System Design Document

MinervHub

Riferimento	NC07_SDD
Versione	2.1
Data	02/01/2026
Destinatario	Docente di Ingegneria del Software 2025/26
Presentato da	NC07 - Gennaro Pio Albano (G.P.A.), Giuseppe Annunziata (G.A.), Alessandro Bonelli (A.B.)
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
03/11/2025	1.0	Prima stesura	Tutto il team
07/11/2025	1.1	Design goals	Tutto il team
10/11/2025	1.2	Scomposizione in sottosistemi	Tutto il team
12/11/2025	1.3	Mapping software hardware	Tutto il team
13/11/2025	1.4	Ristrutturazione Class Diagram	A.B.
13/11/2025	1.4.1	Schema E-R	G. P. A.
13/11/2025	1.4.2	Dizionario delle Entità	G.A
14/11/2025	1.5	Access control and security	A.B.
14/11/2025	1.5.1	Global Software Control	G.P.A.
14/11/2025	1.5.2	Boundary Condition	Tutto il team
17/11/2025	1.6	Revisione e modifiche generali	Tutto il team

24/11/2025	1.7	Modifica del System decomposition	G.A.
28/11/2025	1.8	Modifica sottosistemi bacheca e gestione annunci	G.A.
29/11/2025	1.9	Modifica schema E-R	A.B.
17/12/2025	1.9.1	Modifiche architettura	Tutto il team
22/12/2025	2.0	Revisione finale ed ultime modifiche	Tutto il team
02/01/2026	2.1	Modifica trade-off	A.B.

Indice

System Design Document (SDD) del Progetto MinervHub

1. Introduction

1.1 Purpose of the system

Il sistema fornisce una piattaforma web che facilita l'incontro tra studenti dell'Università degli Studi di Salerno interessati a offrire o ricevere lezioni private, promuovendo la collaborazione, la condivisione delle conoscenze e il supporto reciproco nello studio.
La piattaforma consente agli utenti di creare un profilo personale, pubblicare **annunci** per offrire lezioni su specifici corsi o esami e ricercare **tutor** in base a criteri come corso di laurea, esame e tariffa oraria.

Il sistema fornisce funzionalità avanzate che semplificano la gestione delle interazioni tra studenti, tra cui l'invio e la gestione delle **richieste di contatto** e la possibilità di effettuare **scambi di lezioni** senza compenso economico.
L'architettura web-based garantisce accessibilità da qualsiasi dispositivo, mentre un'interfaccia intuitiva e responsive assicura un'esperienza d'uso fluida e coerente.

Il sistema non si limita a semplificare la ricerca di tutor, ma mira a creare una **community accademica collaborativa**, migliorando l'efficienza, la trasparenza e la qualità delle opportunità di tutoraggio tra pari all'interno dell'Ateneo.

1.2 Design Goals

In questa sezione verranno presentati i Design Goals, ovvero le qualità del sistema che dovranno essere migliorate e ottimizzate. I Design Goals scelti sono divisi nelle seguenti categorie:

- Dependability
- Performance
- Maintenance

- End user

Ogni design goal è descritto da:

- **Rank**: Un livello di priorità compreso da 1 a 9
- **ID Design Goal**: Identificativo univoco
- **Descrizione**: Descrizione del design goal
- **Categoria**: Categoria di appartenenza del design goal
- **RNF di origine**: Requisito non funzionale da cui deriva

Rank	ID	Nome	Descrizione	Categoria	RNF di origine
1	DG_01	Alta disponibilità	Il sistema deve essere operativo 24 ore su 24, 7 giorni su 7, garantendo un'elevata disponibilità del servizio.	Dependability (Availability)	RNF_01
2	DG_02	Interfaccia intuitiva	L'interfaccia utente deve essere semplice, coerente e accessibile, rispettando le 8 regole d'oro di Shneiderman per facilitare la navigazione anche agli utenti meno esperti.	End user criteria (Usability)	RNF_02

9	DG_03	Documentazione degli artefatti	Il codice e l'infrastruttura devono essere adeguatamente documentati e tracciati in modo da agevolare manutenzione, aggiornamenti e interventi futuri da parte di diversi tecnici.	Maintenance	RNF_09
5	DG_04	Protezione dei dati	Il sistema deve garantire la sicurezza e la protezione dei dati degli utenti attraverso meccanismi adeguati di sicurezza, tra cui l'hashing delle password.	Dependability (Security)	RNF_03
4	DG_05	Prestazioni Ottimizzate	Le operazioni principali, come la ricerca e il filtraggio degli annunci, devono essere completate restituendo i risultati entro un tempo massimo di 3 secondi.	Performance (Response time)	RNF_06
3	DG_06	Interfaccia responsive e adattiva	L'interfaccia deve adattarsi automaticamente alle diverse dimensioni dello schermo (desktop, tablet, smartphone), garantendo una	End user criteria (Usability)	RNF_04

			visualizzazione ottimale su ogni dispositivo.		
8	DG_07	Facilità di Manutenzione	Il sistema deve essere sviluppato seguendo le linee guida definite dagli standard IEEE e ISO/IEC al fine di facilitare le attività di manutenzione.	Maintenance	RNF_08
6	DG_08	Gestione dei malfunzionamenti	In caso di errore o malfunzionamento, il sistema deve essere in grado di rilevare la condizione anomala e notificare correttamente all'utente mediante messaggi appropriati.	Dependability (Fault tolerance)	RNF_05
7	DG_09	Web Based	Il sistema deve essere implementato come applicazione web client-server, accessibile tramite browser moderni (es. Chrome, Firefox, Safari), senza necessità di installare software aggiuntivo lato client.	End user criteria	RNF_07

1.2.1 Trade-offs

Trade-Off	Descrizione
Performance vs End User	Un'interfaccia ricca, guidata e molto informativa aumenta la soddisfazione dell'utente, ma aggiunge attività da eseguire per ogni operazione (caricamento di elementi, controlli, visualizzazioni), entrando in tensione con l'obiettivo di restituire risultati di ricerca entro 3 secondi.

End User vs Dependability	Il sistema adotta meccanismi di sicurezza adeguati per la protezione dei dati degli utenti, come l'hashing delle password, mantenendo al contempo un livello di sicurezza bilanciato. La scelta progettuale privilegia la semplicità e la fluidità dell'esperienza utente, evitando soluzioni di sicurezza eccessivamente invasive che potrebbero introdurre passaggi aggiuntivi o rallentare l'interazione con le funzionalità offerte dal sistema.

