Software Requirements Specification

for

Portale Universitario

Version 1.5

Prepared by Gianluca D'Isanto

Università degli Studi della Campania "Luigi Vanvitelli"
Sistemi Web e Basi di Dati

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Inizializzazione Documento	23/06/2025	Creazione della bozza iniziale del documento SRS. Definizione della struttura generale del documento, introduzione, scopo e ambito del sistema.	1.0
Definizione Requisiti funzionali	23/06/2025	Delineazione dei primi requisiti funzionali del sistema. Inclusione delle descrizioni generali delle interazioni utente-sistema.	1.1
Inizializzazione ERD	25/06/2025	Creazione iniziale del Diagramma E-R. Definizione delle entità principali e delle prime relazioni.	1.2
Progettazione logica	27/06/2025	Ristrutturazione dell'ERD con aggiunta delle tabelle	1.3
Aggiunta dei diagrammi UML	30/06/2025	Aggiunta del diagramma dei casi d'uso, delle classi e di deployment	1..4
Correzioni minori	03/07/2025	Correzioni minori	1.5

1. Introduction

1.1 Purpose

Il presente documento descrive i requisiti software del progetto “Portale Universitario”, sviluppato nell’ambito di un’attività didattica universitaria.
Il sistema è stato progettato e implementato a scopo formativo, con l’obiettivo di approfondire competenze pratiche nello sviluppo di applicazioni web e nella progettazione di basi di dati.

Il prodotto consiste in un sistema web pensato per supportare il personale accademico nella gestione delle principali attività relative agli esami universitari, tra cui la creazione dei corsi, la pianificazione degli appelli, la registrazione delle valutazioni e la generazione di report.

Attualmente, il sistema è focalizzato esclusivamente sulle funzionalità rivolte ai docenti, ma è stato progettato con un’architettura modulare e facilmente estendibile, così da poter integrare in futuro funzionalità dedicate anche agli studenti.

1.2 Intended Audience and Reading Suggestions

Il presente documento è destinato a diversi tipi di lettori che possono essere coinvolti nello sviluppo, nella verifica e nella valutazione del sistema:

- **Progettisti**
- **Sviluppatori**
- **Docenti e utenti finali**
- **Tester**
- **Manager**

1.3 Product Scope

L’obiettivo del software è semplificare le attività svolte dal personale docente, come l’organizzazione e la gestione dei corsi, degli appelli d’esame, delle valutazioni e delle comunicazioni tra docenti e sistema, migliorando l’efficienza nella registrazione delle valutazioni e nella consultazione delle informazioni accademiche.

2. Overall Description

2.1 Product Perspective

Portale Universitario è un'applicazione software progettata per semplificare la gestione degli esami universitari. Consente ai docenti di personalizzare gli indicatori di valutazione per prove scritte e colloqui orali, adattandoli alle esigenze specifiche di ogni materia e contesto.

Il sistema permette di inserire e organizzare gli argomenti trattati durante le prove, garantendo una tracciabilità precisa dei contenuti valutati e assicurando trasparenza nell'assegnazione dei voti. Il sistema permette di generare un report dettagliato per ogni esame completato, ma anche per ogni corso, includendo alcune statistiche principali.

Questo strumento nasce con l'obiettivo di offrire ai docenti una soluzione efficiente, che unisce praticità, precisione e trasparenza in un'unica piattaforma.

2.2 Product Functions

Il sistema deve fornire le seguenti funzionalità di alto livello:

- Gestione dei corsi, inclusa la creazione, modifica e cancellazione.
- Pianificazione e programmazione degli appelli d'esame.
- Registrazione delle valutazioni degli studenti, comprensive di voti e annotazioni.
- Personalizzazione degli indicatori di valutazione per le prove scritte e i colloqui orali, al fine di adattarsi alle specifiche esigenze.
- Organizzazione e tracciamento degli argomenti oggetto di valutazione, garantendo la tracciabilità e la coerenza nella valutazione.
- Generazione di report dettagliati in formato PDF.
- Archiviazione storica delle sessioni d'esame, con possibilità di consultazione dei dati nel tempo.
- Pannello notifiche dedicato ai docenti, per segnalare aggiornamenti, scadenze e comunicazioni rilevanti.

2.3 Operating Environment

Per l'utilizzo della web application è necessario utilizzare la seguente configurazione:

- **Web Server**: Apache Versione 2.4.58 con OpenSSL 1.1.1n e PHP 8.2.12

- **Database**: MariaDB Versione 10.4.32

2.4 Design and Implementation Constraints

L'implementazione della web application dovrà tenere conto dei seguenti vincoli che limiteranno le scelte tecniche e progettuali degli sviluppatori:

- La web application deve essere conforme alle normative vigenti in materia di protezione dei dati personali, in particolare al GDPR per i cittadini dell'Unione Europea.
- L'intera applicazione, inclusi tutti i componenti dell'interfaccia utente, la documentazione e le comunicazioni interne, deve essere interamente realizzata in lingua italiana.
- La web application deve essere progettata e sviluppata per consentire interventi di manutenzione ordinaria e straordinaria, facilitando aggiornamenti e modifiche senza compromettere la disponibilità né la funzionalità del servizio.
- Devono essere implementati meccanismi robusti di autenticazione e autorizzazione, al fine di proteggere l'accesso a dati sensibili e funzionalità critiche, garantendo la sicurezza complessiva dell'applicazione.
- La web application dovrà essere compatibile con i principali browser moderni, garantendo un'esperienza utente uniforme.
- Il sistema deve essere progettato considerando la possibilità di un aumento del numero di utenti e dei dati gestiti, per evitare limitazioni prestazionali future.

2.5 Assumptions and Dependencies

Durante la definizione dei requisiti del software, si è fatto riferimento alle seguenti assunzioni, che potrebbero influenzare lo sviluppo e il funzionamento del sistema nel caso in cui risultassero errate o venissero meno:

- Si assume che il Web server Apache, PHP e MySQL utilizzati siano configurati correttamente e disponibili nella versione minima richiesta per garantire il funzionamento ottimale della web application.
- Si presume che gli utenti abbiano accesso a dispositivi con capacità hardware adeguate e browser aggiornati compatibili con le tecnologie web adottate.
- Il progetto dipende dall'utilizzo di librerie o framework PHP di terze parti che, se aggiornati o deprecati, potrebbero richiedere modifiche al codice o alla struttura del software.
- Si assume che la connettività di rete e l'infrastruttura server garantiscano una disponibilità continua del servizio.

