UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA

Ingegneria del Software

UniClass - System Design Document

v1.3

DATA: 22/02/2026

Coordinatore del progetto:

Nome	Matricola
Giuseppe Sabetta	0512117895

Partecipanti:

Nome	Matricola
Giuseppe Sabetta	0512117895
Sara Gallo	0512117262
Saverio D'Avanzo	0512118330
Gerardo Antonio Cetrulo	0512117856

Scritto da:	Lucageneroso Cammarota (LC), Giuseppe Sabetta (GS) Sara Gallo (SG), Saverio D'Avanzo (SD), Gerardo Antonio Cetrulo (AC)

Revision History

Data	Versione	Descrizione	Autore
22/2/2026	1.3	Modifica e Analisi del Sistema Post-Refactoring	LC, GS
30/1/2025	1.2	Correzione del controllo degli accessi	GS, SG, SD, AC
21/12/2024	1.1	Revisione dei sottosistemi presenti	GS, SG, SD, AC
22/11/2024	1.0	System Design	GS, SG, SD, AC

Indice

1 Introduzione **4**
1.1 Scopo del Sistema 4
1.2 Obiettivi di Progettazione 4
1.3 Riferimenti 4
1.4 Panoramica 4

2 Architettura Software Attuale **5**

3 Architettura Software Proposta **5**
3.1 Panoramica 5
3.2 Decomposizione in Sottosistemi 6
3.3 Mappatura Hardware/Software 7
3.4 Gestione dei Dati Persistenti 8
3.4.1 Razionale del Cambiamento Architetturale 8
3.4.2 Nuovo Modello di Persistenza (Refactoring) 8
3.5 Controllo degli Accessi e Sicurezza 9
3.6 Controllo Globale del Software 10
3.7 Condizioni Limite 10

4 Servizi dei Sottosistemi **10**

1 Introduzione

1.1 Scopo del Sistema

Lo scopo di UniClass è di migliorare l'esperienza degli studenti universitari, semplificando le interazioni dei docenti con il sistema e con gli studenti stessi, permettere la reperibilità in tempo reale di informazioni sulle attività degli studenti, dei docenti e del personale accademico.

1.2 Obiettivi di Progettazione

La progettazione di UniClass ha richiesto maggiore rilievo sui seguenti obiettivi di progettazione:

- **Usabilità:** Il primo obiettivo è il punto cardine della progettazione di UniClass, poiché UniClass ha come compito quello di risultare accessibile facilmente, tramite un'interfaccia grafica leggibile per il carattere usato, presentando un contrasto tra i colori degli elementi grafici presenti nell'interfaccia e di un menù facilmente interagibile
- **Reliability:** Nonostante il fatto che UniClass sia stato ideato non principalmente per comunicazioni, bensì per reperibilità di informazioni ed interoperabilità tra gli utenti, ciò che si può modificare, inserire o eliminare all'interno del sistema può essere molto dannoso. Il goal di UniClass in questo caso è garantire operazioni sicure con una gestione di input per autenticazione e maggiori controlli sulla formattazione richiesta dalla piattaforma.
- **Performance:** UniClass deve essere in grado di gestire gli accessi di più di 50 persone connesse all'interno della piattaforma. Questa piattaforma presenta, per definizione del sistema proposto, picchi di connessioni negli orari con più affluenza in Ateneo.
- **Supportability:** Proprio per i picchi di utenza per localizzare aule, aule libere etc., il sistema deve presentare scalabilità. La modularità è un altro punto fondamentale per la possibilità di aggiungere funzionalità in maniera iterativa facilmente.

1.3 Riferimenti

Per la stesura del System Design di UniClass, si sono scelti due documenti di riferimento:

- Libro: Object-Oriented Software Engineering using UML, Patterns, and Java (3rd Edition)
- Documenti: Requirement Analysis Document, Problem Statement

1.4 Panoramica

Il documento di System Design di UniClass fornisce una struttura chiara in grado di spiegare come verranno soddisfatti i requisiti dal punto architetturale e strutturale del sistema.