1.3 Definition, acronyms, e abbreviations

Nel corso del documento verranno utilizzati i seguenti termini e abbreviazioni:.

- **DG:** Design Goal.
- **UCBC**: Use Case Boundary Condition.
- **RAD:** Requirements Analysis Document.
- **SOW**: Statement of Work.
- **JPA**: Java Persistence API.

1.4 References

Libri e documentazioni:

- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition. Autori: Bernd Bruegge & Allen H. Dutoit.
- SOW.
- Slide del corso di Ingegneria del Software.
- NC07_RAD.

1.5 Overview

Il presente documento di System Design è organizzato in tre sezioni principali:

1. **Introduzione**: Una panoramica generale che descrive lo scopo del sistema e gli obiettivi di design che si intende perseguire.
2. **Architettura software corrente**: La descrizione dello stato attuale dell'architettura di qualsiasi software esistente.
3. **Architettura software proposta**: Una descrizione dettagliata della suddivisione del sistema in sottosistemi, del mapping hardware/software e della gestione dei dati persistenti.

2. Current architecture

Al momento, non esiste alcun sistema specifico per l'Università degli Studi di Salerno come MinervHub. Il mercato delle possibili alternative a questo software è pertanto incredibilmente dispersivo e poco mirato, non esiste una reale architettura a cui è possibile confrontare in maniera ragionevole il sistema.

3. Proposed architecture

3.1 Overview

Il sistema adotta un'architettura software a tre layer: Presentation, Business e Data Access. Il Presentation layer è implementato secondo il pattern Spring MVC e comprende controller e viste server-side realizzate con Thymeleaf. L'applicazione è sviluppata con il framework Spring Boot, che gestisce la logica applicativa e la comunicazione con il database ospitato su Microsoft Azure, mentre verranno usati HTML5, CSS3 e Thymleaf per la parte di front-end e la generazione delle view.

Grazie a questa architettura, il sistema risulta strutturato, estensibile e facilmente mantenibile, con una netta separazione tra la logica di presentazione, di elaborazione e di persistenza, e una piena integrazione con l'infrastruttura cloud di Azure.

3.2 System decomposition

I sottosistemi individuati sono:

- **Utente**: Si occupa delle funzionalità di registrazione, login, logout, visualizzazione area utente e la modifica dati account.
- **Richiesta Contatto**: Si occupa di inviare richieste di contatto, visualizzare ed eliminare le richieste di contatto inviate e visualizzare, accettare o declinare le richieste di contatto ricevute.
- **Bacheca**: Permette di visualizzare gli annunci pubblicati sulla piattaforma, cercare e filtrare gli annunci.
- **Annuncio**: Si occupa della creazione, eliminazione, modifica e visualizzazione dei propri annunci.
- **Storage JPA**: Si interpone tra i vari sottosistemi e il sottosistema di Database.
- **Database**: Si occupa di gestire le operazioni di persistenza dei dati, con il Database associato.

Nel modello dei casi d'uso la funzionalità di visualizzazione bacheca (descritta dallo use case UC_02) è definita come parte del flusso dell'invio della richiesta di contatto. In fase di progettazione architetturale, tale funzionalità è stata separata in un sottosistema dedicato, al fine di migliorare la modularità e la chiarezza del sistema.

Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un Component Diagram.