2.6 Uses Case Diagram

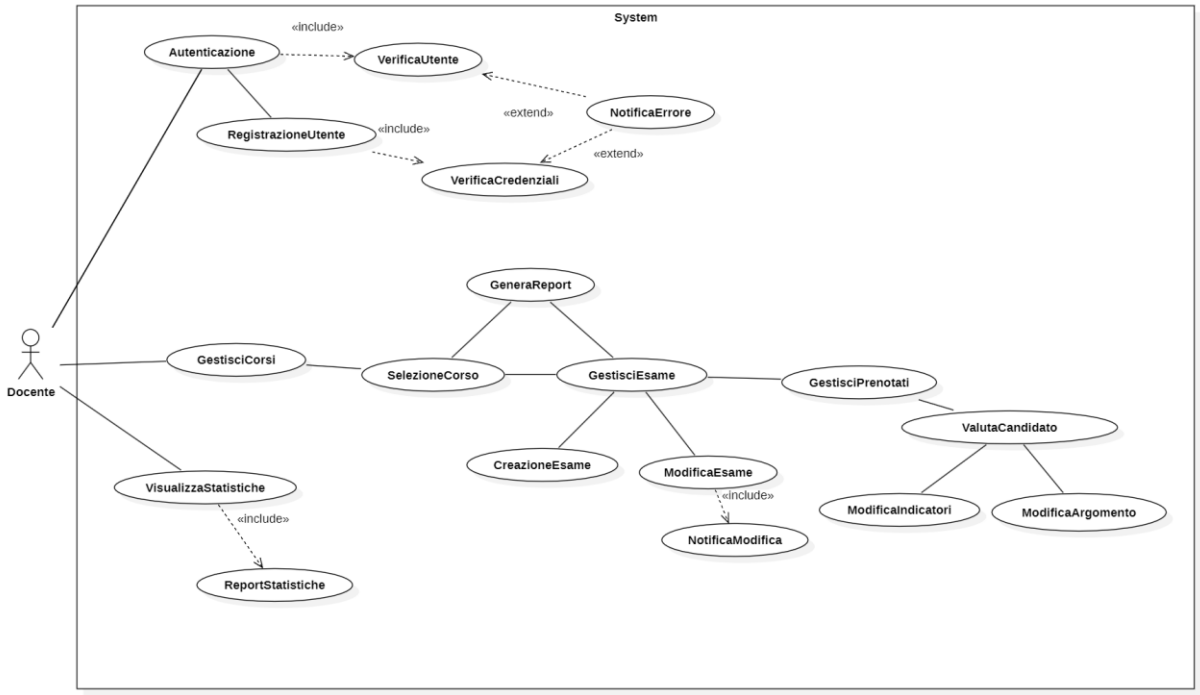

Figura 1 Use case diagram

2.7 Classes Diagram

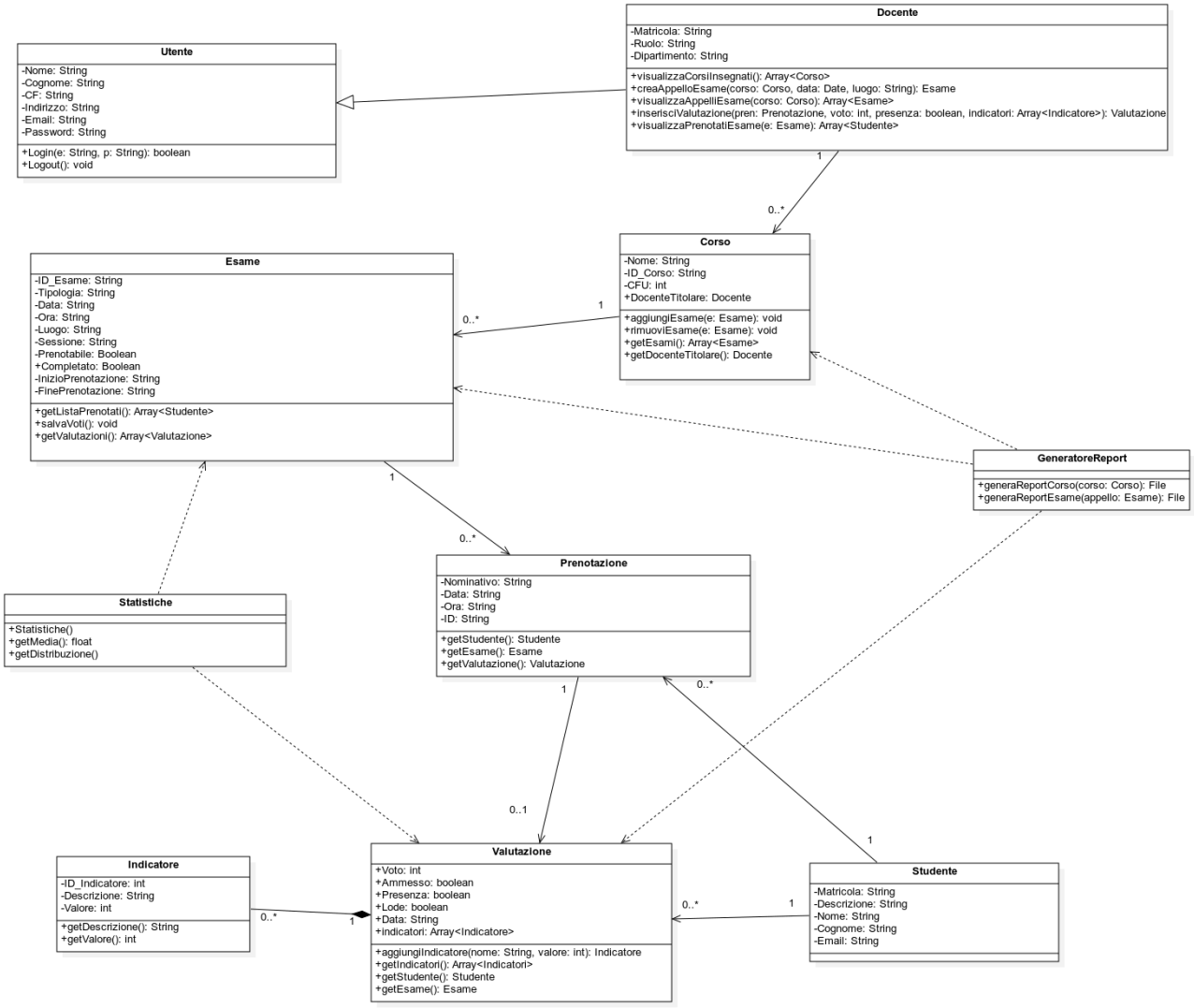

Figura 2 Class diagram

3. External Interface Requirements

3.1 User Interfaces

3.1.1 Login

La pagina di login è la prima interfaccia visualizzata dall'utente al momento dell'apertura dell'applicazione. Questa pagina consente agli utenti di accedere al loro account esistente o di procedere alla registrazione di un nuovo account.

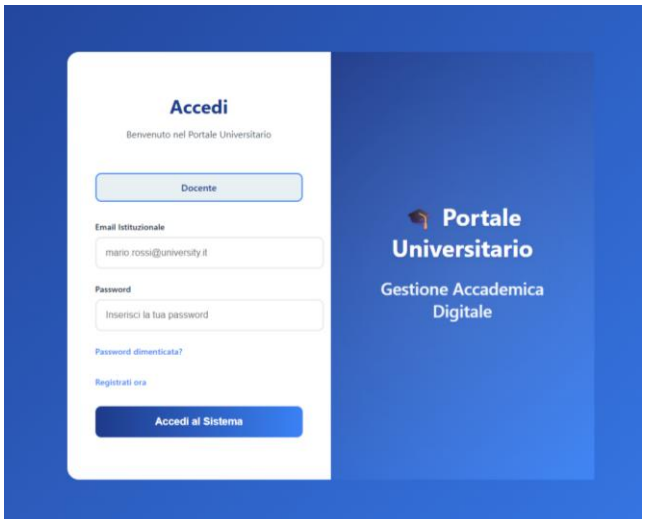

Figura 4 Login interface

3.1.2 Registrazione

La pagina di registrazione è progettata per consentire agli utenti di creare un nuovo account. Se un utente è già registrato, può accedere alla pagina di login tramite un link presente nella pagina di registrazione. Dopo un processo di registrazione riuscito, l'utente viene automaticamente reindirizzato alla pagina di login per effettuare l'accesso.