2 Architettura Software Attuale

La progettazione dell'architettura software di UniClass è di tipo "green-field" e, di conseguenza, non esiste alcuna architettura software precedente da riprogettare o migliorare. Nonostante l'esplicita presa in considerazione dei sistemi esistenti utili all'utente per diversi tipi di servizi per il miglioramento della sua esperienza in ateneo, UniClass non rappresenta una versione simile di questi sistemi, bensì un agglomerato di tutte le possibili funzionalità che l'utente potrà gestire in un'unica piattaforma, per maggiore facilità e accessibilità. UniClass ha preso ispirazione precisamente da tre diversi sistemi per diverse funzionalità e architetture:

- **EasyCourse:** EasyCourse, un sistema utile per la visione degli orari delle varie lezioni all'interno dell'Università. UniClass ha deciso di ereditare le sue funzionalità e la sua idea architetturale, migliorando la visione attuale degli orari.
- **UNISA Lezioni:** Questo sistema è utile allo studente più per l'esperienza accademica in assenza di lezioni, come la visione di aule libere e annunci eventuali, ma anche per la rilevazione della presenza. Le funzionalità non sviluppate o sfruttate verranno riportate nel sistema UniClass, insieme a quelle citate precedentemente.
- **MyUniSalerno:** Questa piattaforma è stata creata per gestire le finanze e il libretto degli studenti universitari, includendo anche funzionalità opzionali come mappe e informazioni. UniClass, con l'obiettivo di diventare la piattaforma centralizzata per l'esperienza universitaria, ha sviluppato un'architettura ispirata a queste funzionalità e la implementa per una delle sue principali operazioni, garantendo comunque consistenza con le funzionalità della piattaforma e coesione, rispettando i confini descritti.

3 Architettura Software Proposta

3.1 Panoramica

L'architettura Software proposta si può visionare attraverso un layering nelle parti principali del sistema, differenziando la vista del client, la visione back-end di gestione della logica di business e la persistenza. Utilizziamo, per una visione bird's eye della nostra architettura, un pattern architetturale noto per la differenziazione dei concetti del nostro sistema, ovvero layering three-tier:

- **Layer di Presentazione:** Questo layer rappresenta il livello di presentazione del nostro sistema, ovvero la visione del client in merito alle operazioni e funzionalità offerte da UniClass. Come componenti principali di questo layer abbiamo l'interfaccia utente.
- **Layer di Business:** Il layer di business, invece, rappresenta l'insieme di tutti i componenti utilizzati per rappresentare e implementare la logica di business del sistema UniClass. Di questo layer fanno parte principalmente i componenti che gestiscono le macro-operazioni descritte nel Requirements Analysis Document e si differenziano in base al tipo di utente che naviga all'interno della piattaforma.

- **Layer di Persistenza:** il layer di persistenza rappresenta la gestione dei dati persistenti, rendendoli consistenti e reperibili in caso di necessità.

3.2 Decomposizione in Sottosistemi

Prima di mostrare il diagramma rappresentante la decomposizione totale in sottosistemi, attraverso layering in uno schema three-tier, bisogna prendere in considerazione lo sviluppo dei vari sottosistemi di UniClass, in base al livello d'appartenenza:

- **Presentation Layer:** Per il presentation layer, vi è solamente l'interfaccia utente, la GUI con cui interagisce il client per l'esecuzione delle sue operazioni e la visione delle informazioni di rilevanza.
- **Business Layer:** Il Business Layer comprende i vari sottosistemi che implementano la logica di business, quindi la gestione dei vari controlli e le varie esecuzioni di operazioni del client. Il Business Layer si interfaccia sia con il presentation layer, per occuparsi di controllare, permettere o negare le varie operazioni o visualizzazioni di qualsiasi tipo sul sistema UniClass da parte dell'utente, che con il persistence layer, per occuparsi di rendere persistenti determinati dati. Nel Business Layer abbiamo vari sottosistemi e ognuno di questi comunica con un sottoinsieme degli altri in base alla natura dei servizi offerti. Introduciamo ora i vari sottosistemi del business layer:
  - **Orari:** gestisce la visualizzazione, creazione e modifica degli orari settimanali, dell'agenda, delle aule e le lezioni in sé, seguendo le indicazioni del Requirements Analysis Document
  - **Utenti**: Gestisce il ciclo di vita degli utenti e l'autenticazione. L'accesso alle funzionalità è centralizzato tramite il componente **UserDirectory** (pattern Facade), che disaccoppia i moduli client dalla complessità interna del dominio Utenti (composizione ruoli). I servizi specifici legacy (es. StudenteService) sono stati accorpati per ridurre il Ripple Effect.
  - **Conversazioni:** gestisce le conversazioni tra i vari utenti registrati, per messaggistica e per annunci. Ogni conversazione può generare delle notifiche per avvisare l'utente a leggere i messaggi non letti e rispondere
- **Persistence Layer:** per il Persistence Layer, si è deciso di utilizzare un Database in grado di memorizzare e rendere i dati richiesti persistenti.

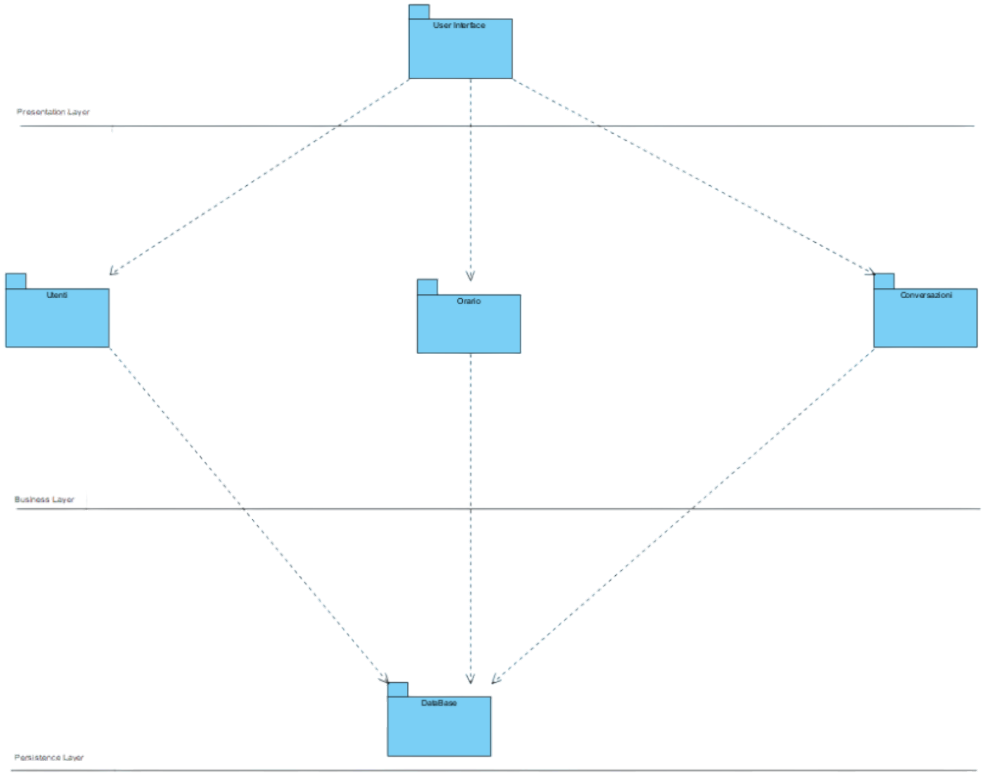


Figura 1: Decomposizione in Sottosistemi - Architettura Three-Tier

3.3 Mappatura Hardware/Software

UniClass è una piattaforma presente in un sistema distribuito, dove si ha la possibilità di accedere ai servizi offerti dal server da qualsiasi possibile macchina client, che sia un personal computer, un dispositivo mobile etc. Il client comunicherà con il server e quest'ultimo con lo storage (Database) per reperire le informazioni richieste.

