoSQL** (MongoDB) per i contenuti non strutturati.

#### **Database Relazionale (MySQL)**

MySQL è utilizzato per gestire dati strutturati e con relazioni complesse, garantendo integrità referenziale per la gestione di dati strettamente correlati. Inoltre facilità di estrazione, manipolazione e aggregazione dei dati dato il supporto delle query SQL.

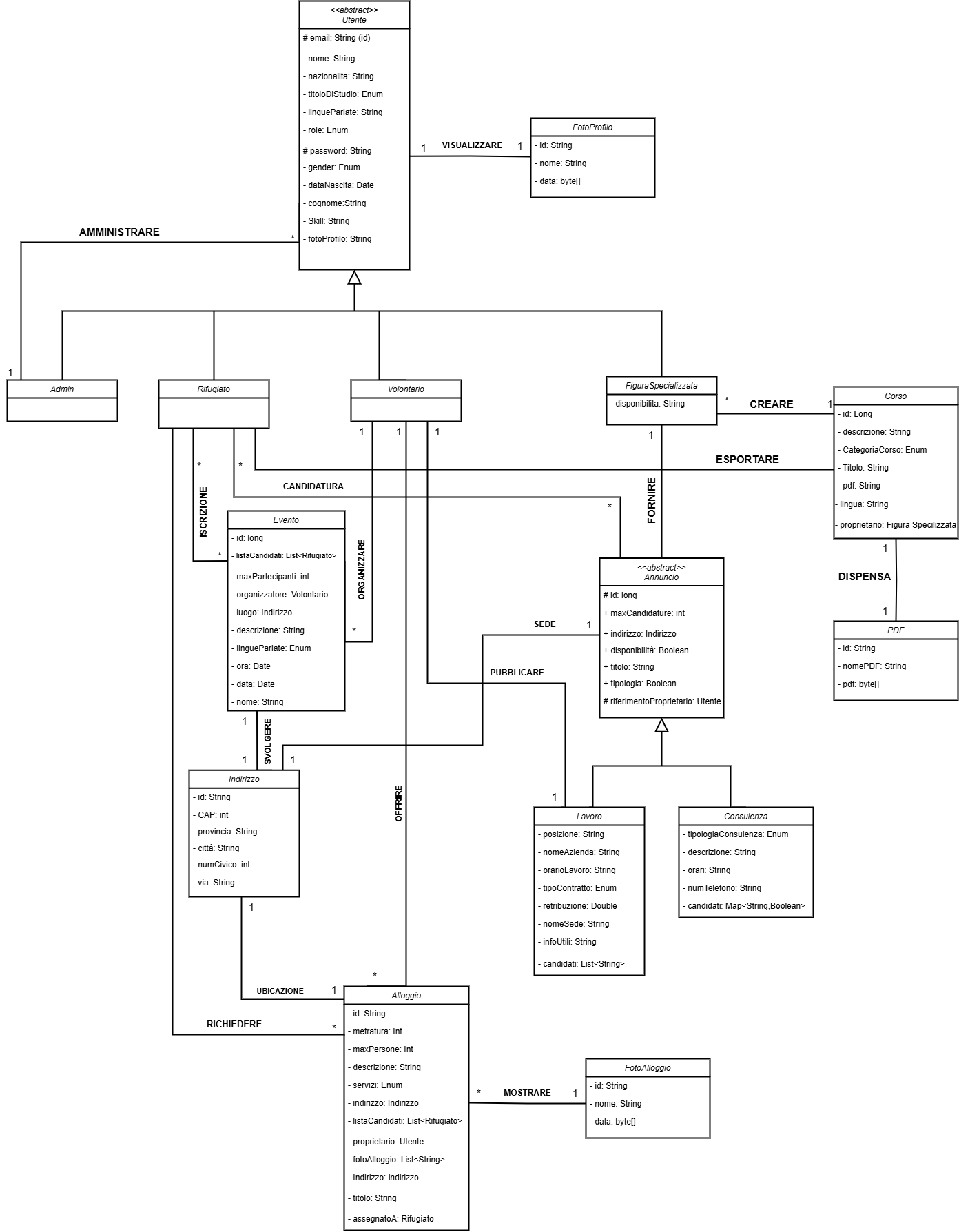
* **Database NoSQL (MongoDB)**

MongoDB è impiegato per gestire file e dati non strutturati (immagini, PDF, documenti), grazie alla sua scalabilità e flessibilità. È integrato con un **Content Delivery Network (CDN)**, che migliora le performance nella distribuzione di contenuti multimediali, riducendo la latenza e ottimizzando l'accesso globale. L'uso di un CDN è risultato particolarmente vantaggioso rispetto alla gestione diretta dei contenuti multimediali come Blob nei database, poiché quest'ultima opzione avrebbe comportato un significativo rallentamento delle prestazioni e una maggiore complessità di gestione.

L'uso combinato di MySQL e MongoDB consente di separare la gestione dei dati relazionali da quella dei file, migliorando le **performance**. Il CDN potenzia la **distribuzione dei file**, mentre MongoDB offre una gestione **scalabile** e flessibile dei contenuti multimediali.

Questo approccio permette di sfruttare al meglio le caratteristiche di ciascun database, ottimizzando l'archiviazione e l'accesso ai dati.

## 3.4.1 - Class Diagram ristrutturato

****

# 4 - Glossario

| **Termine / Sigla** | **Definizione** |
| --- | --- |
| **SQL injection** | Vulnerabilità di sicurezza in cui un attaccante manipola query SQL attraverso input non validati per accedere, modificare o eliminare dati nel database. |
| **Cross-Site Scripting** | Attacco il cui codice malevolo viene iniettato in pagine web visualizzate da altri utenti, sfruttando input non sanitizzati per eseguire script dannosi. |
| **principi SOLID** | Cinque principi fondamentali della programmazione orientata agli oggetti per migliorare la manutenibilità e la scalabilità del codice |
| **Documento SLA** | Contratto che definisce i livelli di servizio garantiti tra fornitori e clienti, specificando metriche misurabili come uptime, tempi di risposta e supporto. |
| **pipeline CI/CD** | Automazione dei processi di integrazione, testing e distribuzione del software, garantendo rapide iterazioni e riducendo errori nel rilascio. |
| **Java Spring Boot** | Framework basato su Java per lo sviluppo di applicazioni web e microservizi. Fornisce configurazioni predefinite per ridurre il codice boilerplate. |
| **GUI** | Interfaccia grafica utente che consente agli utenti di interagire con un'applicazione attraverso elementi visivi come pulsanti, finestre e menu. |
| **Control** | Componente della logica di presentazione in un'applicazione che gestisce la comunicazione tra GUI e servizi sottostanti. |
| **Service** | Componente della logica applicativa che contiene le operazioni principali del dominio, separando la logica dal livello di persistenza. |
| **DAO** | Pattern di progettazione che astrae le operazioni di accesso al database, separando il livello di persistenza dal resto dell'applicazione. |
| **JPA** | Specifiche Java per mappare oggetti a database relazionali, consentendo operazioni di persistenza senza scrivere query SQL dirette. |
| **MongoDB connector** | Driver o libreria utilizzata per connettere un'applicazione a un database MongoDB, facilitando operazioni CRUD su dati NoSQL. |