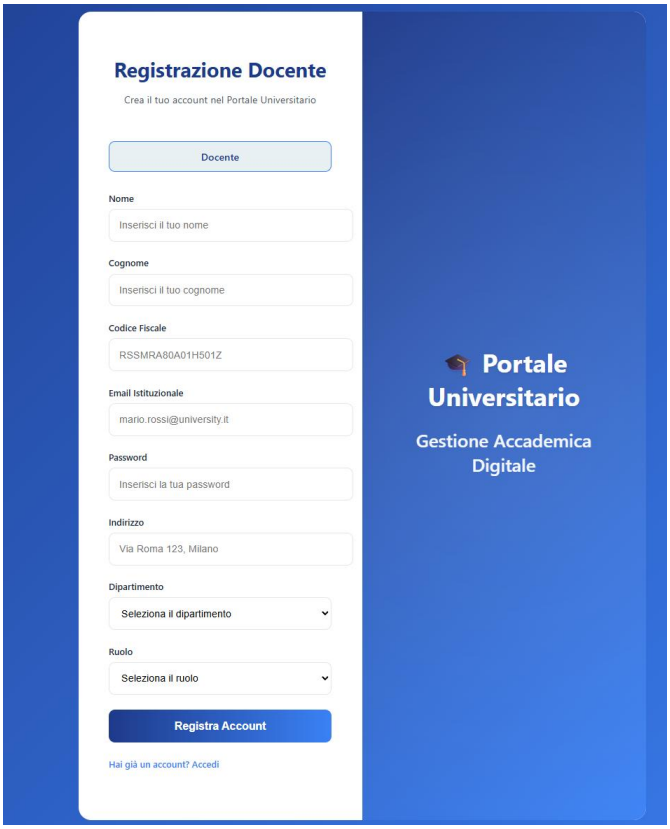

Figura 3 Registration interface

3.2 Software Interfaces

Il prodotto si integra con i seguenti componenti software e tecnologie:

- **Linguaggi e tecnologie client:** HTML, CSS e Javascript, utilizzati per la realizzazione dell'interfaccia utente, la gestione dell'interazione lato client e la validazione di form.
- **Linguaggio di scripting lato server:** PHP, impiegato per la logica di backend, la gestione delle richieste e l'interazione con il database.
- **Database:** MySQL, usato per l'archiviazione e la gestione persistente dei dati relativi a utenti, sessioni d'esame, valutazioni e notifiche.
- **Librerie PHP:**
    - **FPDF**, per la generazione dinamica di report in formato PDF.
    - **PHPUnit**, utilizzato per l'esecuzione di test automatizzati sul codice PHP.

Componenti PHP principali e loro funzionalità:

- **login.php**: pagina di accesso dove gli utenti inseriscono le proprie credenziali per effettuare il login. Offre inoltre un link per accedere alla pagina di registrazione.
- **loginAuth.php**: script che gestisce il processo di autenticazione, verifica le credenziali nel database e autorizza l'accesso al sistema.
- **registrazione.php**: pagina con modulo per la registrazione di nuovi utenti. Visualizza eventuali messaggi di errore e fornisce un link per tornare al login.
- **registrazioneServer.php**: script che elabora i dati forniti durante la compilazione del form di registrazione, salva i dati degli utenti nel database.
- **dashboard.php**: schermata principale post-login che offre accesso alle funzionalità chiave come gestione corsi, esami, notifiche e report. Include un menu di navigazione.
- **corsi.php**: gestione e visualizzazione dell'elenco dei corsi disponibili per il docente.
- **report_corso.php**: genera report dettagliati relativi ai corsi.
- **esami.php**: visualizzazione e gestione degli appelli d'esame collegati ai corsi.
- **nuovo_esame.php**: modulo per la creazione di nuovi appelli d'esame.
- **modifica_esame.php**: consente di modificare le informazioni di un appello d'esame esistente.
- **elimina_esame.php**: permette la cancellazione di un appello d'esame selezionato.
- **report_esame.php**: genera report specifici relativi a singole sessioni d'esame, con valutazioni.

- **salva_voto.php**: gestisce l'inserimento dei voti e delle valutazioni associate agli esami.
- **visualizzaEsami.php**: mostra le informazioni relative ad un appello d'esame, includendo gli studenti prenotati, fornendo la possibilità di compilare le valutazioni per ciascuno di essi, con presenza, voto, indicatori di performance e argomenti di valutazione.
- **rimuovi_notifica.php**: consente la rimozione di notifiche dal pannello dedicato agli utenti.
- **statistiche.php**: presenta analisi e statistiche aggregate relative a esami, voti e partecipazioni.
- **profilo.php**: visualizza le informazioni personali dell'utente.
- **logout.php**: termina la sessione utente e reindirizza alla pagina di login.

3.3 Communications Interfaces

Il sistema si basa su un'infrastruttura in cui il web server Apache gestisce tutte le richieste dei browser degli utenti e recupera le risorse dal database MySQL, che si occupa della gestione dei dati. La comunicazione tra il server e il database avviene tramite query SQL, mentre l'interazione tra il client e il server è gestita attraverso messaggi HTTP.

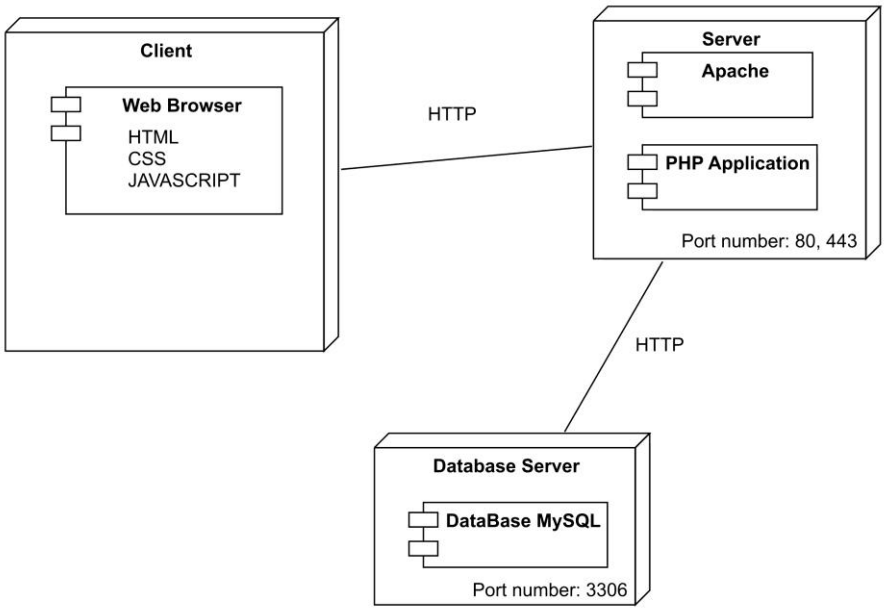

Figura 5 Deployment diagram

4. System Features

4.1 System Feature: Login utente

4.1.1 Description and Priority

Questa funzionalità permette agli utenti registrati di accedere al sistema inserendo la propria email e password.

4.1.2 Stimulus/Response Sequences

Precondizioni:

- L'utente ha precedentemente effettuato la registrazione nel sistema.
- Il sistema è attivo e raggiungibile tramite browser web.

Sequenza:

1. L'utente apre il portale web dell'applicazione.
2. Il sistema mostra il modulo di autenticazione, con i campi per email e password.
3. L'utente inserisce le proprie credenziali e conferma l'invio.
4. Il sistema riceve i dati e verifica:
    - Che l'indirizzo email sia presente nel database.
    - Che la password corrisponda a quella associata all'email.
5. In caso di credenziali non valide:
    - Il sistema mostra un messaggio di errore.
    - L'utente può riprovare.
6. In caso di credenziali corrette:
    - Il sistema inizializza la sessione dell'utente.
    - Reindirizza l'utente alla dashboard principale.

4.1.3 Functional Requirements

- **REQ-1**: Il sistema deve fornire una pagina di login accessibile pubblicamente, contenente un modulo con i campi "email" e "password".

- **REQ-2**: Il sistema deve validare che i campi del modulo non siano vuoti prima dell'invio dei dati.
- **REQ-3**: Il sistema deve verificare la corrispondenza tra le credenziali inserite dall'utente e quelle memorizzate nel database.
- **REQ-4**: In caso di credenziali errate, il sistema deve mostrare un messaggio di errore.
- **REQ-5**: In caso di credenziali corrette, il sistema deve avviare una sessione utente e reindirizzare l'utente autenticato alla pagina principale.
- **REQ-6**: Il sistema deve impedire l'accesso a funzionalità riservate se l'utente non ha effettuato il login.
- **REQ-7**: Il sistema deve mantenere una sessione attiva per l'utente autenticato, che può essere utilizzata per verificare l'accesso durante le operazioni successive.

4.2 System Feature: Registrazione utente

4.2.1 Description and Priority

La funzionalità consente a un nuovo utente di creare un account sulla piattaforma inserendo i dati richiesti tramite un modulo.