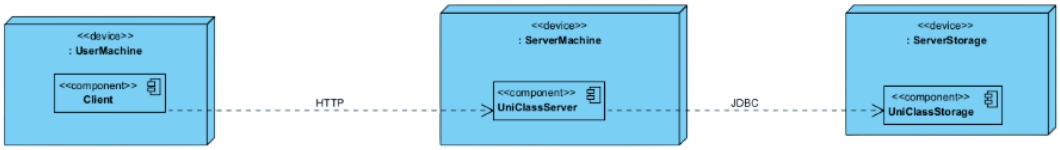


Figura 2: Mappatura Hardware/Software - Diagramma di Deployment

3.4 Gestione dei Dati Persistenti

Per la gestione delle operazioni transazionali all'interno del sistema UniClass, i dati vengono resi persistenti utilizzando un Database relazionale (PostgreSQL) e la Java Persistence API (JPA) come standard di mapping oggetto-relazionale (ORM).

3.4.1 Razionale del Cambiamento Architetturale

La versione precedente del sistema adottava una strategia di persistenza basata sull'ereditarietà rigida (*Joined Table Strategy*). Sebbene corretta dal punto di vista normalizzato, questa scelta presentava i limiti evidenziati nella Change Request:

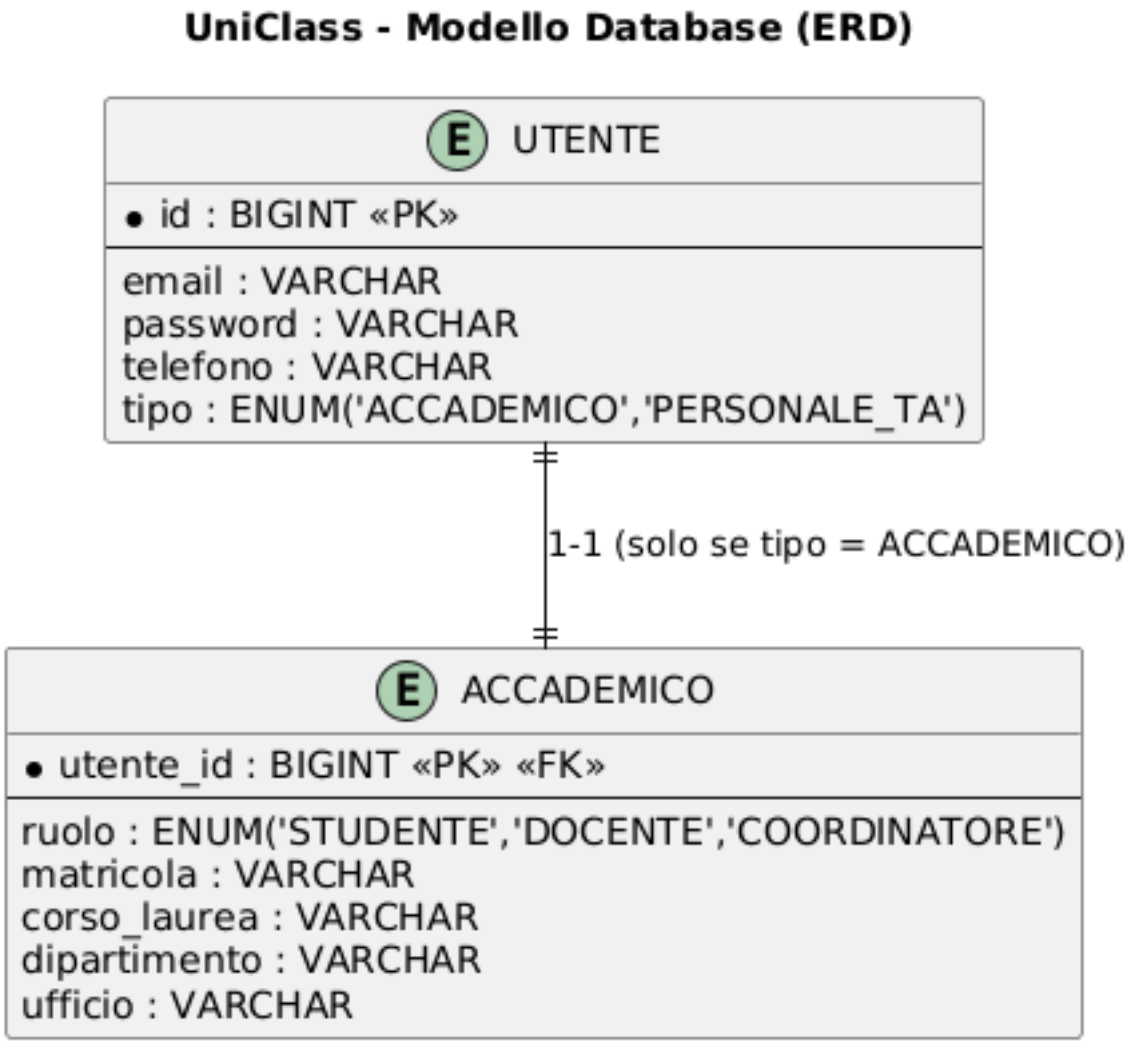
- **Verticalizzazione dei Dati**: Le informazioni specifiche (es. matricola studente, dipartimento docente) erano frammentate in tabelle satellite distinte.
- **Overhead Query**: La ricostruzione di un'entità completa richiedeva costose operazioni di JOIN su molteplici tabelle, impattando negativamente sulle performance di autenticazione e recupero profilo.
- **Rigidità Ruoli**: La struttura impediva a un utente di possedere più ruoli contemporaneamente, vincolando l'identità digitale dell'utente alla sua classificazione funzionale.