3.2.1 Component Diagram

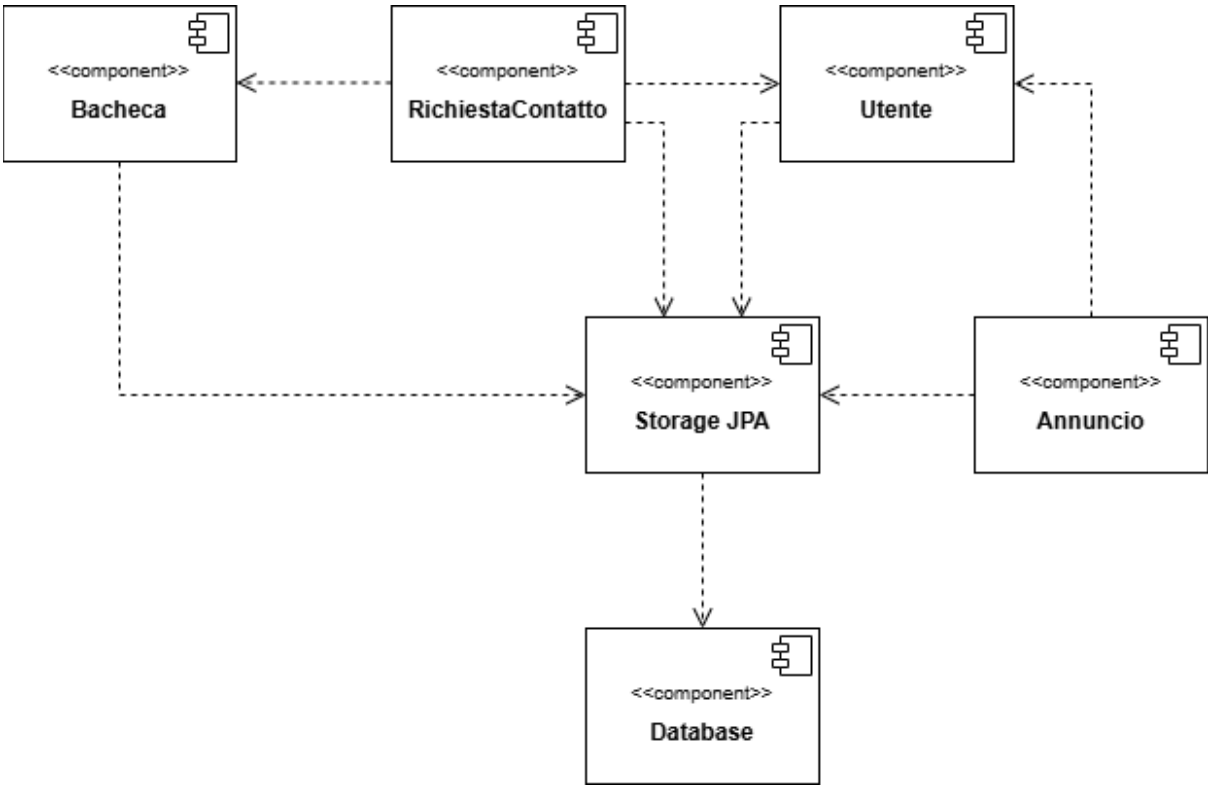
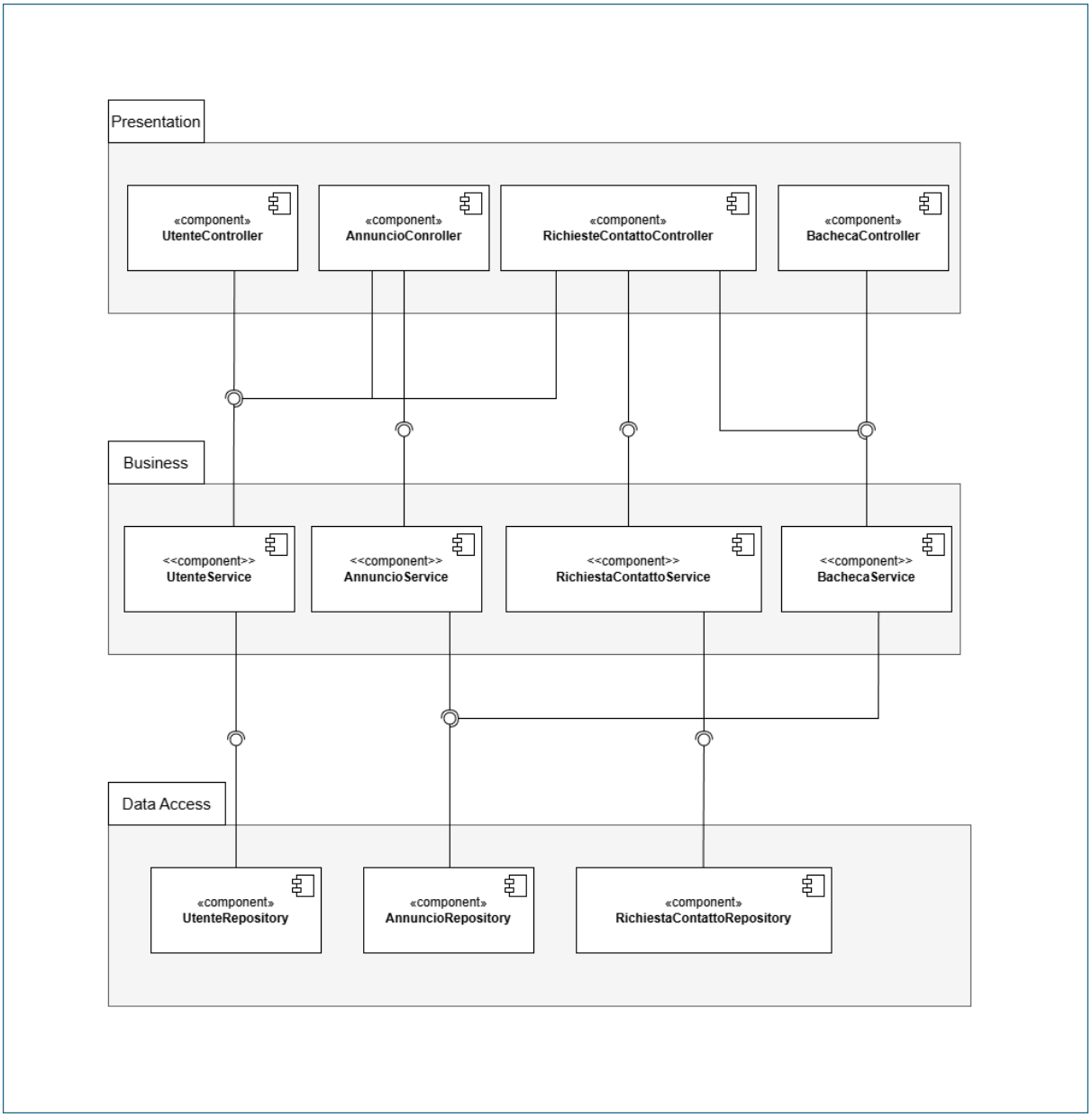


Diagramma architetturale



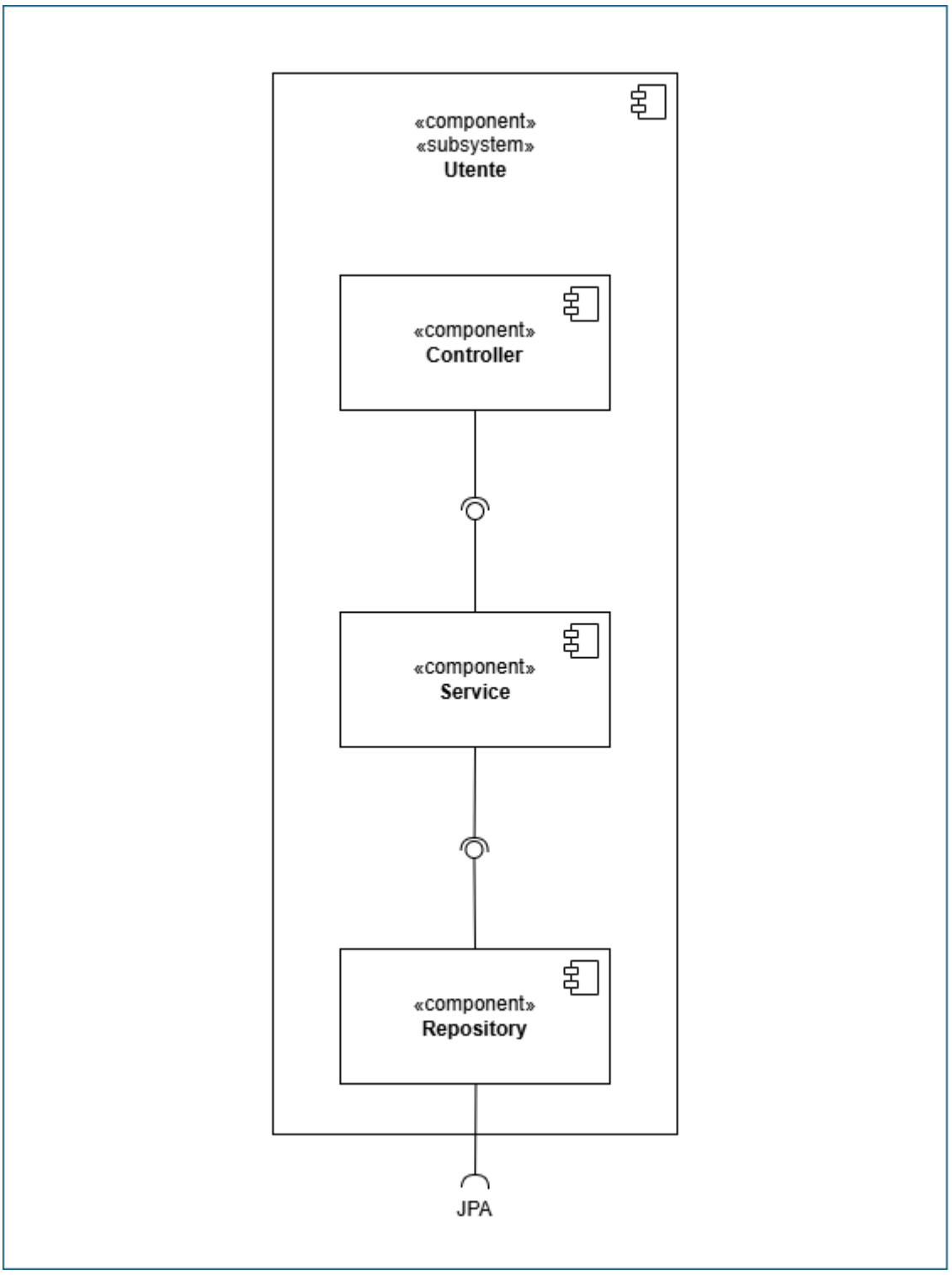
Sebbene, dal punto di vista del riuso del codice, sarebbe stato possibile accorpare le funzionalità di *BachecaService* all'interno di *AnnuncioService*, si è scelto di mantenere i due servizi distinti al fine di garantire una maggiore coesione, una