4.2.2 Stimulus/Response Sequences

Precondizioni

- L'utente non deve essere autenticato.
- Deve essere disponibile una connessione attiva al database.

Sequenza:

1. L'utente accede alla pagina di registrazione dal link disponibile nella pagina di login.
2. L'utente compila il modulo con: nome, cognome, email, password e altri campi.
3. L'utente invia il modulo.
4. Il sistema verifica che i dati siano corretti e che l'email non sia già registrata.
5. Se tutto è valido, i dati vengono salvati nel database.
6. Il sistema mostra un messaggio di avvenuta registrazione.
7. L'utente viene reindirizzato alla pagina di login.

4.2.3 Functional Requirements

- **REQ-1**: Il sistema deve fornire un modulo di registrazione accessibile da utenti non autenticati.
- **REQ-2**: Il modulo deve contenere i campi obbligatori: nome, cognome, email, password e altri campi.
- **REQ-3**: Il sistema deve validare che tutti i campi richiesti siano compilati correttamente prima dell'invio, per esempio email in formato valido.
- **REQ-4**: Il sistema deve verificare che l'email non sia già associata a un account esistente.
- **REQ-5**: In caso di errori, il sistema deve informare l'utente con un messaggio esplicito.
- **REQ-6**: In caso di successo, il sistema deve salvare i dati dell'utente nel database in forma sicura.
- **REQ-7**: Il sistema deve notificare all'utente l'avvenuta registrazione e reindirizzarlo alla pagina di login, senza avviare una sessione (login da effettuare).

4.3 System Feature: Rimozione notifica

4.3.1 Description and Priority

Questa funzionalità consente al docente di gestire il proprio pannello notifiche, rimuovendo singoli avvisi non più rilevanti.

4.3.2 Stimulus/Response Sequences

Precondizioni:

- L'utente è autenticato.
- Sono presenti una o più notifiche nel pannello notifiche.

Sequenza:

1. L'utente accede alla dashboard.
2. Il sistema visualizza l'elenco delle notifiche attive, con l'opzione "Rimuovi" accanto a ciascuna.
3. Il docente clicca sul pulsante "Rimuovi" per una specifica notifica.
4. Il sistema elimina la notifica dal database e aggiorna la lista.

4.3.3 Functional Requirements

- **REQ-1**: Il sistema deve mostrare un elenco aggiornato delle notifiche disponibili per l'utente autenticato.
- **REQ-2**: Il sistema deve fornire, per ciascuna notifica, un controllo "Rimuovi" accessibile all'utente.
- **REQ-3**: Alla conferma dell'eliminazione, il sistema deve rimuovere la notifica dal database e aggiornare immediatamente l'interfaccia utente.

4.4 System Feature: Visualizzazione profilo

4.4.1 Description and Priority

Questa funzionalità consente all'utente di accedere ai propri dati personali all'interno dell'applicazione e visualizzare le informazioni di registrazione salvate nel sistema.

4.4.2 Stimulus/Response Sequences

Precondizioni:

- L'utente è autenticato.
- Le informazioni del profilo sono presenti nel database.

Sequenza:

1. L'utente accede alla dashboard.
2. L'utente clicca su "Profilo".
3. Il sistema carica e visualizza una pagina con i dati del docente: nome, cognome, email, ruolo, dipartimento).
4. L'utente può solo consultare le informazioni, senza modificarle.
5. Il sistema mostra eventuali messaggi informativi o errori in caso di problemi di caricamento.

4.4.3 Functional Requirements

- **REQ-1**: Il sistema deve fornire un collegamento accessibile dalla dashboard per accedere al profilo utente.

- **REQ-2**: Il sistema deve recuperare dal database i dati personali associati all'account dell'utente autenticato.
- **REQ-3**: Il sistema deve visualizzare i seguenti campi: nome, cognome, email, ruolo, dipartimento.
- **REQ-4**: L'interfaccia della pagina profilo deve essere in sola lettura (non modificabile).
- **REQ-5**: Il sistema deve gestire errori di caricamento dati mostrando un messaggio d'errore adeguato.

4.5 System Feature: Generazione report

4.5.1 Description and Priority

Questa funzionalità permette al docente di generare e scaricare report dettagliati in formato PDF relativi ai corsi e alle singole sessioni d'esame.

4.5.2 Stimulus/Response Sequences

Precondizioni:

- L'utente è autenticato come docente.
- Sono presenti dati relativi a corsi o esami nel sistema.

Sequenza:

1. Il docente accede alla sezione "Corsi" (o "Esami").
2. Il docente individua un corso o una sessione d'esame e clicca sul pulsante "Genera Report" (per corso o esame).
3. Il sistema raccoglie i dati relativi all'elemento selezionato (studenti, voti, …).
4. Il sistema genera un documento PDF strutturato.

4.5.3 Functional Requirements

- **REQ-1**: Il sistema deve rendere disponibile un pulsante "Genera Report" in corrispondenza di corsi o esami selezionabili.
- **REQ-2**: Il sistema deve raccogliere i dati necessari relativi all'elemento selezionato.
- **REQ-3**: Il sistema deve generare un report PDF strutturato, con intestazioni, tabelle e riepiloghi.

- **REQ-4**: Il sistema deve includere nei report tutti i campi rilevanti e le valutazioni associate.
- **REQ-5**: Il sistema deve generare un messaggio d'errore se insorgono errori di generazione.

4.6 System Feature: Creazione esame

4.6.1 Description and Priority

La funzionalità consente al docente di creare un nuovo esame, associandolo a un corso e impostando i dettagli necessari.

4.6.2 Stimulus/Response Sequences

Precondizioni:

- L'utente è autenticato.
- È presente almeno un corso associato al docente.

Sequenza:

1. Il docente seleziona un corso dall'elenco di quelli assegnati.
2. Visualizza la sezione dedicata agli esami del corso.
3. Clicca sull'opzione per creare un nuovo esame.
4. Il sistema mostra un modulo da compilare con i dettagli dell'esame (data, ora, tipo, aula, descrizione).
5. Il docente inserisce i dati richiesti e conferma la creazione.
6. Il sistema verifica la correttezza e la completezza dei dati inseriti.
    - In caso di errore, mostra un messaggio e richiede la correzione.
    - In caso di esito positivo, registra l'esame nel sistema.
7. Il sistema aggiorna la lista esami del corso selezionato e notifica l'avvenuta creazione.

4.6.3 Functional Requirements

- **REQ-1**: Il sistema deve permettere ai docenti autenticati di accedere a un modulo per la creazione di una nuova sessione d'esame.
- **REQ-2**: Il sistema deve validare i dati inseriti nel modulo prima di registrarli nel database.
- **REQ-3**: Il sistema deve mostrare un messaggio di conferma al termine della creazione.
- **REQ-4**: Il sistema deve mostrare messaggi di errore chiari in caso di input non valido o problemi con il database.
- **REQ-5**: Il sistema deve impedire l'accesso alla funzionalità a utenti non autenticati o non autorizzati.

4.7 System Feature: Modifica di un esame

4.7.1 Description and Priority

La funzionalità permette ai docenti di aggiornare le informazioni relative a un esame già creato.

4.7.2 Stimulus/Response Sequences

Precondizioni:

- L'utente è autenticato.
- È presente almeno un esame, associato ad un corso, precedentemente creato.

Sequenza:

1. Il docente visualizza la sezione dedicata alla lista degli esami.
2. Il docente seleziona un esame dalla lista, cliccando su "Modifica".
3. Il sistema mostra un modulo da compilare con i dettagli dell'esame (data, ora, tipo, aula, descrizione).
4. Il docente inserisce i dati richiesti e conferma la modifica.
5. Il sistema verifica la correttezza e la completezza dei dati inseriti.
    i. In caso di errore, mostra un messaggio e richiede la correzione.
    ii. In caso di esito positivo, registra le modifiche dell'esame nel sistema.