3.4.2 Nuovo Modello di Persistenza (Refactoring)

In risposta a tali criticità, il modello dati è stato ristrutturato secondo il principio della **Composizione** e della centralizzazione. La nuova strategia si fonda sui seguenti pilastri:

1. **Collapse delle Gerarchie**: Le entità legacy `Studente`, `Docente` e `Coordinatore` non sono più mappate come tabelle distinte ereditate. I loro attributi specifici sono stati consolidati o promossi alla tabella principale.
2. **Modello Ibrido Utente-Accademico**: È stata introdotta una distinzione tra l'entità `Utente` (tabella principale contenente credenziali e dati anagrafici comuni) e l'entità `Accademico` (tabella collegata tramite relazione opzionale 1-a-1).
    - Questo approccio elimina la necessità di JOIN obbligatorie per gli utenti non accademici (es. Personale TA), ottimizzando i tempi di accesso.
    - Permette la gestione del **Multi-Ruolo**: il ruolo non è più determinato dalla tabella in cui risiede il record, ma diventa un attributo (o una relazione separata) dell'entità Utente, gestito via codice attraverso la `UserDirectory`.
3. **Accesso ai Dati**: Le query JPQL sono state riscritte per sfruttare questa nuova topologia, riducendo il numero di join necessari nelle operazioni CRUD critiche (es. *login*, *profilo utente*).

La Figura **??** (Modello ER aggiornato) illustra la nuova struttura delle relazioni, che favorisce l'estensibilità futura (es. aggiunta di nuovi ruoli) senza impatti strutturali sullo schema del database.



3.5 Controllo degli Accessi e Sicurezza

La seguente tabella mostra la matrice di controllo degli accessi per i diversi attori del sistema:

Ruolo	**Orari**	**Aule**	**Utenti**	**Conversazioni**
Accademico	visualizzaOrari	visualizzaAule	visualizzaProfilo	creaMessaggio inviaBroadcast
Personale TA	visualizzaOrari	visualizzaAule	creaUtente modificaUtente rimuoviUtente	X

Tabella 2: Controllo degli Accessi basato su Ruoli

Nota tecnica: Il controllo degli accessi è implementato tramite i metodi di `UserDirectory` che verificano i Ruoli associati all'Utente autenticato. Un Utente con ruoli multipli eredita le funzionalità di tutti i ruoli assegnati.