migliore separazione delle responsabilità e una maggiore chiarezza dell'architettura complessiva.

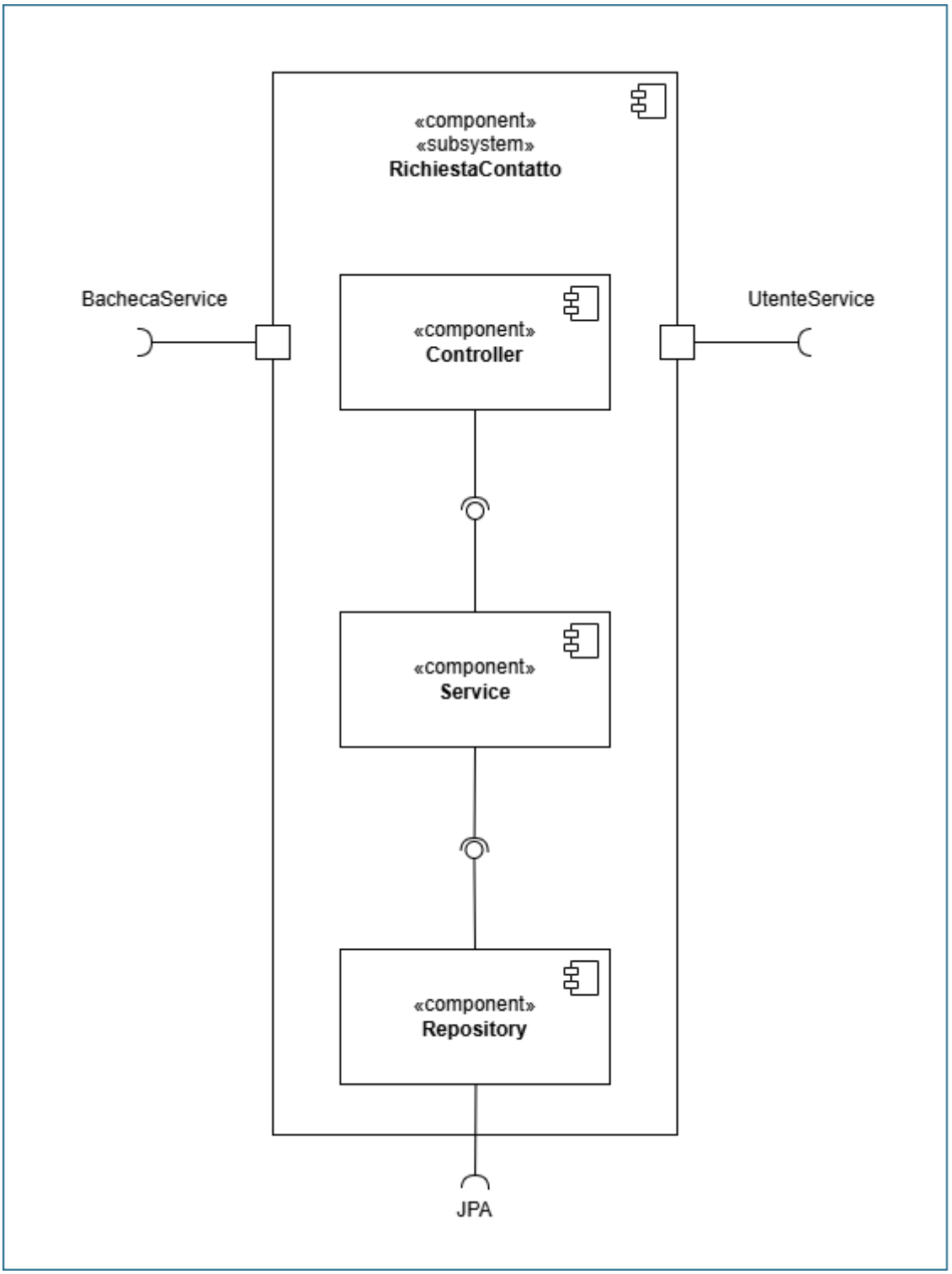
Di seguito una vista dettagliata di ciascun sottosistema evidenziando le componenti principale:

- **Controller**: si occupa della logica per il controllo del sistema.
- **Service**: si occupa della logica di business.
- **Repository**: si occupa di fornire accesso ai dati persistenti.