6. Il sistema aggiorna la lista esami del corso selezionato e notifica l'avvenuta modifica.

4.7.3 Functional Requirements

- **REQ-1**: Il sistema deve permettere ai docenti autenticati di accedere a un modulo per la modifica di un esame precedentemente creato.
- **REQ-2**: Il sistema deve validare i dati inseriti nel modulo prima di registrarli nel database.
- **REQ-3**: Il sistema deve recuperare e mostrare i dati correnti dell'esame selezionato.
- **REQ-4**: Il sistema deve consentire la modifica dei dati e la successiva memorizzazione.
- **REQ-5**: Il sistema deve mostrare messaggi di errore chiari in caso di input non valido o problemi con il database.

4.8 System Feature: Eliminazione di un esame

4.8.1 Description and Priority

La funzionalità consente ai docenti di di eliminare un esame non più necessario.

4.8.2 Stimulus/Response Sequences

Precondizioni:

- L'utente è autenticato.
- È presente almeno un esame associato ad un corso, a sua volta associato al docente.

Sequenza:

1. Il docente visualizza la sezione dedicata agli esami.
2. Il docente seleziona un esame dalla lista, cliccando su "Modifica".
3. Il sistema mostra un modulo con i dettagli dell'esame.
4. Il docente clicca su "Elimina esame".
5. Il sistema chiede conferma dell'eliminazione:
    - Se ci sono studenti prenotati, il sistema chiede la conferma, in tal caso verranno eliminate anche le prenotazioni.

- Se ci sono studenti “valutati”, il sistema rende impossibile l’eliminazione dell’esame.

6. Il sistema verifica la correttezza e la coerenza dei dati:
   - In caso di errore, mostra un messaggio e richiede la correzione.
   - In caso di esito positivo, registra l’esame nel sistema.
7. Il sistema aggiorna la lista esami del corso selezionato e notifica l’avvenuta eliminazione.

4.8.3 Functional Requirements

- **REQ-1**: Il sistema deve permettere ai docenti autenticati di accedere a un modulo per l’eliminazione di un esame.
- **REQ-2**: Il sistema deve permettere la cancellazione di un esame.
- **REQ-3**: Il sistema deve richiedere una conferma esplicita prima dell’eliminazione definitiva di un esame.
- **REQ-4**: Il sistema deve mostrare messaggi di errore chiari in caso di input non valido o problemi con il database.

4.9 System Feature: Visualizzazione e valutazione esami

4.9.1 Description and Priority

La funzionalità consente ai docenti di visualizzare i dettagli di un esame (tipologia, data, ora, stato delle prenotazioni, …), l’elenco degli studenti iscritti con la possibilità di assegnare una valutazione.

4.9.2 Stimulus/Response Sequences

Precondizioni:

- L’utente è autenticato.
- È presente almeno un corso associato al docente.

Sequenza:

1. Il docente visualizza la sezione dedicata agli esami.
2. Il docente clicca su “Visualizza”, relativo all’appello di interesse.

3. Il sistema mostra tutte le informazioni relative all'appello d'esame e mostra anche un elenco di studenti iscritti.
4. Il docente inserisce per ogni studente la presenza, il voto ed eventuali indicatori e/o argomenti della valutazione.
5. Il sistema verifica la correttezza e la completezza dei dati inseriti.
    - In caso di errore, mostra un messaggio e richiede la correzione.
    - In caso di esito positivo, registra le valutazioni nel sistema.
6. Il sistema aggiorna la lista esami del corso selezionato e notifica che l'operazione è avvenuta con successo.

4.9.3 Functional Requirements

- **REQ-1**: Il sistema deve mostrare una lista degli esami creati dal docente.
- **REQ-2**: Il sistema deve consentire la visualizzazione dell'elenco degli studenti iscritti a ogni esame.
- **REQ-3**: Il sistema deve permettere l'inserimento del voto e l'attribuzione della lode.
- **REQ-4**: Il sistema deve permettere l'inserimento di eventuali indicatori di performance e di eventuali argomenti della valutazione.
- **REQ-5**: Il sistema deve validare i dati inseriti nel modulo prima di registrarli nel database.
- **REQ-6**: Il sistema deve salvare i voti, associandoli correttamente all'esame e allo studente.
- **REQ-7**: Il sistema deve mostrare messaggi di errore chiari in caso di input non valido o problemi con il database.

4.10 System Feature: Visualizzazione statistiche

4.10.1 Description and Priority

Questa funzionalità consente al docente di visualizzare statistiche relative ai corsi e agli esami. Le statistiche possono includere il numero di studenti iscritti, la distribuzione dei voti, la media, la percentuale di ammissioni.

4.10.2 Stimulus/Response Sequences

Precondizioni:

- L'utente deve essere autenticato come docente.
- Devono esistere dati nel database relativi a corsi o esami.

Sequenza:

1. Il docente accede al sistema e alla dashboard.
2. Clicca sulla sezione "Statistiche".
3. Il sistema recupera i dati aggregati dal database.
4. Il sistema visualizza le statistiche sotto forma di tabelle o grafici.

4.10.3 Functional Requirements

- **REQ-1:** Il sistema deve rendere disponibile la sezione "Statistiche" solo agli utenti autenticati come docenti.
- **REQ-2:** Il sistema deve estrarre dal database i dati aggregati relativi a:
  - Numero studenti.
  - Distribuzione dei voti.
  - Media dei voti.
- **REQ-3:** Il sistema deve visualizzare le statistiche in formato leggibile.
- **REQ-4:** Il sistema deve gestire eventuali errori o assenze di dati mostrando un messaggio chiaro all'utente**.**

4.11 System Feature: Logout

4.11.1 Description and Priority

Questa funzionalità permette agli utenti di effettuare il logout dal sistema. Al logout, la sessione dell'utente viene terminata e l'utente viene reindirizzato alla pagina di login.

4.11.2 Stimulus/Response Sequences

Precondizioni:

- L'utente deve essere autenticato come docente.

Sequenza:

1. Il docente clicca su "Logout".
2. Il sistema termina la sessione dell'utente.
3. Il sistema reindirizza l'utente alla pagina di login.
4. L'utente visualizza la pagina di login.

4.11.3 Functional Requirements

- **REQ-1**: Il sistema deve terminare la sessione dell'utente al logout.
- **REQ-2**: Il sistema deve reindirizzare l'utente alla pagina di login dopo il logout.
- **REQ-3**: Il sistema deve garantire che tutte le informazioni della sessione vengano rimosse in modo sicuro.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

La web application dovrà garantire prestazioni accettabili e tempi di risposta coerenti con un uso efficiente da parte degli utenti finali.

- Il sistema deve rispondere rapidamente alle interazioni utente, assicurando tempi di attesa contenuti per tutte le operazioni principali come login, visualizzazione di dati, inserimento e modifica.
- Le operazioni di elaborazione lato server, come la generazione di report o la registrazione di valutazioni, devono essere gestite in modo da minimizzare l'attesa percepita dall'utente.
- Il sistema deve supportare l'accesso concorrente da parte di più utenti, garantendo la coerenza dei dati e l'integrità delle operazioni eseguite.
- Il sistema deve adottare transazioni atomiche per garantire che operazioni critiche come l'attribuzione della matricola ad un nuovo utente non venga compromessa da operazioni concorrenti.

- Il sistema deve essere in grado di gestire un numero ragionevole di accessi contemporanei senza rallentamenti evidenti o errori di carico.
- L'applicazione deve assicurare la compatibilità con browser moderni.
- L'interfaccia utente deve essere progettata per evitare un utilizzo eccessivo delle risorse hardware.

5.2 Safety Requirements

Il sistema deve essere progettato per prevenire la compromissione dei dati e danni al sistema. I seguenti requisiti di sicurezza devono essere rispettati:

- Devono essere implementati meccanismi robusti di validazione lato client e lato server per prevenire l'inserimento di dati incoerenti.
- Il sistema deve impedire operazioni potenzialmente distruttive o sensibili, come cancellazione di esami, rimozione di notifiche, modifica di voti, da parte di utenti non autorizzati.