3.6 Controllo Globale del Software

In base ai diagrammi dinamici presenti all'interno del documento di riferimento "RAD", possiamo notare che il controllo del software è basato su un'architettura di tipo "stairs", per aumentare l'efficienza e la distribuzione della gestione dei vari oggetti presenti nella piattaforma. Tutto ciò dovrà rispettare comunque il paradigma MVC (Model-View-Control). In base al tipo di autenticazione (eventuale) del client, l'utente effettua, tramite il web browser (UserMachine), la richiesta al server di visualizzazioni o operazioni riguardanti informazioni distribuite per la piattaforma. Il server allocherà un thread per la richiesta e restituirà il risultato delle operazioni richieste inizialmente. Per la gestione della concorrenza, prima di ogni commit di transazione, vi saranno diversi controlli, gestiti sia automaticamente grazie alle API disponibili in Java, che manualmente dagli sviluppatori, rispettando così le proprietà ACID.

3.7 Condizioni Limite

Sarà l'amministratore di UniClass a gestire, visualizzare e operare in base alle possibili problematiche avute durante le varie fasi delle richieste al server della piattaforma, come le eccezioni o perdite di consistenza/informazioni.

4 Servizi dei Sottosistemi

Per la comprensione di questo punto, vedremo per ogni sottosistema (visibile nel punto 3.2), i servizi offerti:

- **Utenti**: Questo sottosistema espone un'interfaccia unificata attraverso il componente `UserDirectory`. I servizi offerti includono:
  - *Autenticazione*: Verifica delle credenziali e caricamento dei Ruoli dinamici.
  - *Gestione Profili*: Recupero e modifica delle informazioni utente (anagrafiche, contatti).
  - *Amministrazione*: Creazione account, assegnazione Ruoli, attivazione e rimozione utenti.

  I sottoservizi specifici (StudenteService, DocenteService) sono stati rimossi in favore di una logica centralizzata che gestisce le differenze comportamentali tramite la composizione dei ruoli.
- **Orari:** questo sottosistema garantisce il servizio sulla gestione degli orari presenti tra i vari corsi dell'Ateneo, comprendendo le lezioni, le agende degli utenti e le aule.
- **Conversazioni:** questo sottosistema permette il servizio di gestione delle conversazioni, come la creazione e scrittura di chat/avvisi tra studenti, docenti e coordinatori.

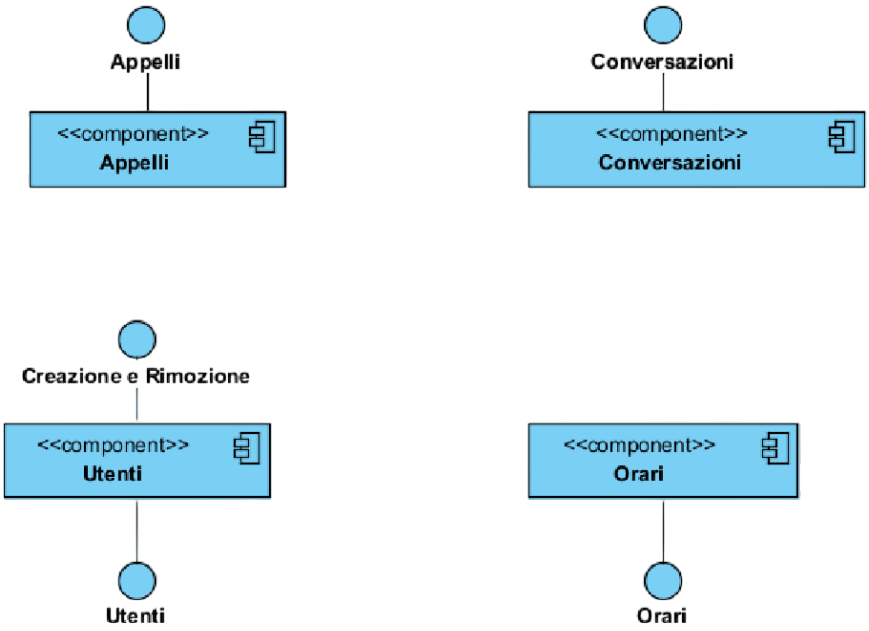


Figura 3: Diagramma dei Componenti - Servizi dei Sottosistemi