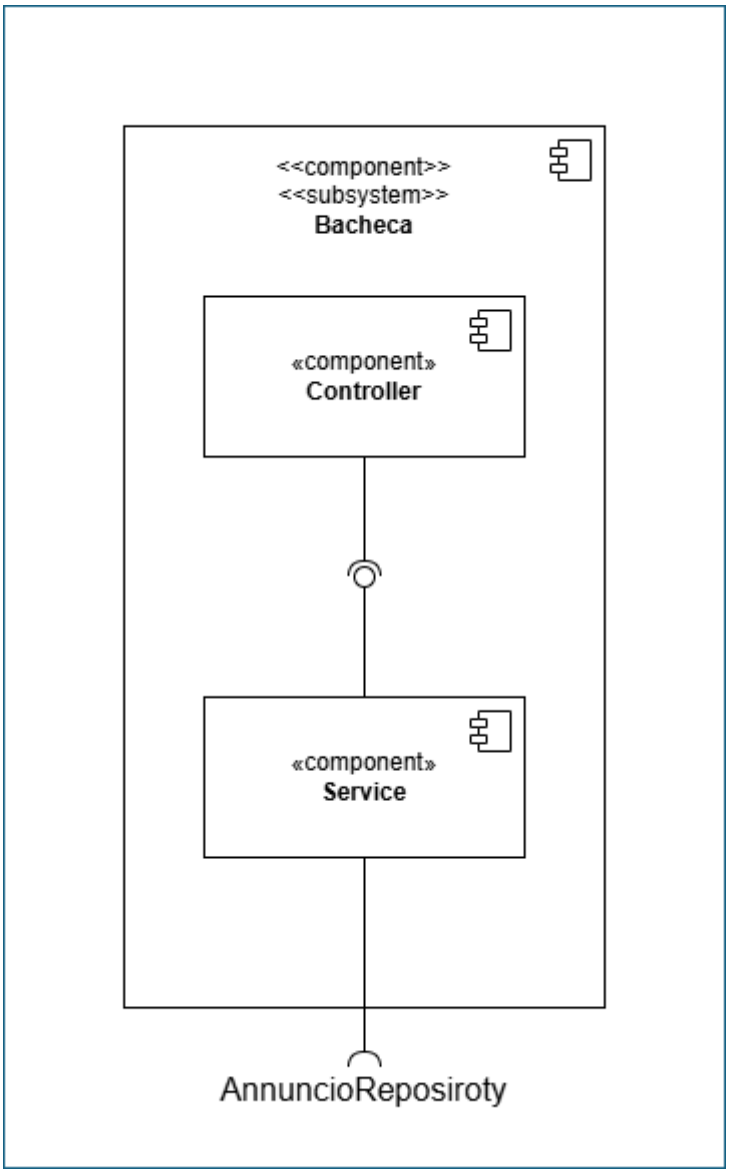
3.2.2 Sottosistema Utente



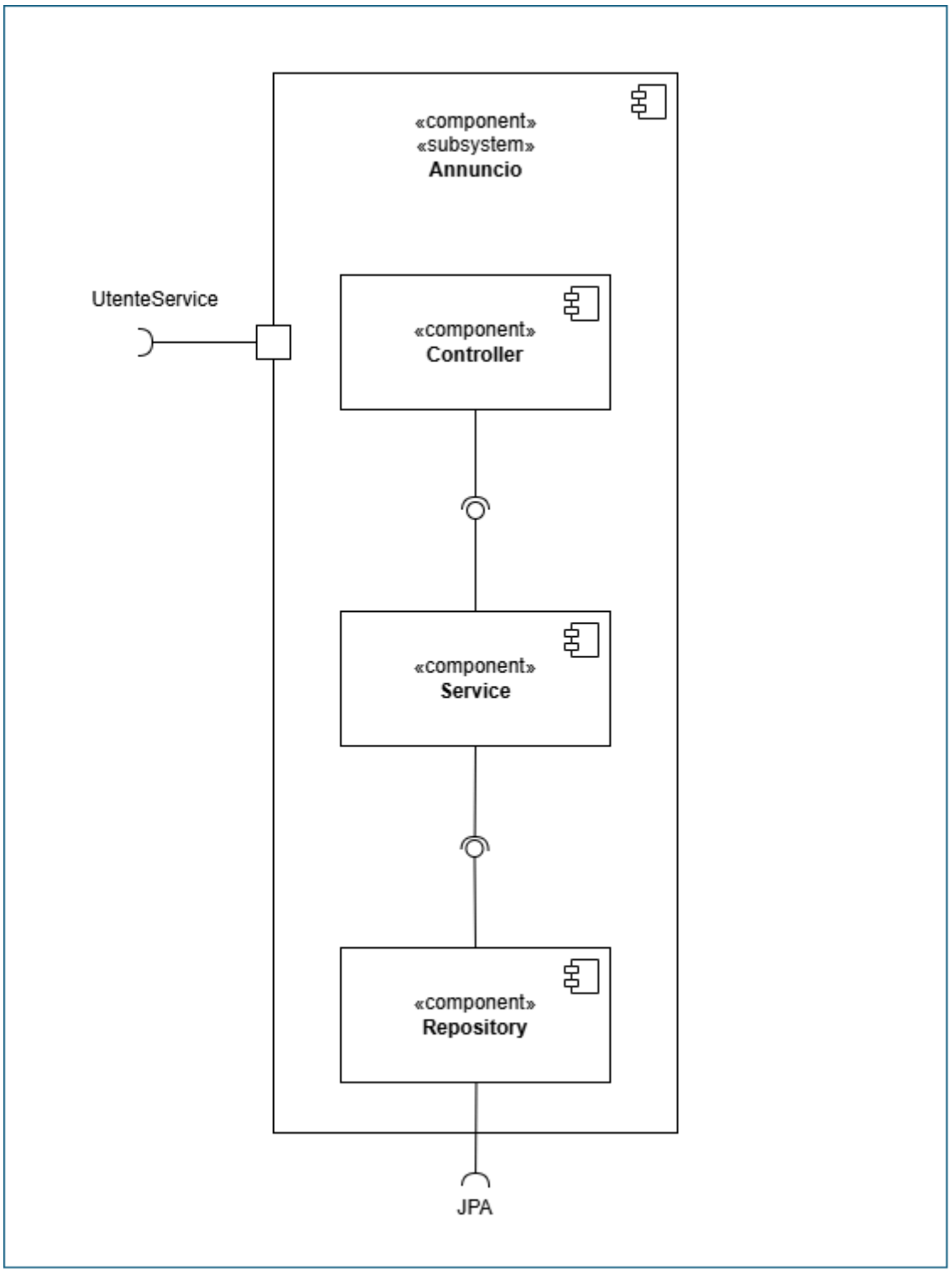
3.2.3 Sottosistema Richiesta Contatto



3.2.4 Sottosistema Bacheca

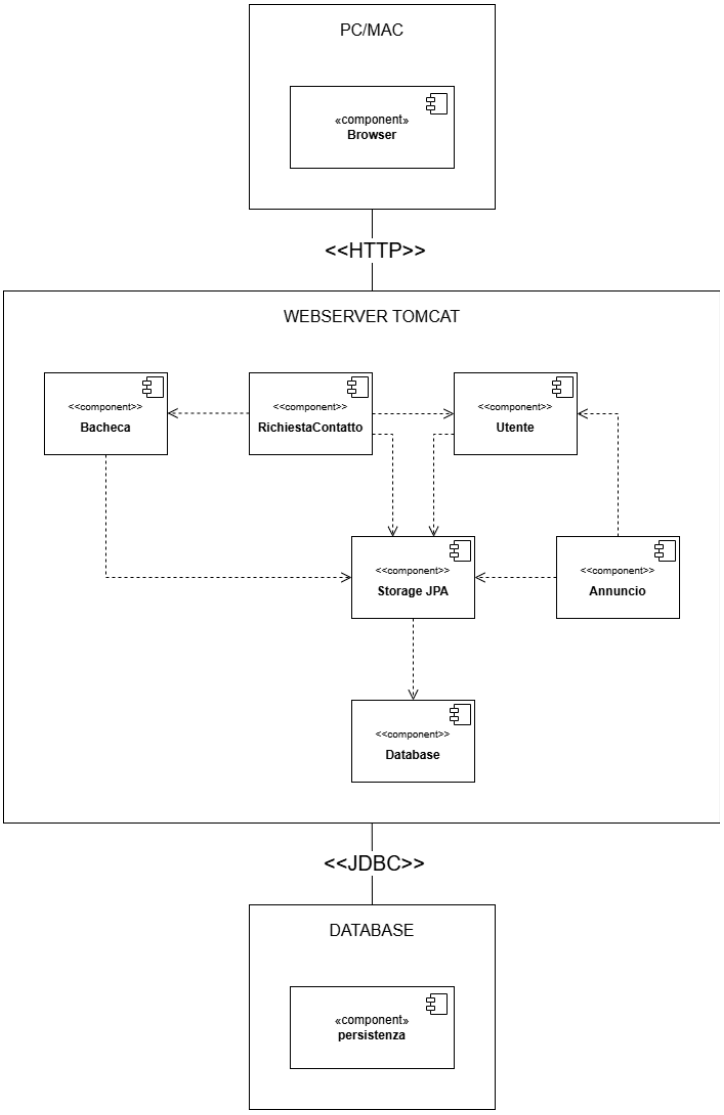


3.2.5 Sottosistema Annuncio



3.3 Mapping Hardware/Software

L'applicazione web sviluppata si basa su un'infrastruttura hardware composta da un server centrale, che gestisce e risponde alle richieste provenienti dai client connessi tramite un browser web e una connessione Internet.
L'applicazione è eseguita su un unico nodo logico (server web Spring), mentre il database è gestito come servizio cloud esterno tramite Azure Database for MySQL.
Di seguito è riportato un diagramma UML di deployment che illustra la corrispondenza tra componenti software e risorse hardware.



3.4 Persistent Data Management

Nel progetto è stato scelto l'utilizzo di un DBMS relazionale, in particolare MySQL, ospitato su Microsoft Azure.