5.3 Security Requirements

Il sistema deve garantire un elevato livello di sicurezza e protezione della privacy per tutti i dati gestiti. I seguenti requisiti devono essere rispettati:

- L'accesso al sistema deve essere consentito solo agli utenti autenticati tramite credenziali personali. Le password devono essere archiviate in forma cifrata.
- Ogni funzionalità del sistema deve essere accessibile solo agli utenti autorizzati in base al loro ruolo.
- L'interfaccia utente e il backend devono implementare controlli per limitare l'accesso a funzionalità e dati.
- Il sistema deve rispettare il Regolamento Generale sulla Protezione dei Dati (GDPR).
- Tutte le comunicazioni tra client e server devono avvenire tramite protocollo HTTPS per garantire la confidenzialità e l'integrità dei dati trasmessi.

5.4 Software Quality Attributes

Il sistema dovrà soddisfare i seguenti attributi qualitativi:

- Usabilità: l'interfaccia utente deve essere semplice, intuitiva e coerente.
- Manutenibilità: il codice sorgente dovrà essere organizzato e documentato in modo da facilitare interventi futuri di correzione, aggiornamento o estensione.

- Affidabilità: il sistema deve funzionare in modo stabile e gestire correttamente situazioni anomale, come input non validi, errori di rete, evitando comportamenti imprevisti.
- Disponibilità: il sistema deve essere disponibile agli utenti in modo continuativo.
- Testabilità: i singoli componenti devono essere progettati per facilitare il testing automatico e manuale.
- Portabilità: il sistema dovrà essere in grado di operare su diversi ambienti server compatibili.

6. Database Design

6.1 Conceptual Design

- **La relazione che intercorre tra UTENTE e NOTIFICA è di tipo molti a molti (N:N)** poiché un Utente può ricevere una o più Notifiche, ogni Notifica può essere ricevuta da più Utenti.
- **La relazione che intercorre tra DOCENTE e CORSO è di tipo uno a molti (1:N)** poiché un Docente può insegnare uno o più Corsi, ma ogni Corso è impartito da un solo Docente.
- **La relazione che intercorre tra CORSO e ESAME è di tipo uno a molti (1:N)** poiché un Corso può avere opzionalmente una o più prove d'Esame associate, ma ogni Esame è relativo ad un solo Corso.

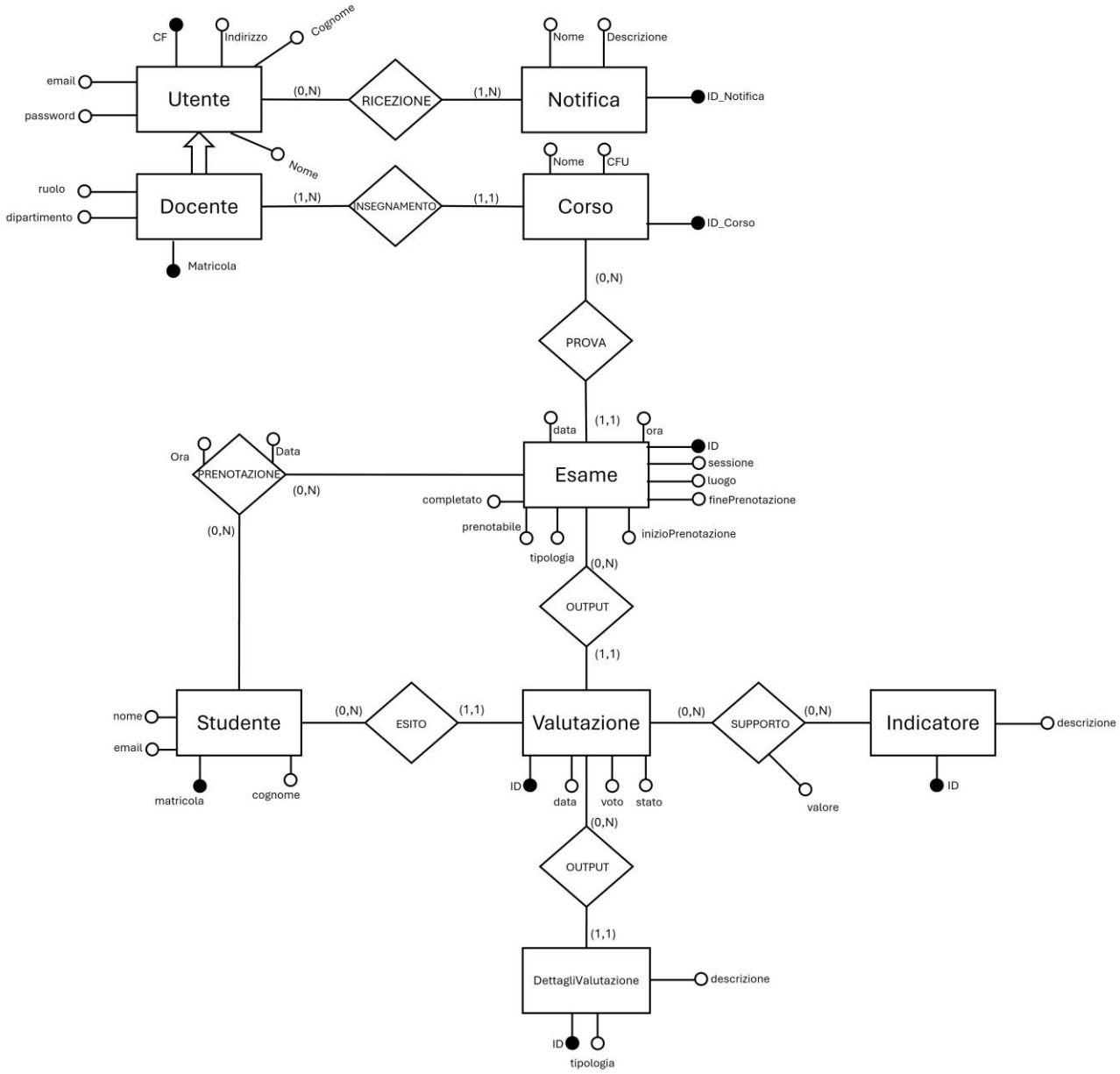

Figura 6 ER Diagram - Conceptual

- **La relazione che intercorre tra ESAME e STUDENTE è di tipo molti a molti (N:M)** poiché un Esame può avere più Studenti prenotati e uno Studente può prenotarsi a più Esami.
- **La relazione che intercorre tra ESAME e VALUTAZIONE è di tipo uno a molti (1:N)** poiché ad un Esame si può associare opzionalmente una o più Valutazioni e una Valutazione si riferisce a un solo Esame.
- **La relazione che intercorre tra STUDENTE e VALUTAZIONE è di tipo uno a molti (1:N)** poiché uno Studente può avere più Valutazioni relativamente ad esami diversi e una Valutazione si riferisce a un solo Studente.
- **La relazione che intercorre tra VALUTAZIONE e INDICATORE è di tipo molti a molti (N:M)** poiché una Valutazione può avere più Indicatori e un Indicatore può essere associato a più Valutazioni.
- **La relazione che intercorre tra VALUTAZIONE e DETTAGLIVALUTAZIONE è di tipo uno a molti (1:N)** poiché ad una Valutazione possono essere associati opzionalmente uno o più DettagliValutazione (intesi come argomenti della valutazione) e ogni DettaglioValutazione è relativo a una singola Valutazione.

NB: in questo sottosistema progettato, l'entità "Studente" non è stata generalizzata nell'entità "Utente". Questo perché il sistema prevede il "Docente" come unico tipo di utente attivo. Gli studenti, pur essendo registrati nel sistema per scopi di tracciamento degli esiti e delle valutazioni, non accedono o interagiscono con le funzionalità del sistema come farebbe un utente loggato. La loro presenza nel diagramma è esclusivamente per associare esiti e valutazioni ai rispettivi studenti.

6.2 Logical Design

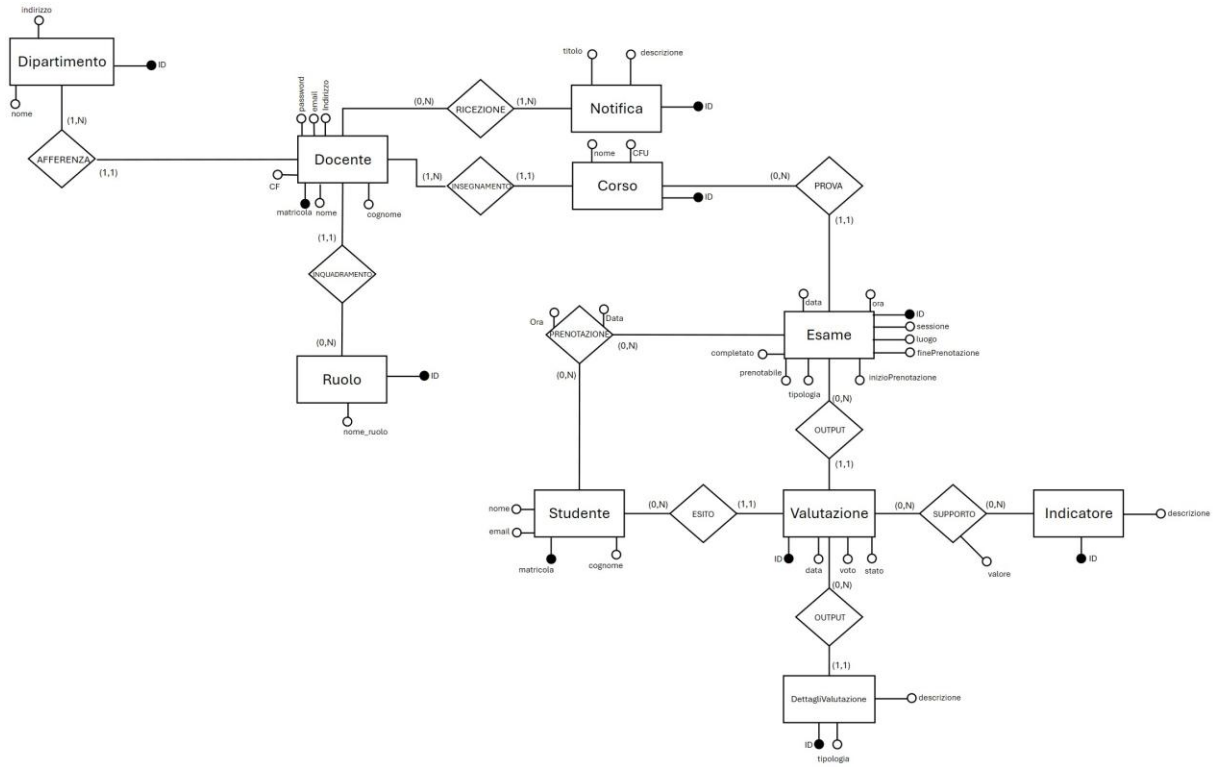

Figura 7 ER Diagram - restructured

Dipartimenti (**id**, nome, indirizzo)

Ruoli (**id**, nome_ruolo)

Docenti (**matricola**, nome, cognome, cf, indirizzo, email, password, ***dipartimento***, ***ruolo***)

Notifiche (**id**, titolo, descrizione)

Ricezioni (**docente**, **notifica**)

Corsi (**id**, nome, cfu, ***docente***)

Esami (**id**, data, ora, sessione, luogo, inizioPrenotazione, finePrenotazione, prenotabile, completato, ***corso***)

Valutazioni (**id**, data, voto, stato, lode, ***esame***, ***studente***)

Studenti (**matricola**, nome, cognome, email)

Prenotazioni (***studente***, ***esame***, data, ora)

Indicatori (**id**, descrizione)

Supporto (***valutazione***, ***indicatore***, valore)

DettagliValutazione (**id**, tipologia, descrizione, ***valutazione***)

6.3 Implementation

6.3.1 Table creation

```
CREATE TABLE IF NOT EXISTS Dipartimenti (
id int NOT NULL AUTO_INCREMENT,
nome varchar(255) NOT NULL,
PRIMARY KEY(id)
);
```

```
CREATE TABLE IF NOT EXISTS Ruoli (
id int NOT NULL AUTO_INCREMENT,
nome_ruolo varchar(255) NOT NULL,
PRIMARY KEY(id)
);

CREATE TABLE IF NOT EXISTS Docenti (
matricola varchar(20) NOT NULL,
nome varchar(255) NOT NULL,
cognome varchar(255) NOT NULL,
cf varchar(16) NOT NULL UNIQUE,
email varchar(255) NOT NULL UNIQUE,
password varchar(255) NOT NULL,
indirizzo varchar(255) DEFAULT NULL,
ruolo int NOT NULL,
dipartimento int NOT NULL,
PRIMARY KEY(matricola),
FOREIGN KEY(ruolo) REFERENCES Ruoli(id),
FOREIGN KEY(dipartimento) REFERENCES Dipartimenti(id)
);
CREATE TABLE IF NOT EXISTS Notifiche (
id int NOT NULL AUTO_INCREMENT,
titolo varchar(255) NOT NULL,
descrizione varchar(255) NOT NULL,
data_creazione TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
PRIMARY KEY(id)
);
```

```
CREATE TABLE IF NOT EXISTS Studenti (
matricola varchar(20) NOT NULL,
nome varchar(255) NOT NULL,
cognome varchar(255) NOT NULL,
email varchar(255) NOT NULL UNIQUE,
PRIMARY KEY(matricola)
);
CREATE TABLE IF NOT EXISTS Ricezioni (
docente varchar(20) NOT NULL,
notifica int NOT NULL,
PRIMARY KEY(docente, notifica),
FOREIGN KEY(notifica) REFERENCES Notifiche(id),
FOREIGN KEY(docente) REFERENCES Docenti(matricola)
);
CREATE TABLE IF NOT EXISTS Corsi (
id int NOT NULL AUTO_INCREMENT,
nome varchar(255) NOT NULL,
cfu TINYINT NOT NULL,
docente varchar(20) NOT NULL,
PRIMARY KEY(id),
FOREIGN KEY(docente) REFERENCES Docenti(matricola)
);
CREATE TABLE IF NOT EXISTS Esami (
id int NOT NULL AUTO_INCREMENT,
data date NOT NULL,
ora time NOT NULL,
tipologia ENUM('scritto', 'orale', 'scritto/orale') NOT NULL,
```

```
sessione ENUM('Straordinaria', 'Estiva', 'Invernale') NOT NULL,
luogo varchar(255) DEFAULT NULL,
inizioPrenotazione date NOT NULL,
finePrenotazione date NOT NULL,
prenotabile Boolean NOT NULL,
completato Boolean NOT NULL,
corso int NOT NULL,
PRIMARY KEY(id),
FOREIGN KEY(corso) REFERENCES Corsi(id),
CHECK(inizioPrenotazione < finePrenotazione),
CHECK(finePrenotazione < data),
UNIQUE(corso, data, ora)
);
CREATE TABLE IF NOT EXISTS Valutazioni (
id int NOT NULL AUTO_INCREMENT,
data date NOT NULL,
voto tinyint NOT NULL CHECK (voto = 0 OR (voto >= 18 AND voto <= 30)),
lode Boolean NOT NULL CHECK ((lode = FALSE) OR (lode = TRUE AND
voto = 30)),
stato ENUM('Presente', 'Assente') NOT NULL,
ammesso Boolean NOT NULL,
esame int NOT NULL,
studente varchar(20) NOT NULL,
PRIMARY KEY(id),
CONSTRAINT valutazione_unica UNIQUE (esame, studente),
FOREIGN KEY(esame) REFERENCES Esami(id),
FOREIGN KEY(studente) REFERENCES Studenti(matricola),
```

```
CHECK ((ammesso = FALSE) OR (ammesso = TRUE AND stato =
'Presente'))

);

CREATE TABLE IF NOT EXISTS Prenotazioni (

studente varchar(20) NOT NULL,

esame int NOT NULL,

data date NOT NULL,

ora time NOT NULL,

PRIMARY KEY(studente, esame),

FOREIGN KEY(esame) REFERENCES Esami(id),

FOREIGN KEY(studente) REFERENCES Studenti(matricola)

);

CREATE TABLE IF NOT EXISTS Indicatori (

id int NOT NULL AUTO_INCREMENT,

descrizione varchar(255) NOT NULL UNIQUE,

PRIMARY KEY(id)

);

CREATE TABLE IF NOT EXISTS Supporto (

valutazione int NOT NULL,

indicatore int NOT NULL,

valore tinyint NOT NULL CHECK(valore >= 1 AND valore <= 10),

PRIMARY KEY(valutazione, indicatore),

FOREIGN KEY(valutazione) REFERENCES Valutazioni(id),

FOREIGN KEY(indicatore) REFERENCES Indicatori(id)

);

CREATE TABLE IF NOT EXISTS DettagliValutazione (

    id int NOT NULL AUTO_INCREMENT,

    tipologia ENUM('scritto', 'orale') NOT NULL,
```

```
    descrizione TEXT NOT NULL,
    valutazione int NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(valutazione) REFERENCES Valutazioni(id)
);
```