Questa decisione è motivata dalla natura dei dati gestiti dal sistema e dalla necessità di garantire integrità, coerenza e affidabilità delle informazioni.

L'impiego di un DBMS relazionale offre inoltre i seguenti vantaggi:

- **Struttura rigorosa dei dati:** l'organizzazione in tabelle con schema predefinito garantisce chiarezza nella modellazione e semplifica la manutenzione del database.

- **Integrità e vincoli:** l'uso di vincoli assicura che i dati rispettino regole di validità e consistenza.

- **Linguaggio standardizzato (SQL):** l'adozione di SQL permette di eseguire query complesse e di mantenere la compatibilità con diversi ambienti e strumenti di sviluppo.

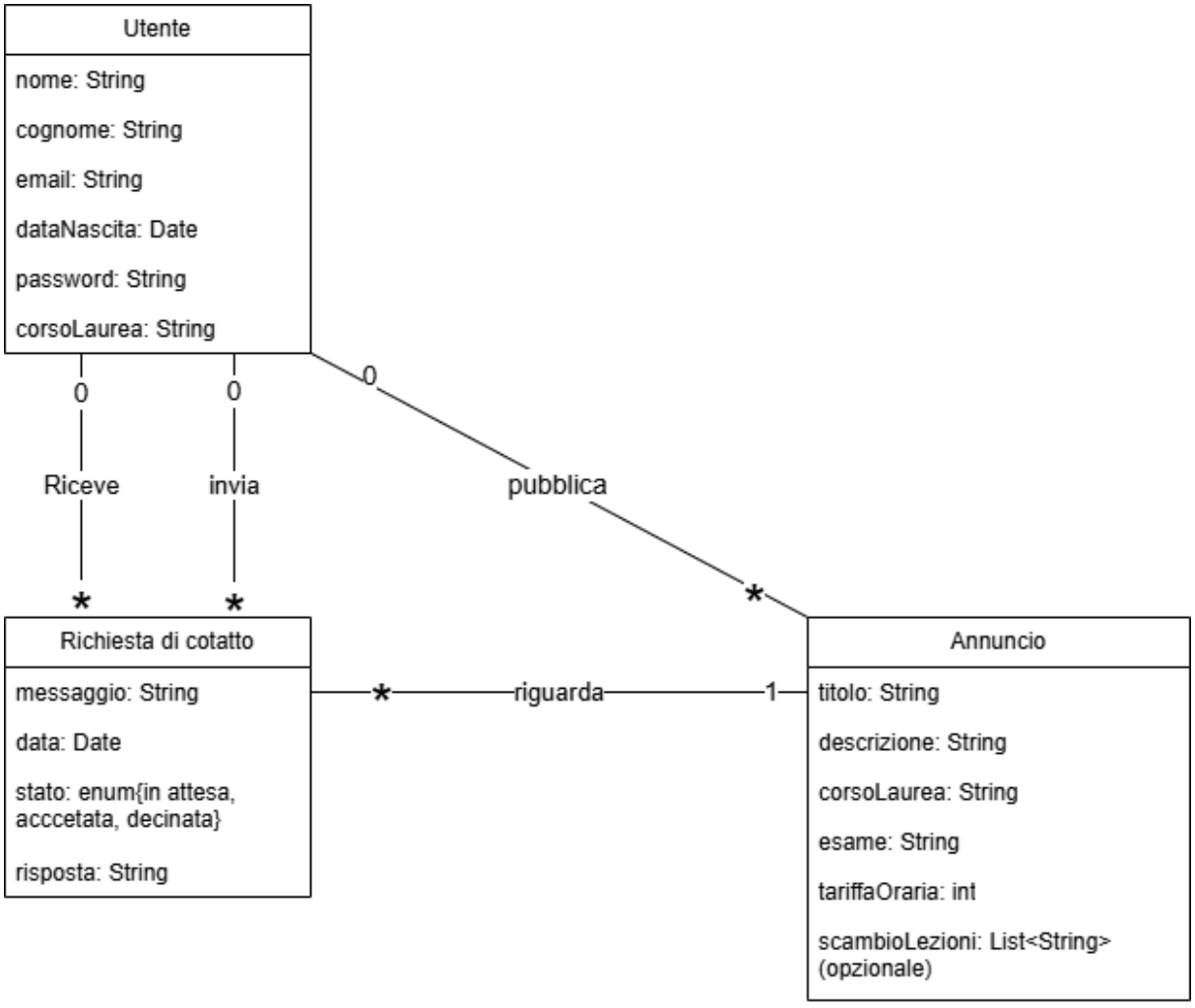
- **Supporto nativo di Spring:** il framework Spring integra nativamente il supporto ai database relazionali tramite Spring Data.