6.3.2 Procedures and events

Per garantire l'aggiornamento automatico e puntuale della disponibilità alla prenotazione degli esami, il sistema utilizza una **procedura** in combinazione con un evento pianificato.

La procedura *aggiorna_prenotabilita_esami* contiene la logica necessaria a determinare se un esame debba essere considerato prenotabile, confrontando la data corrente con l'intervallo di prenotazione definito per ciascun esame. Viene eseguito un aggiornamento sul campo prenotabile in base a queste condizioni. Questa soluzione automatizza un'operazione ricorrente, garantisce la coerenza dello stato del database ed elimina la necessità di interventi manuali.

```
DROP PROCEDURE IF EXISTS aggiorna_prenotabilita_esami;
DELIMITER //
CREATE PROCEDURE aggiorna_prenotabilita_esami()
BEGIN
   UPDATE esami
   SET prenotabile = CASE
      WHEN CURDATE() >= inizioPrenotazione AND CURDATE() <=
finePrenotazione THEN 1
      ELSE 0
   END;
END //
DELIMITER ;


SET GLOBAL event_scheduler = ON;
```

```
CREATE EVENT IF NOT EXISTS aggiorna_prenotabilita_esami
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_DATE + INTERVAL 0 SECOND
DO
  CALL aggiorna_prenotabilita_esami();
END;
```

7. Test

Questo capitolo descrive le strategie adottate per garantire che il sistema soddisfi i requisiti funzionali e non funzionali, attraverso test automatici e test manuali.
L'obiettivo è identificare eventuali malfunzionamenti e garantire un comportamento coerente dell'applicazione.
Al fine di garantire l'affidabilità, la correttezza e la robustezza di tali funzioni, è stata sviluppata una suite di test unitari utilizzando il framework **PHPUnit**. I test sono stati progettati secondo un approccio **white-box**, con l'obiettivo di coprire i principali percorsi logici delle funzioni, incluse le condizioni di successo e di errore. Poiché le funzioni interagiscono direttamente con il database, sono stati utilizzati oggetti **mock** per simulare il comportamento delle classi mysqli, mysqli_stmt e mysqli_result, consentendo così di eseguire i test in completa assenza di un database reale.

7.1 PHPUnit Test

Di seguito sono descritte le funzioni PHP sottoposte a test unitari con PHPUnit. Tutte le funzioni condividono una struttura simile: accettano una connessione al database, un identificativo (ovvero la matricola del docente) e una variabile di output per gestire messaggi d'errore.

7.1.1 getEsami(mysqli $conn, string $docente, string &$message): array

Questa funzione restituisce un array di esami associati a un determinato docente. Esegue una query al database per ottenere informazioni sugli esami. In caso di errore nella query, imposta un messaggio nella variabile $message.

7.1.2 getCategorieEsami(mysqli $conn, mysqli_result $esami, string &$message): array

Dopo aver ottenuto la lista di esami, questa funzione analizza le categorie di ciascun esame e restituisce una mappatura delle categorie presenti. La funzione è utile per costruire filtri dinamici per categoria. Richiede in input la lista di esami già ottenuta da getEsami.

7.1.3 getDatiDocente(mysqli $conn, string $matricola, string &$message): array

Recupera i dati anagrafici e istituzionali di un docente. In caso di errore, aggiorna $message con un messaggio coerente con l'esito.

7.1.4 getStatsDocente(mysqli $conn, string $matricola, string &$message): array

Esegue query per calcolare statistiche sugli esami gestiti dal docente. Può includere il numero di esami svolti, il numero di studenti promossi e altri parametri utili. Restituisce un oggetto *mysqli_result* con le statistiche, mentre la variabile $message comunica eventuali problemi o conferme.

7.1.5 getEsamiProgrammati(mysqli $conn, string $matricola, string &$message): array

Recupera gli esami programmati per il docente specificato, includendo data, orario, luogo e corso associato. Eventuali problemi vengono comunicati tramite $message.

7.1.6 getDistribuzioneVoti(mysqli $conn, string $matricola, string &$message): array

Analizza i voti assegnati agli studenti negli esami tenuti dal docente e determina la distribuzione in percentuale dei voti. In caso di errore, viene impostato un messaggio nella variabile $message.

7.2 Manual client-side validation test

Obiettivo	*Prerequisiti*	*Sequenza*	*Dati di input*	*Risultato atteso*
Verificare la validazione di email e password prima dell'invio del form di login	Pagina di login caricata correttamente	1. Inserire email vuota o non conforme al dominio @university.it 2. Inserire password vuota o non conforme ai criteri 3. Premere il pulsante di login	- email: "", "user@gmail.com", "user@university.com" - password: "", "pass", "Password1"	Visualizzazione del messaggio di errore specifico e blocco dell'invio finché i dati non sono validi
Verificare la validazione dei campi del form di registrazione prima dell'invio	Pagina di registrazione caricata correttamente	1. Inserire nome e cognome con meno di 2 caratteri o contenenti caratteri non validi 2. Inserire codice fiscale non valido 3. Inserire email senza dominio "@university.it" 4. Inserire password non conforme (almeno 8 caratteri, una maiuscola, una minuscola, una cifra, un simbolo) 5. Non selezionare dipartimento o ruolo 6. Premere il pulsante di registrazione	Esempio, - codice fiscale: "ABC123", "123456789" - email: "utente@gmail.com", "user@university.com"	Visualizzazione del messaggio di errore specifico al primo campo non valido e blocco dell'invio del form finché non viene corretto
Validare la compilazione e la coerenza dei dati nei form di voto prima dell'invio al server	Pagina "Visualizza esame" con form di voti, indicatori e argomenti caricata correttamente	1. Lasciare vuoto il campo "voto" 2. Lasciare vuoto un indicatore o argomento creato 3. Premere il pulsante salva	- Campo voto vuoto - Indicatori senza valore o descrizione - Argomenti senza tipologia o descrizione	Visualizzazione alert che segnala l'errore e blocco della procedura di salvataggio.
Verificare che le date inserite rispettino le condizioni logiche tra loro	Pagina "Nuovo esame" caricata correttamente	1. Inserire una data esame precedente a oggi 2. Inserire dataFine ≤ dataInizio 3. Inserire dataFine > dataEsame 4. Inviare il form	1. dataEsame < oggi 2. dataFine = dataInizio o dataFine < dataInizio 3. dataFine > dataEsame	Viene mostrato messaggio di errore e il form non può essere inviato finché i dati non sono corretti