- **Affidabilità e scalabilità garantite da Azure:** l'utilizzo di Azure Database for MySQL assicura elevata disponibilità, backup automatici e possibilità di scalare le risorse in base alle esigenze future del sistema.

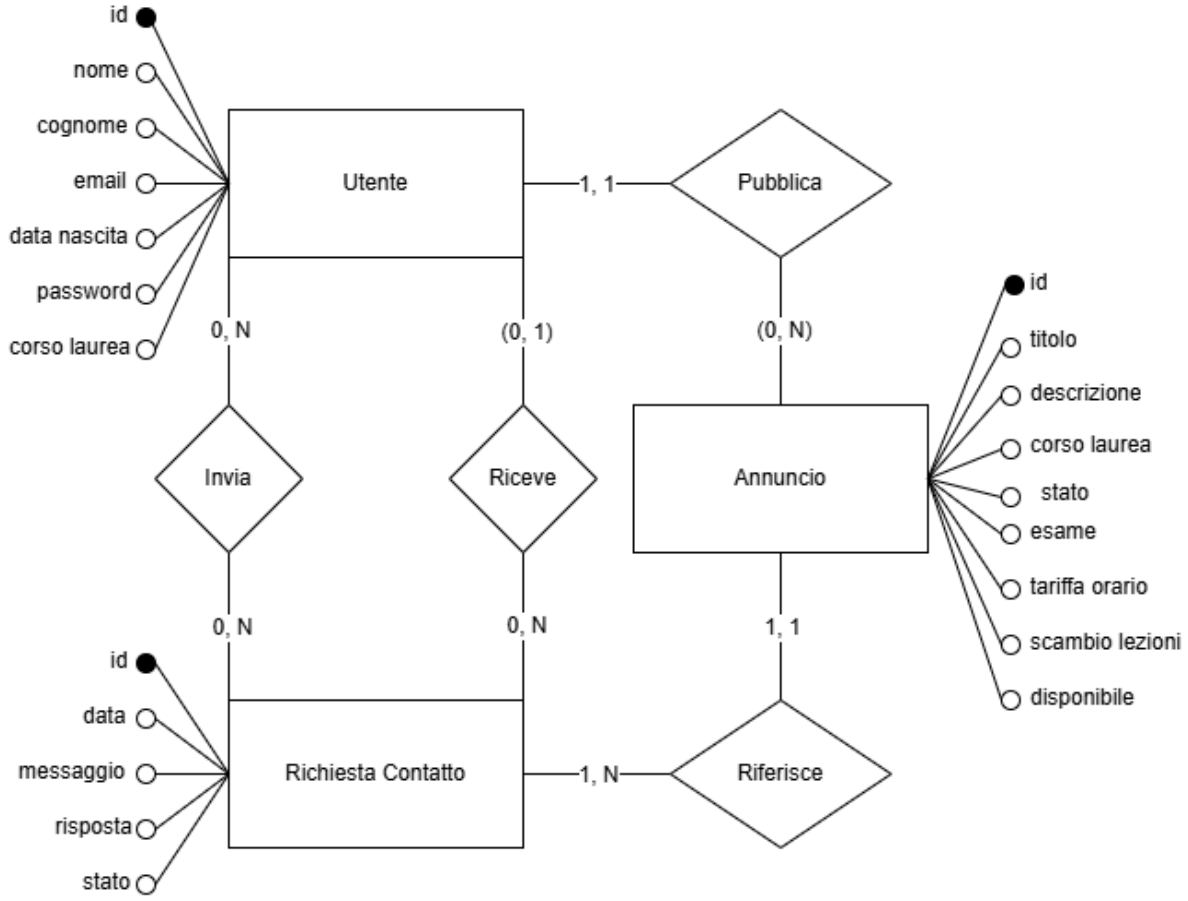
Nel complesso, un DBMS relazionale rappresenta la soluzione più adeguata per un sistema come questo, in cui le informazioni devono essere coerenti, correlate e persistentemente condivise tra più utenti.

3.4.1 Class Diagram Ristrutturato

Di seguito il **Class Diagram** completo dei tipi di ogni attributo:



3.4.2 Schema E-R



Dizionario dei dati

Nome Entità	**Utente**		
Descrizione	Contiene i dati relativi ad un utente		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
id	int	PRIMARY KEY	

nome	Varchar(50)		NOT NULL
cognome	Varchar(50)		NOT NULL
email	Varchar(100)		NOT NULL
password	Varchar(50)		NOT NULL
dataNascita	Date		NOT NULL
corsoLaurea	Varchar(50)		NOT NULL

Nome Entità	**Annuncio**		
Descrizione	Contiene i dati relativi ad un annuncio pubblicato sulla piattaforma		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
id	int	PRIMARY KEY	AUTO INCREMENT
utente	Varchar(100)	FOREIGN KEY(Utente)	NOT NULL
titolo	Varchar(50)		NOT NULL

stato	Boolean		NOT NULL
descrizione	Varchar(150)		NOT NULL
esame	Varchar(50)		NOT NULL
tariffaOraria	int		NOT NULL 5 <= tariffaOraria <=50
scambioLezioni	Varchar(150)		*Lista opzionale di stringhe separate da virgola, rappresentante gli esami disponibili per lo scambio.*
corsoLaurea	Varchar(50)		corsoLaurea=utente_corsoLaurea
disponibile	Boolean		NOT NULL

Nome Entità	**Richiesta di Contatto**
Descrizione	Contiene i dati relativi ad una richiesta di contatto

Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id	int	PRIMARY KEY	
messaggio	Varchar(200)		default NULL
risposta	Varchar(200)		default NULL
data	DateTime		NOT NULL
stato	enum		Può assumere • 0: in attesa • 1: accettata • 2: declinata NOT NULL
allievo	int	FOREIGN KEY (Utente)	NOT NULL && allievo≠tutor
tutor	int	FOREIGN KEY (Utente)	NOT NULL && allievo≠tutor
id_annuncio	int	FOREIGN KEY (Annuncio)	NOT NULL

3.5 Access control and security

Nel sistema è presente un'unica tipologia di utente, che può accedere alle varie funzionalità della piattaforma dopo il processo di autenticazione.

Tutti gli utenti autenticati dispongono dello stesso livello di permessi, mentre gli utenti non registrati possono esclusivamente registrarsi e visualizzare gli annunci pubblici senza possibilità di interazione.

Il processo di autenticazione dell'utente è implementato tramite credenziali di accesso (email e password), garantendo l'accesso ai dati personali solo al proprietario dell'account.

3.6 Global Software Control

Nel sistema *MinervHub*, ogni operazione viene attivata a seguito di un'azione compiuta dall'utente tramite l'interfaccia web.

Quando l'utente seleziona una funzionalità l'interfaccia richiama il corrispondente componente di controllo, che prende in carico la richiesta e la inoltra al sottosistema dedicato alla gestione della logica applicativa.

Questo sottosistema coordina il flusso delle operazioni e utilizza i servizi interni per eseguire le elaborazioni necessarie, restituendo poi il risultato all'interfaccia utente.

Quindi essendo il sistema una web application, il flusso operativo è di tipo **event-driven**.

3.7 Boundary Condition

Nel presente paragrafo verranno presentate le boundary conditions inerenti all'avvio del sistema, spegnimento del sistema e fallimento del sistema.

3.7.1 Inizializzazione

Identificativo	UCBC_01 – Inizializzazione del sistema	**Data**	14/11/2025
		Versione	1.0
		Autori	G.P.A.
Descrizione	Lo UC rappresenta il comportamento del sistema al momento dell'attivazione, garantendo che tutte le componenti necessarie siano caricate e che la piattaforma diventi operativa e accessibile tramite browser web dagli utenti finali.		
Attore principale	Amministratore di sistema (attore implicito - non utente finale)		
Attori secondari	NA		
Entry condition	• L'ambiente in cui il sistema opera è disponibile e funzionante. • Le risorse necessarie all'avvio (es. servizio interno, archivio dati, configurazioni) sono accessibili		
Exit condition **On success**	• Tutte le componenti software sono state caricate correttamente. • Le risorse necessarie sono state verificate e risultano disponibili. • Il sistema è pienamente operativo e pronto a ricevere richieste da parte degli utenti tramite interfaccia web.		
Exit condition **On failure**	• L'inizializzazione non viene completata. • Il sistema rimane non operativo. • Viene generata una segnalazione o un messaggio di errore rivolto all'Amministratore di sistema.		
Flusso di eventi principale			
1	Amministratore di sistema	Avvia il processo di inizializzazione del sistema.	
2	Sistema	Carica le componenti principali necessarie al funzionamento.	
3	Sistema	Verifica la disponibilità delle risorse fondamentali (es. dati, configurazioni, servizi interni).	
4	Sistema	Conclude l'inizializzazione e si pone in stato operativo, pronto a ricevere richieste provenienti dagli utenti tramite interfaccia web.	

I Flusso di Eventi Alternativo: Risorse necessarie non disponibili		
A1	Sistema	Durante la verifica delle risorse fondamentali, rileva che una o più risorse necessarie non sono disponibili o non raggiungibili.
A2	Sistema	Interrompe il processo di inizializzazione.
A3	Amministratore di sistema	Notifica all'Amministratore di sistema l'indisponibilità delle risorse, fornendo un messaggio di errore appropriato.
A4	Amministratore di sistema	Ripristina la disponibilità delle risorse non raggiungibili.
A5	Amministratore di sistema	Ripete il tentativo di inizializzazione eseguendo nuovamente il passo 1 del flusso principale.
II Flusso di Eventi Alternativo: Dati persistenti non disponibili o non validi		
B1	Sistema	Rileva un problema con i dati persistenti necessari all'avvio e interrompe il processo di inizializzazione.
B2	Sistema	Notifica l'Amministratore di sistema dell'anomalia riscontrata, fornendo un messaggio di errore appropriato.
B3	Amministratore di sistema	Corregge i dati persistenti non disponibili o non validi.
B4	Amministratore di sistema	Ripete il tentativo di inizializzazione eseguendo nuovamente il passo 1 del flusso principale.

3.7.2 Fallimento

Identificativo	UCBC_02 – Fallimento del Sistema	**Data**	14/11/2025
		Versione	1.0
		Autori	G.A.
Descrizione	L'UC definisce il comportamento del Sistema in caso di malfunzionamento, informando l'utente senza esporre dettagli tecnici		
Attore principale	Utente (allievo/tutor)		
Attori secondari	NA		
Entry condition	Errore anomalo che interrompe il corretto flusso degli eventi		
Exit condition On success	Il sistema mostra all'utente un messaggio di errore		
Exit condition On failure	Il Sistema non risponde		

Flusso di eventi principale		
1	Utente	Esegue un'operazione (es ricerca annunci)
2	Sistema	Si verifica un errore imprevisto (es problema di connessione al database)
3	Sistema	Mostra all'utente un errore generico avvisando che l'operazione non è andata a buon fine

3.7.3 Terminazione

Identificativo		UCBC_03 – Terminazione del sistema	**Data**	14/11/2025
			Versione	1.0
			Autore	A.B.
Descrizione		Lo UC permette l'arresto del sistema in modo ordinato, garantendo la chiusura corretta delle risorse.		
Attore principale		Amministratore di sistema (implicito, non utente finale)		
Attori secondari		NA		
Entry condition		Il sistema è in esecuzione sul server.		
Exit condition On success		Il sistema non è più disponibile agli utenti fino al successivo avvio.		
Exit condition On failure		Il sistema non riesce a completare l'arresto e rimane disponibile agli utenti.		
Flusso di eventi principale				
1	Amministratore	Invia un segnale di spegnimento al Sistema		
2	Sistema	Controlla che non ci siano connessioni ancora aperte da o verso l'esterno e, se non ci sono, termina l'esecuzione del sistema.		
I Flusso di Eventi Alternativo: I Dati Persistenti sono danneggiati				
2.a	Sistema	Notifica l'amministratore di connessioni ancora aperte e non effettua l'arresto.		
3.a	Amministratore	Attende che le connessioni ancora aperte si chiudano e riparte dal punto 1.		